Livia Oushiro

Introdução à Estatística para Linguistas

EDITORA DA **ABRALIN**

Palavra dos editores

Esta publicação, digital e gratuita, compõe o catálogo de livros digitais da Editora da ABRALIN, uma editora *open access*, criada em 2020, que busca oferecer mecanismos efetivos de publicação e circulação de obras de Linguística no país. A ideia que norteia seu funcionamento encontra melhor expressão nas palavras de seu idealizador, Prof. Dr. Miguel Oliveira Jr., então presidente da ABRALIN: "acreditamos que dar acesso livre à produção intelectual de excelência, que é fruto – na maioria das vezes – de investimento público, é o caminho mais democrático no contexto socioeconômico em que vivemos". Sem dúvida, essas palavras foram definitivas para o nosso engajamento na criação da Editora da ABRALIN. Queremos contribuir para fazer da Editora da ABRALIN um canal permanente de apoio à divulgação da sólida pesquisa feita nas muitas áreas da Linguística no Brasil.

Como todos sabemos, a ABRALIN desempenha papel fundamental na consolidação dos estudos linguísticos no Brasil, contribuindo de maneira crucial para a criação e a preservação de espaços de acolhimento da diversidade de ideias linguísticas, algo que tem urgência ética e é – no nosso entendimento – atitude necessária para manter o indispensável diálogo entre a sociedade e a comunidade científica. A Editora da ABRALIN nasce dentro desse contexto e com esse desígnio maior.

A excelência do trabalho da Editora e das obras por ela publicadas será garantida – disso temos certeza – pela esperada contribuição dos associados da ABRALIN. Tal contribuição constantemente vem em atendimento aos editais e aos critérios tornados públicos periodicamente, na forma de propostas de publicação, na colaboração junto ao Conselho Editorial e com as demais atividades envolvidas no funcionamento da Editora.

Nossa expectativa é que a Editora da ABRALIN possa fornecer obras de qualidade, acessíveis gratuitamente ao público-leitor interessado, fomentando, assim, a pesquisa em Linguística, contribuindo com o diálogo constante entre pesquisadores e sociedade.

Valdir do Nascimento Flores
Gabriel de Ávila Othero
Editores

Introdução à Estatística para Linguistas

Livia Oushiro

EDITORA DA **ABRALIN**

SUMÁRIO

PREFÁCIO

Elisa Battisti (UFRGS/CNPq)

Este livro reúne lições de estatística para linguistas, disciplina necessária a todos os estudiosos da linguagem que, com análise quantitativa, investigam a recorrência das formas linguísticas no desempenho (*performance*), na produção e percepção da língua.

Nascido como um manual para um curso de pós-graduação em linguística, *Introdução à Estatística para Linguistas*, de Livia Oushiro, interessa tanto àqueles que buscam iniciar-se na análise quantitativa de dados linguísticos quanto àqueles que, como eu, vinham usando o programa Goldvarb em análises sociolinguísticas variacionistas e querem aprender a usar um programa computacional com mais opções de análise.

O livro, assim, atende a dois possíveis anseios dos interessados: introduz os leitores à estatística e ao R. Foi escrito no pacote swirl, um ambiente didático da plataforma R para a aprendizagem dessa linguagem de programação a partir de sua aplicação prática. Cada lição de estatística contempla explicações, exemplos e exercícios que o leitor resolve no programa R e para os quais obtém retorno imediato, sobre a precisão do cálculo estatístico efetuado. Além disso, as lições fornecem os *scripts* usados na plataforma R para as análises estatísticas e uma lista de referências para aprofundar os estudos de estatística e da linguagem R.

Introdução à Estatística para Linguistas subsidia os pesquisadores do planejamento à execução da análise quantitativa. Na obra, nós, pesquisadores, professores e alunos de pós-graduação encontramos orientações seguras e fundamentação consistente para a condução das análises. Alegro-me ao vê-la agora publicada no formato *e-book*, chancelada pela ABRALIN e disponível a um número maior de usuários. É, sem dúvida, uma grande contribuição à pesquisa linguística.

APRESENTAÇÃO

Este manual de Introdução à Estatística para Linguistas[1] foi criado originalmente como um curso a ser oferecido no Programa de Pós-Graduação em Linguística do Instituto de Estudos da Linguagem da UNICAMP em 2017. Sua primeira versão foi escrita no pacote swirl, um ambiente didático, dentro da plataforma R, para a aprendizagem dessa linguagem de programação a partir de sua aplicação prática. Desde então, o curso também tem sido ofertado como curso de extensão na UNICAMP, em dezenas de minicursos em programas de pós-graduação em todo o Brasil e como curso *on-line* assíncrono na Plataforma EAD da ABRALIN.[2]

A apresentação deste tutorial em formato de *e-book* tem o objetivo de torná-lo acessível a um público mais amplo, assim como facilitar sua consulta sempre que necessário. Aos interessados e às interessadas em se iniciar na análise quantitativa de dados linguísticos, recomendo seguir o curso em seu formato swirl, dentro da plataforma R (ver Lição 0). Dentro do ambiente didático do swirl, o usuário digita as respostas às questões e imediatamente tem um retorno sobre sua precisão – se a resposta está incorreta, o swirl pede que se tente novamente. Neste *e-book*, o texto vem acompanhado do código que pode ser reproduzido dentro do R, de modo que também é possível segui-lo fora do ambiente swirl.

Tenho testemunhado, nos últimos anos, uma procura cada vez maior por materiais e discussões sobre tratamento estatístico de dados linguísticos. Contudo, até o momento, a formação em Estatística não é componente obrigatório da maior parte dos cursos de Letras e de Linguística, e tem de ser buscada por iniciativa própria de

---

[1] Esta versão do manual corresponde à versão 2.0.3 do curso *Introdução à Estatística para Linguistas*, de 12 mai. 2021 (disponível em https://doi.org/10.5281/zenodo.4755739). Este *e-book* foi escrito em formato .Rmd, na versão 4.1.1 "Kick Things" (10/08/2021) do R e 2021.09.0+351 "Ghost Orchid" (20/09/2021) da interface RStudio.

[2] Disponível em https://ead.abralin.org/. Último acesso em 14 mai. 2022.

pesquisadores, professores e estudantes de pós-graduação ou graduação. O presente tutorial foi pensado para que possa ser útil mesmo em situação de estudo autônomo, mas seu aproveitamento certamente será melhor se o estudo for realizado em grupos de estudo ou em cursos regulares.

Linguistas precisam de Estatística?

Sempre que se lida com variação, o tratamento estatístico de dados é bem-vindo. Diversos aspectos linguísticos são variáveis, na Fonética, Fonologia, Morfologia, Sintaxe, Léxico, Discurso etc. Por exemplo, nenhum falante produz uma vogal /e/ exatamente do mesmo modo que outros falantes – com efeito, nem o mesmo indivíduo o faz da mesma maneira em todas as ocasiões. Até as intuições dos falantes são variáveis quanto à gramaticalidade, adequação ou interpretação de certas estruturas em determinados contextos. O linguista que trabalha com dados empíricos (de usos, avaliações, julgamentos de adequação, gramaticalidade, tempos de reação etc.) inevitavelmente se depara com variação.

Análises estatísticas não concorrem com, nem substituem análises qualitativas: ambas são importantes para uma compreensão mais global do funcionamento das línguas naturais. O objetivo da análise estatística não é – e nunca deveria ser! – a quantificação de dados por si só. As análises estatísticas têm três propósitos principais: resumir, explicar e prever.

Quando se lida com variação, é desejável ter uma amostra representativa do que se está analisando, o que muitas vezes implica um conjunto de dados maior do que aquilo de que somos capazes de fazer sentido "a olho (ou ouvido?) nu", digamos. Com mais de 50, 100 dados, dificilmente é possível depreender um padrão mais geral que não seja impressionístico, sob forte risco de viés de acordo com nossas próprias expectativas ou previsões. Por meio de gráficos ou tabelas, é possível resumir um conjunto extenso de

dados para dimensões inteligíveis, sobre as quais podemos fazer alguma afirmação mais segura.

Tabelas e gráficos permitem descrever o que ocorre nos dados, mas isoladamente não permitem explicá-los: a explicação sobre certos padrões deve sempre estar alinhada às teorias e aos modelos linguísticos com que se trabalha. Análises estatísticas, no entanto, podem auxiliar na interpretação de padrões ao revelar correlações entre aspectos concomitantes. É claro que encontrar uma correlação não é sinônimo de ter encontrado a causa motivadora para determinado fenômeno, mas perceber que certas variáveis coocorrem sistematicamente pode fornecer pistas valiosas para explicar o que está por trás das estruturas observadas. Constatar, por exemplo, que a vogal pretônica /e/ é realizada mais frequentemente como [i] quando a sílaba seguinte contém uma vogal [+alta] (como *menino*) do que quando contém uma vogal [-alta] (como em *metade*), mesmo que essa relação não seja categórica (também é possível dizer *t*[i]*atro* para *teatro*), diz-nos algo sobre os fatores possivelmente em operação na articulação de vogais pretônicas.

Prever, por sua vez, não significa (necessariamente) determinar o que vai acontecer no futuro, próximo ou distante, mas ser capaz de extrapolar as conclusões para além do conjunto de dados específicos observados. Afinal, na maior parte das vezes, não temos interesse em fazer afirmações apenas sobre a língua falada pelas poucas dezenas de voluntários de um estudo, mas sobre a língua de um conjunto muito maior de falantes, e possivelmente – em comparação com outros trabalhos – sobre o funcionamento das línguas de modo geral. Para fazer previsões, é necessário criar modelos plausíveis, que simulam o funcionamento do que está sendo investigado numa escala manejável para nossa compreensão. Por exemplo, em Cinemática, a fórmula para o movimento retilíneo uniforme ($v = \Delta S / \Delta t$, ou seja, a velocidade média é igual à razão entre o deslocamento espacial e a diferença entre tempo final e inicial) permite prever que um objeto em movimento a 100 km/h, em uma hora, terá se deslocado 100 km; ou que minha viagem de São Paulo a Campinas deve levar cerca de uma hora, se eu mantiver a velocidade média de 100 km/h, ainda que, na prática, eu dificilmente terei mantido a velocidade constante

durante todo o percurso, tampouco terei me deslocado apenas retilineamente do ponto inicial ao final. Só é possível fazer previsões a respeito de fenômenos que não são totalmente aleatórios – e a língua, certamente, é um sistema estruturado.

Os manuais básicos de Estatística costumam fazer uma divisão entre Estatística Descritiva e Estatística Inferencial, estrutura que o presente livro também segue, mas vale reforçar que não se trata de análises separadas quando se trata de um mesmo conjunto de dados. Todas as análises devem convergir para que se possa entender o que está acontecendo nos dados e se possa fazer afirmações relevantes – em nosso caso, sobre o funcionamento linguístico. Análises estatísticas, quando feitas conscientemente, permitem detectar padrões mais ou menos robustos, estimar o peso de diferentes fatores sobre determinado fenômeno variável (ou seja, não exagerar o papel de certos fatores frente a outros), e evitar generalizações equivocadas. Não é raro ainda hoje encontrar trabalhos que insistem em afirmar que, por exemplo, as mulheres favoreceram o uso da forma padrão apenas com base em proporções, mesmo quando não se verificou uma diferença significativa a partir de modelo de regressão. Aprender a desenvolver análises estatísticas implica entender e respeitar os passos da análise científica, mesmo que isso contrarie as expectativas iniciais.

Além do devido manejo de dados empíricos, conhecer diferentes tipos de representação gráfica, sistematização de dados, testes e modelos estatísticos fornece ao pesquisador maior liberdade para condução de suas pesquisas, buscando novas questões. Em muitas pesquisas verifica-se a reprodução de uma mesma pergunta que é feita repetidas vezes, apenas com pequenos ajustes. Para dar um exemplo de minha própria área de pesquisa, muitos estudos sociolinguísticos se voltam à descrição de padrões de uma variável "X" (p.ex., concordância nominal, pronúncia de /r/ em coda, vogais médias pretônicas etc.) em uma comunidade "Y" (em geral, definida como uma cidade, mas também por agrupamentos menores como comunidades de práticas); sociolinguistas interessados em outros tipos de análises, como de avaliações ou percepções, muitas vezes acabam não as realizando por desconhecimento de como coletar e tratar os dados. Desse modo, é de grande valia buscar novos tipos de análise, mesmo que não se apliquem no

atual conjunto de dados com que se está trabalhando, com vistas ao alargamento de possibilidades de análises futuras e ao desenvolvimento do campo.

Antes das análises estatísticas

As lições nesta obra pressupõem que o linguista já está pronto para realizar a análise estatística de seus dados. Entretanto, antes de chegar a esse ponto, o pesquisador certamente terá percorrido uma lista de tarefas, que inclui a coleta de dados e sua organização em uma planilha. Nesse sentido, embora tratar dessas tarefas não seja o intuito da obra, vale aqui fazer alguns breves apontamentos sobre aspectos dessas etapas que são pertinentes às posteriores análises quantitativas.

Análises quantitativas quase sempre operam sobre *amostras* de dados, ou seja, uma parcela de um universo bastante maior do que poderia ter sido coletado: a *população*. Em Estatística, *população* é um termo técnico, que se refere ao conjunto total de itens pertinentes ao objeto sob estudo – ou seja, não se trata necessariamente de pessoas. Por exemplo, em um estudo sobre a realização variável de /r/ em posição de coda silábica na fala de paulistanos, a população é o conjunto de todas as ocorrências de /r/ em final de sílaba – *porta*, *caderno*, *mulher*, *celular*, *porque* etc. – produzidas por pessoas que nasceram na cidade de São Paulo. Em um caso como este, é evidente que seria impossível trabalhar com toda a população de dados, pois os paulistanos produziram, produzem e continuarão produzindo outras tantas ocorrências de /r/ em coda para além da janela de observação do pesquisador.

As perguntas que surgem, então, são "de quantos dados preciso, no mínimo?" ou "de quantos participantes preciso, no mínimo?". Não há resposta única para essas perguntas, pois o cálculo depende do grau de variabilidade do objeto de estudo, do número de hipóteses (operacionalizadas em variáveis previsoras) que se deseja testar, e também de questões práticas à pesquisa, como o tempo de que se dispõe para recrutamento de participantes e coleta de dados, capacidade de processamento dos dados

coletados etc. No entanto, um caminho para estimar o tamanho mínimo de uma amostra é realizar um teste de *poder estatístico* (ver p.ex., Crawley 2013).

Por outro lado, também é importante considerar a *representatividade* da amostra. Se se coletam dados apenas com falantes paulistanos, ou apenas com falantes universitários, há que se ter cautela ao generalizar os resultados para o "Português Brasileiro", a depender do que se está examinando. Ao mesmo tempo, a possibilidade de se chegar a generalizações, ainda que mais modestas, sobre o "português paulistano" ou sobre o "português culto" depende do modo de recrutamento dos participantes: a amostra será tanto mais representativa se tiver sido aleatória, ou seja, se todos os itens da população tiveram a mesma chance de ser selecionados. Ora, sabemos que este dificilmente é o caso em estudos linguísticos: em geral, coletam-se dados de voluntários razoavelmente próximos de nossas redes sociais.

A dificuldade em se obter amostras verdadeiramente aleatórias e representativas, contudo, não implica que não podemos confiar em nossos resultados. A *replicabilidade* e a *comparação* entre resultados de diferentes estudos são os aspectos acadêmicos que permitem uma avaliação constante da validade das pesquisas. Para esse fim, é imprescindível que os pesquisadores reportem os passos e as decisões tomadas, desde a coleta de dados até seu processamento e análise, para que se possa avaliar a comparabilidade entre estudos e reproduzir procedimentos.

Uma vez que os dados estejam coletados, é necessário sistematizá-los para que possam ser facilmente manipulados e processados. A melhor maneira de organizar os dados digitalmente é em planilhas eletrônicas, em programas como o Excel ou o Calc, que permitem que os dados sejam exportados em diversos formatos (.xlsx, .csv, .txt etc.). Recomendo seguir o formato "caso por variável"[3] apresentado por Gries (2019, p. 29–30) – também chamado de formato *long table* –, cujas convenções são as seguintes:

- A primeira linha contém os nomes das variáveis;

[3] Ver Lição 3 para os conceitos de variável, variantes, níveis de variáveis, tipos de variáveis.

- Cada linha representa um e apenas um dado (uma observação da variável dependente);
- A primeira coluna enumera as *n* observações de 1 a *n*;
- Cada uma das colunas representa uma e apenas uma variável;
- Dados/observações faltantes são indicados por "NA", e não por células vazias.

Além disso, Gries (2019) sugere que se utilize caixa alta para o nome das variáveis; que variáveis nominais nunca sejam codificadas com números (p.ex., para *primeira faixa etária*, é preferível usar "1a" em vez de "1"); e que nunca se utilizem caracteres especiais como espaço, vírgula, marca de tabulação, #, aspas, acentos para codificar variáveis ou variantes/níveis de variáveis.

Cabe notar que quando se coletam dados por meio de questionários *on-line*, o formato acima descrito normalmente não é aquele gerado automaticamente pelas plataformas; neles, cada linha representa os dados de cada formulário respondido (em geral, um para cada participante) – formato conhecido como *wide table*. É possível fazer a conversão do formato *wide* para *long* por meio do pacote tidyr do R e, para isso, há vários tutoriais na Internet.[4]

A plataforma R e o ambiente swirl

O R é uma linguagem de programação voltada para computações gráficas e estatísticas.[5] É uma plataforma livre e gratuita, de código aberto, disponível para Linux, MacOS e Windows, e seus usuários se dispersam pelas mais variadas áreas de conhecimento e aplicação de análise de dados. Especificamente para a Linguística, por meio do R é

[4] Por exemplo, o *Cookbook for R*: http://www.cookbook-r.com/Manipulating_data/Converting_data_between_wide_and_long_format/. Último acesso em 14 mai. 2022.

[5] Disponível em https://cran.r-project.org/. Último acesso em 14 mai. 2022.

possível, por exemplo, processar dados textuais, compilar e anotar *corpora* de dados, criar concordâncias e listas de frequências, converter arquivos em formato .wav para .mp3, além de realizar análises estatísticas e criar representações gráficas dos dados e dos padrões encontrados pelo pesquisador.

Para utilizar os recursos dessa linguagem, pode-se usar a própria interface do R ou a interface RStudio, também gratuita.[6] Independentemente da interface que se usar, no entanto, a plataforma é o R, pois não se trata de um programa, mas de uma linguagem de programação.

Em um primeiro contato com o R, o usuário pode estranhar a falta de botões e opções em que clicar. No R, a maior parte daquilo que se quer executar é feito por meio de *linhas de comando*, algo que, em geral, pode parecer mais trabalhoso do que simplesmente clicar sobre comandos pré-programados em botões. No entanto, a execução de comandos por meio de linhas de códigos traz mais vantagens do que desvantagens. Primeiro, justamente por ser uma linguagem de programação, o usuário não está restrito a opções pré-programadas por um *software* (como o GoldVarb X, SPSS, Excel etc.), cujos programadores dificilmente podem ter previsto tudo o que um usuário pode ter interesse em executar. Segundo, ainda que digitar linhas de comando possa parecer trabalhoso, normalmente isso é feito apenas na primeira vez; na maior parte das vezes, trabalha-se com *scripts*, que são conjuntos de linhas de comando salvas em um arquivo, que podem ser reutilizadas ou adaptadas, parcial ou integramente, em novos usos posteriores. Além disso, programas com botões pré-definidos podem ser antes um risco do que uma vantagem: é grande a chance de que um usuário comece a clicar sobre as opções disponíveis, "para ver no que dá", e acabe desenvolvendo análises que nem sempre fazem sentido. Embora análises absurdas também possam ser realizadas em linguagem de programação, é menor a chance de que um usuário chegue a certo

[6] Disponível em https://www.rstudio.com/products/rstudio/download/. Último acesso em 14 mai. 2022.

resultado aleatoriamente, pois será necessário conscientizar-se da análise adequada a ser desenvolvida antes de executá-la.

Como sociolinguista, posso relatar o que tenho observado com frequência entre colegas da área, que até a década de 1990 ou a primeira década de 2000 costumavam usar o programa Varbrul, em suas diversas implementações – o GoldVarb X sendo a última. Este é um programa de fácil manuseio e de interface (relativamente) amigável, o que permite começar a usá-lo em poucas horas de treinamento. É inegável que a facilidade de uso do GoldVarb X trouxe muitas vantagens à área da Sociolinguística Variacionista, sendo a principal delas o uso de uma ferramenta unificada de tratamento de dados que torna os resultados, em diferentes estudos, razoavelmente comparáveis. Por outro lado, a maior parte dos usuários do GoldVarb X tem apenas um conhecimento técnico de como manejá-lo: sabe que deve preparar os dados de certo modo, inseri-los no programa em determinada janela, e depois selecionar uma série de opções para que o programa "rode" os dados; no outro extremo, saem os resultados, sem garantias de que sejam confiáveis – qualquer sociolinguista experiente tinha ciência da máxima: *garbage in, garbage out*. O GoldVarb X é uma ferramenta útil, contanto que se saiba o que ela é capaz de fazer e o que não é capaz de fazer. É um programa feito para realizar apenas um tipo de análise: *modelos multivariados fixos de regressão logística com variáveis nominais* – ou seja, a variável dependente deve, necessariamente, ser binária (variante A vs. variante B, como "concordância nominal padrão" vs. "concordância nominal não padrão"),[7] e as variáveis independentes não podem ser contínuas (como, p.ex., idade do falante: 20, 22, 27 anos), mas devem ser variáveis nominais (como, p.ex., faixas etárias: de 20 a 35, de 36 a 50 etc.). No GoldVarb X, não é possível modelar, adequadamente, variáveis contínuas, como a altura das vogais de acordo com suas medidas de F1; variáveis dependentes/resposta eneárias, com mais de dois níveis de variantes; o papel de efeitos aleatórios, próprios de cada amostra, como falantes individuais; não se pode visualizar

[7] Embora implementações anteriores do Varbrul permitissem a realização de análises eneárias com o programa MVarb.

os dados em diferentes tipos de gráficos; e não se pode comparar as medidas geradas com resultados de outros estudos, em outros programas, pois os pesos relativos são uma medida usada apenas por sociolinguistas.

No R, é possível realizar todo e qualquer tipo de análise estatística (modelos de regressão linear, análises de componentes principais, testes de qui-quadrado, anova, teste-t, árvores de inferências condicionais etc.). Isso porque, sendo uma linguagem de programação com código aberto, usuários em todo o globo constantemente produzem novas funções, para novos comandos. A instalação base do R vem com um conjunto extenso de funções e várias outras estão disponíveis por meio de pacotes (também chamados de bibliotecas) que contêm conjuntos de funções para aplicações específicas. Vários pacotes são utilizados neste manual de Estatística.

Um deles é o pacote swirl, um ambiente didático dentro da plataforma R, criado justamente para que se possa aprender R na prática, usando-o. Este curso de Estatística foi elaborado dentro desse ambiente, de modo que, caso o leitor decida segui-lo no formato swirl, é importante saber algumas de suas especificidades.

Após ter instalado o R e os pacotes necessários para este curso (ver Lição 0), o ambiente swirl pode ser aberto com os comandos abaixo:

```
library(swirl)
swirl()
```

No início de cada sessão, o swirl sempre pedirá para o usuário se identificar; se é o primeiro acesso – que o swirl reconhece pela digitação de um nome não usado anteriormente – o swirl fornecerá uma série de informações básicas: as reticências na tela indicam que o usuário deve pressionar ENTER para continuar; quando há opções numeradas, o usuário deve escolher uma delas (1, 2 etc.); quando o usuário quiser sair temporariamente do ambiente swirl, deve digitar `play()`; se quiser voltar ao menu principal, deve digitar `main()`; quando quiser pular uma questão, deve digitar `skip()`. Esta última opção pode ser usada apenas em perguntas para as quais o usuário deve digitar uma linha de comando para prosseguir (não pode ser usada em questões de múltipla escolha). Vale notar que o comando `skip()` dentro do swirl fornece a resposta

da pergunta, de modo que é bom tê-lo em mente quando tiver dúvidas sobre como prosseguir na lição. Por esse motivo, este manual não contém uma seção de respostas das lições e dos exercícios. O mais importante, claro, é tomar nota da resposta correta ou esperada, e buscar entendê-la, não apenas copiá-la.

Estrutura da obra

Este livro se encontra dividido em 15 capítulos, organizados em lições temáticas. A Lição 0 contém as instruções iniciais necessárias para o acompanhamento do curso (instalações necessárias e explicações básicas sobre a plataforma R, RStudio e o ambiente didático swirl). As três lições seguintes introduzem os fundamentos da linguagem de programação R (p.ex., como manipular vetores e dataframes, importação de dados ao R) e conceitos básicos de Estatística (o que são variáveis, variantes; tipos de variáveis; sua manipulação dentro do R). As Lições 4 a 7 apresentam os fundamentos da estatística descritiva: a feitura de tabelas; cálculo de medidas estatísticas como proporções, médias e desvio padrão; e elaboração de visualização gráfica de dados em gráficos de barras, de linhas, histogramas, boxplots e de dispersão. A Lição 8 apresenta os conceitos básicos de estatística inferencial: formulação de hipóteses (H0 e H1); distribuição normal; significância; erros do Tipo I e do Tipo II; nível-α e intervalo de confiança. As Lições 9 a 11 apresentam testes estatísticos univariados (que correlacionam uma variável dependente a uma variável independente): teste-t, teste de proporção, teste de qui-quadrado e teste de correlação de Pearson, em suas versões paramétricas e não-paramétricas. Por fim, as Lições 12 a 15 discutem a realização de análises de regressão linear e logística, em modelos de efeitos fixos e mistos.

Quadro 1: Estrutura dos capítulos e conteúdos do livro. (Ln): número da lição. VD: variável dependente. VI: variável independente.

<table>
<tr><td colspan="4">(L1) Fundamentos de R</td></tr>
<tr><td colspan="4">(L2) Estruturas básicas no R</td></tr>
<tr><td>(L3) Tipos de variáveis</td><td>VD nominal</td><td>VD ordinal</td><td>VD numérica</td></tr>
<tr><td rowspan="2">Estatística descritiva</td><td>(L4)
Tabela de frequências
Tabela de proporções</td><td></td><td>(L6)
Média, mediana, moda
Desvio padrão, variância</td></tr>
<tr><td>(L5)
Gráfico de barras
...</td><td>(L5)
Gráfico de linhas
...</td><td>(L7)
Gráfico de linhas
Boxplots
Histogramas
...</td></tr>
<tr><td colspan="4">(L8) Estatística inferencial</td></tr>
<tr><td rowspan="2">Análises univariadas
(VD ~ VI)</td><td rowspan="2">(L9)
Teste de proporção (VDnom ~ VInom)
Teste de qui-quadrado (VDnom ~ VInom)</td><td rowspan="2"></td><td>(L10)
Teste-t / Teste de Wilcoxon (VDnum ~ VIbinária)
(L11)
Teste de correlação (VDnum ~ VInum)</td></tr>
<tr><td></td></tr>
<tr><td>Análises multivariadas
(VD ~ VI + VI + VI...)</td><td>(L14-L15)
Regressão logística (VDbinária)</td><td></td><td>(L12-L13)
Regressão linear (VDnum)</td></tr>
</table>

O Quadro 1 mostra essa estrutura, que organiza as possibilidades de análises (estatística descritiva ou inferencial, análises uni e multivariadas) fundamentalmente de acordo com o tipo de variável dependente (VD) com que se está trabalhando: nominal, ordinal ou numérica, e secundariamente de acordo com o tipo de variável independente (VI): nominal/nominal binária ou numérica.

Não se trata, evidentemente, de um manual completo de estatística. O objetivo é o de proporcionar ao leitor as ferramentas mínimas necessárias para que possa aplicar o conteúdo das lições a seus próprios dados e para que possa prosseguir os estudos em Estatística em outras obras mais avançadas. Como livro-base, pode ser aplicado em cursos semestrais de introdução à estatística em nível de graduação ou de pós-graduação.

Cada lição se conclui com a indicação de novas leituras para aprofundamento dos conteúdos apresentados e com uma lista de exercícios para reforço e checagem do aprendizado. O texto é escrito de modo a simular uma conversa com o leitor, para que o assunto introduzido – algo que normalmente é objeto de receio dentre nossos alunos de Letras e de Linguística – seja acessível ao público pretendido. Também de modo consciente, evitam-se fórmulas matemáticas em excesso, ao mesmo tempo em que se apresentam referências complementares para o leitor interessado.

Todos os *scripts* estão disponíveis para *download* – você encontrará os links na seção "Links para *scripts* das lições", nas referências, junto a uma lista de indicações selecionadas para o aprofundamento dos estudos sobre Estatística e sobre a linguagem R. Há também, para cada lição, uma versão em videoaula disponível na playlist https://www.youtube.com/playlist?list=PLS6gbp0lYlXM5trsXb7RBUG2JR0zkc8Og em meu canal no YouTube.

Cabe destacar, por fim, que, embora o leitor possa ir direto a um capítulo específico, sua organização pressupõe a compreensão básica das lições que o precedem. A compreensão dessas lições também requer a aplicação do conteúdo na prática: dificilmente se chegará a um entendimento somente através da leitura deste livro; é necessário abrir a plataforma R, reproduzir os exemplos e aplicar as funções aos próprios dados. Boa programação!

Livia Oushiro

Lição 0: Instruções iniciais

Para seguir este manual em formato swirl, siga todos os passos abaixo. Para segui-lo por esta versão em *e-book*, siga os passos 1 a 5.

1. Acesse https://cran.r-project.org/, baixe o R correspondente a seu sistema operacional (Windows, Mac ou Linux) e instale-o.
2. Acesse https://www.rstudio.com/products/rstudio/download/, baixe a versão gratuita de RStudio correspondente a seu sistema operacional e instale-o. O RStudio é uma interface mais amigável do R; sua instalação é opcional, mas recomendada. Mesmo que já tenha os dois programas instalados em seu computador, certifique-se de que tem a versão mais recente de ambos. Se for o caso, baixe e instale a versão mais recente.
3. Preferencialmente, crie um ícone de atalho ao RStudio em um local de fácil acesso em seu computador (p.ex., na Área de Trabalho).
4. Abra o RStudio. Copie e cole a seguinte linha na janela Console, e em seguida pressione ENTER: (N.B.: Você precisa estar conectado à Internet quando fizer isso. Não estranhe se demorar alguns minutos para instalar os pacotes abaixo.)

```
install.packages(c("swirl", "png", "tidyverse", "GGally", "languageR",
"base64enc", "aod", "car", "effects", "gvlma", "multcomp", "rgl", "rms
", "lme4", "lmerTest", "MuMIn", "Hmisc", "RColorBrewer", "gridExtra"))
```

5. No topo da janela do RStudio, clique sobre Tools > Global options > Code > Saving. No campo "Default text encoding", escolha a opção UTF-8. Clique em OK para sair dessa janela.
6. Baixe os arquivos Introducao_a_Estatistica_para_Linguistas.swc e Introducao_a_Estatistica_para_Linguistas_Exercicios.swc de https://doi.org/10.5281/zenodo.822069 a um lugar acessível de seu computador (p.ex., na Área de Trabalho).

7. Volte ao RStudio. Copie e cole a seguinte linha na janela Console, e em seguida pressione ENTER:

```
library(swirl)
```

8. No RStudio, copie e cole a seguinte linha na janela Console, e então pressione ENTER:

```
install_course()
```

9. Uma janela de navegação se abrirá em seu computador. Procure o arquivo Introducao_a_Estatistica_para_Linguistas.swc, baixado em seu computador, e clique sobre ele. A mensagem abaixo deve aparecer no Console do RStudio.

```
| Course installed successfully!
```

10. Repita o passo 9 para instalar o arquivo Introducao_a_Estatistica_para_Linguistas_Exercicios.swc.
11. No RStudio, copie e cole a seguinte linha na janela Console, e então pressione ENTER:

```
install_course("R Programming")
```

12. (Passo opcional) Se você preferir o português ao inglês, copie e cole a seguinte linha no Console:

```
select_language("portuguese")
```

13. Digite no Console:

```
swirl()
```

14. Siga as instruções que aparecerem na tela!
15. Todas as vezes que você for acessar o swirl nas próximas sessões, será necessário digitar no Console:

```
library(swirl)
swirl()
```

16. O swirl pedirá para que você se identifique. Use sempre o mesmo ID.

Dentro do R, as linhas de comando podem ser digitadas em arquivos de *script*, cuja extensão é .R, ou diretamente no Console, quando a plataforma apresenta o símbolo > – que indica que está pronta para receber novos comandos. Os comandos são rodados com CTRL + ENTER quando digitados em *script* e com ENTER quando digitados no Console. Nesta obra, as linhas precedidas por ## são o resultado gerado pelo programa e, portanto, não se referem a linhas de comando. Se quiser saber mais e fazer outros cursos de programação no swirl, veja a página https://swirlstats.com/students.html. Em especial, recomendo os cursos "Regression models", "Statistical inference" e "Exploratory data analysis".

Lição 1: Fundamentos

Bem-vinde ao curso de Introdução à Estatística para Linguistas!

Primeiro, vamos explorar alguns fundamentos da linguagem. O R pode funcionar como uma calculadora. Digite `4 + 9` no *script* desta lição[8] e pressione CTRL + ENTER para ver o resultado.

```
4 + 9
```

```
## [1] 13
```

O R apresenta o resultado 13. Mas usar o R para fazer esse tipo de cálculo, em vez de usar uma simples calculadora, pode parecer desnecessário. Na maior parte das vezes, usamos o R para automatizar algum processo ou evitar repetição. Podemos querer usar o resultado de `4 + 9` em outros cálculos. Para isso, em vez de redigitar 4 + 9 várias vezes, podemos guardar o resultado em uma variável.

Para atribuir um valor a uma variável, use o operador de atribuição `<-`, que é o símbolo de menor seguido do símbolo de menos. O operador de atribuição representa iconicamente uma flecha, que atribui o valor à direita a uma variável à esquerda. Para atribuir o resultado de 4 + 9 a uma nova variável chamada `x`, digite `x <- 4 + 9` e rode essa linha de comando com CTRL + ENTER.

```
x <- 4 + 9
```

Você deve ter percebido que o R não exibiu o resultado 13 desta vez. Quando se usa o operador de atribuição, o R assume que você não precisa do resultado imediatamente, mas sim para algum uso futuro. Para ver o conteúdo da variável `x`, digite `x` e pressione CTRL + ENTER.

```
x
```

```
## [1] 13
```

[8] Os *scripts* de todas as lições podem ser acessados pelos links disponibilizados na seção "Links para *scripts* das lições".

Como uma variável, `x` agora pode entrar em outros cálculos. Qual seria o resultado de x + 10?

- 21
- 23
- 25

Vamos testar o resultado da expressão (pra qual já sabemos o resultado). Digite `x + 10` e pressione CTRL + ENTER.

```
x + 10
```

```
## [1] 23
```

O uso de atribuição de variáveis é recursivo. Podemos atribuir o valor de x + 10 a uma nova variável `y`. Tente fazer isso agora.

```
y <- x + 10
```

Agora visualize o valor de `y`.

```
y
```

```
## [1] 23
```

Acho que deu pra pegar o jeitão da coisa, né? A variável `x` terá o valor de 13 até que se atribua novo valor a ela ou até que se encerre a sessão do R sem salvar o espaço de trabalho.

Procure no RStudio a janela com as abas Environment e History. Em Environment, o R lista os objetos criados e seus respectivos valores. Na aba History, o R mostra o histórico de comandos.

Para nós, linguistas, o interesse em usar o R é em sua capacidade de lidar bem não só com números, mas também com caracteres e elementos textuais. Criamos uma variável com uma sequência de caracteres usando as aspas. Digite `z <- "gato"` para criar a variável `z`.

```
z <- "gato"
```

Agora visualize o conteúdo de `z`.

```
z
```

```
## [1] "gato"
```

Note as aspas em volta do termo `gato`, que indica que a variável contém uma sequência de caracteres. O que você acha que vai acontecer se tentarmos somar a variável `z` com o número 1?

- O R retornará o plural de `gato`;
- O R retornará "gato" + 1;
- O R retornará um erro seguido de uma explicação;
- O resultado é imprevisível;
- Aparecerá uma mensagem de que seu computador se autodestruirá em 5 segundos

Toda vez que o R não consegue executar uma linha de comando, ele dá uma mensagem de erro e indica o que está errado. Ao tentar computar `z + 1`, ele indicará que um dos argumentos não é numérico e que, portanto, não consegue fazer a soma. Ao ocorrer uma mensagem de erro, não se desespere! Basta ler a mensagem e tentar descobrir onde estava o erro. Se quiser testar, digite `z + 1` no Console para ver o resultado.

Aqui vale a pena mencionar um recurso que o R e diversas outras linguagens de programação têm: pode-se voltar a uma linha de comando anterior pressionando a flecha para cima ↑, estando com o cursor no Console. Faça isso até voltar ao comando `x + 10`, que executamos acima.

```
x + 10
## [1] 23
```

Até agora, inserimos apenas um elemento nas variáveis `x`, `y` e `z`. Para juntar mais de um elemento, usamos a função `c()`, em que `c` significa "combinar". Suponhamos então que você queira juntar os números 1, 5 e 7. Digite `c(1, 5, 7)`. Note que não há espaço entre o nome da função e a abertura de parênteses, e que a letra `c` é minúscula.

```
c(1, 5, 7)
## [1] 1 5 7
```

Também é possível juntar outros objetos previamente criados. Vamos juntar as variáveis `x`, `y`, criadas anteriormente, e o número 230. Além disso, vamos guardar esses elementos dentro de um objeto, que chamaremos de `aleatorio`. Digite `aleatorio <- c(x, y, 230)`.

```
aleatorio <- c(x, y, 230)
```

Agora veja o conteúdo do objeto `aleatorio`.

```
aleatorio
```

```
## [1]  13  23 230
```

O objeto `aleatorio`, assim como `x`, `y` e `z`, é um *vetor*. Vetores são sequências de dados de um mesmo tipo (numérico, de caracteres, ou lógico). Este último, que ainda não vimos, conteria apenas valores TRUE (= "verdadeiro") ou FALSE (= "falso"). Mas fiquemos, por ora, apenas nos vetores numéricos ou de caracteres. Vetores podem ter qualquer número de elementos – e muitas vezes vamos trabalhar com vetores de centenas ou milhares de dados.

Sempre que tiver dúvidas sobre uma função específica, você pode chamar o Help do R por meio do comando `?` seguido do nome da função, sem os parênteses. Por exemplo, digite `?c` para visualizar mais informações sobre essa função na aba Help.

```
?c
```

Veja na aba Help da janela Files, Plots, Packages... que o R abriu a ajuda sobre essa função. Fique à vontade para explorá-la.

Outro operador útil no R é `:`, que representa um intervalo. Crie um vetor chamado `numeros` que contém os números de 20 a 39. Digite `numeros <- c(20:39)`.

```
numeros <- c(20:39)
```

Veja agora o conteúdo do vetor recém-criado.

```
numeros
```

```
##  [1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
```

Esse é um jeito muito mais fácil de criar um vetor numérico do que digitar c(20, 21, 22, 23...).

Você notou que os vetores que criamos não contêm caracteres especiais, como diacríticos (p.ex. ´^), espaços e outros como - & $...? Ao nomear objetos no R, evite esses caracteres!

Vejamos agora outras funções úteis no R. Algo importante a se fazer a cada nova sessão do R é gerenciar o espaço de trabalho. Para isso, usamos as funções `getwd()` ou `setwd()`. `wd` nessas funções significa "working directory", ou diretório de trabalho, que é a pasta em seu sistema que o R está usando como referência para interagir com arquivos fora do R (p.ex., seu arquivo de dados linguísticos!).

Vamos descobrir qual é o diretório de trabalho atual. Digite `getwd()`.

```
getwd()
## [1] "/Users/livia/Dropbox/_R/RMarkdown/IEL_203_EbookABRALIN"
```

N.B.: O resultado em seu computador será diferente do que aparece aqui!

Observe o formato com que o R retorna o atual diretório de trabalho. Trata-se do caminho completo para a pasta, entre aspas, e com barras para a direita `/`. Para mudar o diretório de trabalho, usamos a função `setwd()`. Dentro dos parênteses dessa função, digitamos o caminho completo do novo diretório no mesmo formato – entre aspas e com barras para a direita. P.ex.: `setwd("C:/Users/IntroaEstatistica")`.

Mas há um jeito relativamente mais fácil de fazer isso no RStudio. Procure a janela que contém as abas Files, Plots, Packages... e clique sobre a aba Files. Dentro dessa aba, no canto superior direito, clique sobre as reticências ..., como se vê na Figura 1.1.

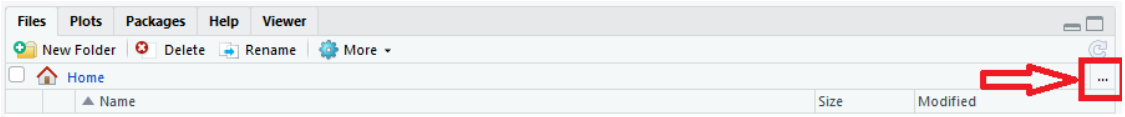

Figura 1.1: Localização das reticências na aba Files. Fonte: própria.

Abrirá uma janela que permite navegar até a pasta desejada. Selecione a pasta que deseja transformar em novo diretório de trabalho (Figura 1.2). Recomendo que você crie uma nova pasta em seu computador só para isso!

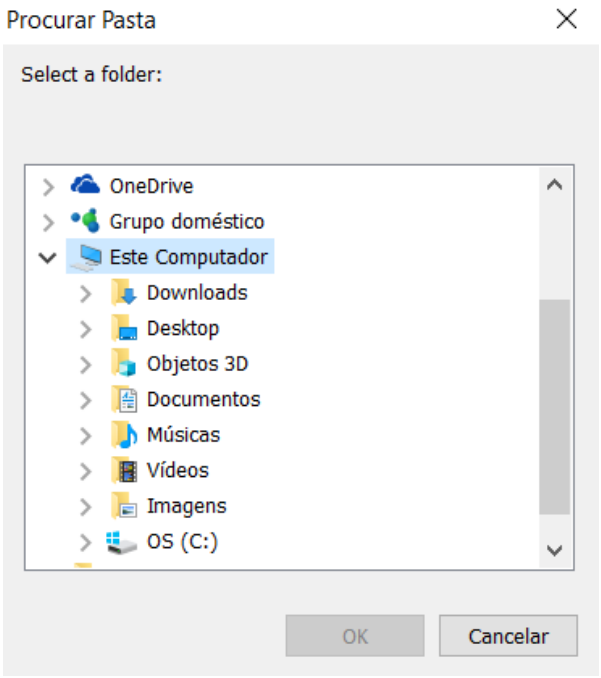

Figura 1.2: Janela Procurar Pasta. Fonte: própria.

Você verá seu conteúdo na aba Files. Em seguida, clique sobre "More" e em "Set As Working Directory" (Figura 1.3). Faça isso agora.

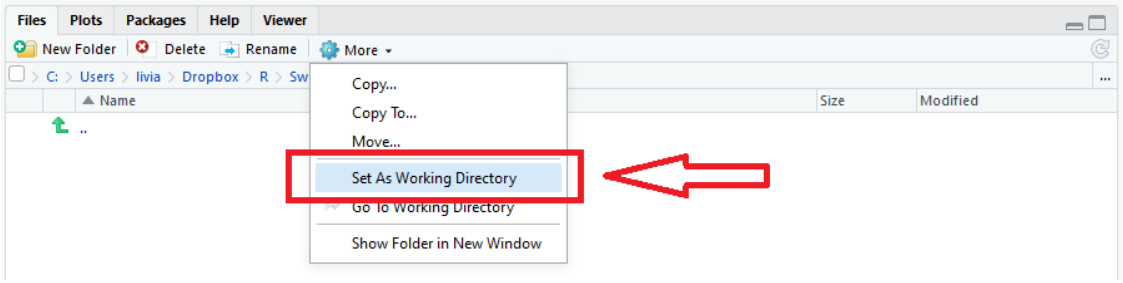

Figura 1.3: Definir diretório de trabalho pela janela. Fonte: própria.

```
setwd("~/Dropbox/_R/swirl/Introducao_a_Estatistica_para_Linguistas/data")
```

N.B.: O resultado em seu computador provavelmente será diferente do que aparece aqui!

O R executou a linha de comando no Console e estabeleceu um novo diretório de trabalho. Nesse caso, utilizamos a interface do RStudio para executar um comando sem ter que digitar uma linha. Embora isso seja mais fácil quando executado uma única vez, essa ação pode ser mais trabalhosa se se tornar repetitiva. Por exemplo, se quisermos

estabelecer o mesmo diretório de trabalho a cada vez que se inicia uma sessão no R, aí já não vale mais a pena estabelecer o diretório de trabalho desse modo.

Uma boa ideia é selecionar e copiar (CTRL/Command + C) essa linha de comando do Console e colá-la no seu *script*. Faça isso agora. Note que, ao fazer uma alteração no arquivo do *script*, aparece um asterisco vermelho ao lado do nome do arquivo. Isso indica que há modificações não salvas. Salve com CTRL/Command + S. Aliás, é uma boa ideia fazer isso periodicamente, para não perder seu trabalho!

Os arquivos de *scripts* funcionam como qualquer outro arquivo comum. Pode-se abri-los, fechá-los, salvá-los com outro nome com Salvar como... Se em seus *scripts* você for usar sequências de caracteres em português, uma dica é usar a opção File > Save with encoding e escolher a opção UTF-8, que lê corretamente caracteres especiais como diacríticos (´ ` ^). (Mas lembre-se que é bom evitá-los ao criar ou nomear objetos no R!)

Por fim, vamos ver mais uma função que será usada extensivamente em suas análises – `read_csv()`. Na maior parte dos casos para as análises estatísticas, queremos analisar uma planilha de dados preparada previamente em um programa como o Excel ou o Calc. A função `read_csv()` faz parte dos pacotes readr e tidyverse – este último, um conjunto de pacotes para manipulação de dados. Os pacotes (também chamados de "bibliotecas") são conjuntos de funções criadas por usuários do R e disponibilizadas aos demais usuários, e que não fazem parte da instalação base do R. Sempre que quiser instalar um novo pacote, você deve usar a função `install.packages()`, um comando que você usou para instalar os pacotes swirl e vários outros para este curso!

Para usar funções que não fazem parte da instalação base do R, é necessário acessar, a cada nova sessão, o pacote/biblioteca por meio da função `library()`. Aplique-a ao pacote tidyverse.

```
library(tidyverse)
```

A planilha DadosRT tem o formato .csv (= comma separated values). Para ler esse tipo de arquivo no R, podemos usar a função `read_csv()`, do pacote tidyverse. Rode a linha de comando a seguir com a função `read_csv()` – já pronta! –, para que o R leia os dados da planilha e os insira dentro de um objeto chamado `dados`. Basta colocar o cursor

em qualquer ponto da linha de comando e pressionar CTRL + ENTER. É possível que demore um tempinho para rodá-la; enquanto o R executa esse comando, note que aparece um círculo vermelho no topo direito da janela Console (STOP), que indica que o R está executando determinado comando e que você deve esperar antes de dar novos comandos.

```
dados <- read_csv("DadosRT.csv",
                  col_types =
                    cols(.default = col_factor(),
                         cont.precedente = col_character(),
                         ocorrencia = col_character(),
                         cont.seguinte = col_character(),
                         IDADE = col_integer(),
                         INDICE.SOCIO = col_double(),
                         FREQUENCIA = col_double()
                         )
                  )
```

Essa linha de comando é mais complexa do que as anteriores, de modo que vale a pena analisar sua estrutura. Primeiro, note que, mesmo que o comando ocupe mais de uma linha, trata-se de uma única linha de comando. Ao ler `dados <- read_csv("DadosRT.csv",`, o R reconhece que esta é uma linha de comando incompleta, e isso é sinalizado com o símbolo "+" no Console. Ela está incompleta porque, nessa primeira linha, um parêntese foi aberto, mas não foi fechado. Essa é uma regra sagrada na programação: tudo que abre tem que fechar! Há, ademais, uma vírgula, que indica que haverá mais argumentos na função.

Usamos a função `read_csv()` com dois argumentos, file e col_types. Todas as funções têm parênteses e, dentro deles, podem ser inseridos zero, um ou mais argumentos – como nas funções `getwd()`, `library()` e `c()`, respectivamente. Acesse a Ajuda da função `read_csv()`.

```
?read_csv
```

O primeiro argumento, file, foi preenchido com "DadosRT.csv". Porque colocamos esse argumento em sua posição *default*, não foi necessário digitar file = "DadosRT.csv" para especificar qual argumento é esse. O segundo argumento usado foi col_types. Esse argumento tomou ele mesmo outra função, chamada `cols()`, para

definir o tipo das colunas nesse dataframe. Note aí a recursividade! Na função `cols()`, especificamos que o tipo *default* de cada coluna é `factor`, exceto pelas colunas cont.precedente, ocorrencia e cont.seguinte, que devem ser lidas como "character", e as colunas IDADE, INDICE.SOCIO e FREQUENCIA, que devem ser lidas como "integer" e "double" (tipos de número). Na Lição 3, vamos ver essa tipologia de dados com mais detalhes.

Note, além disso, que `col_factor()`, `col_character()`, `col_integer()` e `col_double()` são também funções – algo que sabemos pelos parênteses. É possível criar outras especificações dentro dessas funções como, por exemplo, mudar a ordem dos níveis de uma variável. Também veremos isso na Lição 3. Por ora, interessa chamar a atenção para essa estrutura mais geral de funções, que sempre têm parênteses, e cujos argumentos, dentro dos parênteses, são separados por vírgulas. Os diferentes argumentos das funções `read_csv()` e `cols()` foram colocados em linhas distintas no *script* justamente para mais bem visualizar a estrutura da função `read_csv()` e as demais funções nela encaixadas.

Por fim, você também deve ter notado, na Ajuda da função `read_csv()`, que existem vários outros argumentos nas funções da família read_delim. Alguns argumentos podem ter um valor *default*; quando o usuário não especifica nada para determinado argumento, o R assume que seu valor é o *default*, pré-especificado pelo autor ou autora da função. Por exemplo, para `read_csv()`, o argumento col_names é pré-especificado como TRUE, de modo que, se nada for especificado quanto aos nomes das colunas, o R assume que a primeira linha contém o nome das colunas do dataframe.

Vamos inspecionar, então, o conteúdo do objeto recém-criado `dados`.

```
dados
```

```
## # A tibble: 9,226 × 20
##    VD         PARTICIPANTE SEXO.GENERO IDADE FAIXA.ETARIA ESCOLARID
ADE
##    <fct>      <fct>        <fct>       <int> <fct>        <fct>
##  1 retroflexo IvanaB       feminino       30 1a           fundament
al
##  2 retroflexo IvanaB       feminino       30 1a           fundament
al
##  3 retroflexo IvanaB       feminino       30 1a           fundament
```

```
al
##  4 tepe       IvanaB       feminino       30 1a           fundament
al
##  5 retroflexo IvanaB       feminino       30 1a           fundament
al
##  6 retroflexo IvanaB       feminino       30 1a           fundament
al
##  7 retroflexo IvanaB       feminino       30 1a           fundament
al
##  8 retroflexo IvanaB       feminino       30 1a           fundament
al
##  9 tepe       IvanaB       feminino       30 1a           fundament
al
## 10 retroflexo IvanaB       feminino       30 1a           fundament
al
## # … with 9,216 more rows, and 14 more variables: REGIAO <fct>,
## #   INDICE.SOCIO <dbl>, ORIGEM.PAIS <fct>, CONT.FON.PREC <fct>,
## #   CONT.FON.SEG <fct>, TONICIDADE <fct>, POSICAO.R <fct>,
## #   CLASSE.MORFOLOGICA <fct>, FREQUENCIA <dbl>, ESTILO <fct>,
## #   ITEM.LEXICAL <fct>, cont.precedente <chr>, ocorrencia <chr>,
## #   cont.seguinte <chr>
```

Dentro do universo tidyverse, o R retorna um tibble – um resumo de um dataframe. O arquivo DadosRT, agora disponível em `dados`, é uma planilha com dados da pronúncia variável de /r/ em coda, em palavras como "porta" e "mulher", na fala de paulistanos. Encontraremos esse arquivo novamente em lições futuras.

Nesta lição, você aprendeu alguns comandos fundamentais no R, que serão bastante utilizados nas próximas lições – como criar vetores, como acessar a Ajuda, como usar as funções `c()`, `getwd()`, `setwd()`, `library()`, `read_csv()`. Além disso, vimos as quatro janelas que compõem o ambiente do RStudio.

Embora possa parecer complicado digitar linhas de comando em vez de clicar em botões, normalmente só se tem que digitar os comandos uma ou poucas vezes. Na maior parte do tempo, trabalha-se com *scripts*, que é um conjunto de linhas de comando que são guardadas em arquivos de extensão .R.

Para saber mais

Recomendo que você faça as lições 1 ("Basic Building Blocks") e 2 ("Workspace and Files") do curso R Programming, do swirl.

Exercícios

1. Crie um vetor chamado `numeros1` que contém os números 1, 3 e 5.
2. Crie um vetor chamado `numeros2` que contém os números 11, 13 e 15.
3. Crie um vetor chamado `classe_morfologica` que contém os termos "substantivo", "adjetivo", "adverbio", "pronome" e "verbo".
4. Guarde o caminho do atual diretório de trabalho em uma variável chamada `dir_antigo`.
5. Defina como diretório de trabalho a pasta em que se localiza a planilha DadosRT.csv.
6. Cheque se o novo diretório de trabalho está correto.
7. Instale o pacote lme4. Ele será usado em lições futuras para criar modelos de efeitos mistos.
8. Carregue o pacote tidyverse.
9. Carregue a planilha DadosRT.csv em um objeto chamado `dadosR`. Use a função `read_csv()` para isso.
10. Inspecione os elementos do vetor `numeros1`.
11. Se somarmos os vetores `numeros1` e `numeros2`, qual será o resultado?
    a. Um vetor de um elemento
    b. Um vetor de três elementos
    c. Um erro
12. Some os vetores `numeros1` e `numeros2` e cheque a resposta.
13. Se multiplicarmos o vetor `numeros1` por 2, qual será o resultado?
    a. Um vetor de um elemento
    b. Um vetor de três elementos
    c. Um erro
14. Multiplique o vetor `numeros1` por 2. O sinal de multiplicação no R é o asterisco *.
15. Inspecione os valores do vetor `classe_morfologica`.
16. Se somarmos o vetor `classe_morfologica` com 2, qual será o resultado?

a. Um vetor de um elemento
b. Um vetor de três elementos
c. Um erro

17. A função `length()` informa quantos elementos há dentro de um objeto do R. Veja a descrição dessa função por meio da Ajuda.
18. Escolha a linha de comando correta para aplicar a função `length()` no vetor `classe_morfologica`.
    a. `length(classe_morfologica)`
    b. `length("classe_morfologica")`
    c. `classe_morfologica <- length()`
19. Aplique a função `length()` no vetor `classe_morfologica`.
20. Aplique a função `length()` no vetor `numeros2` e atribua o resultado a uma variável chamada `N.numeros2`.
21. Qual é o conteúdo do vetor `N.numeros2`?
    a. Um vetor de um elemento
    b. Um vetor de três elementos
    c. Um erro
22. Qual é o valor contido no vetor `N.numeros2`?
    a. 1
    b. 3
    c. 5
23. Defina o antigo diretório de trabalho novamente como o diretório de trabalho atual. Use a função `setwd()` e a variável `dir_antigo` para isso.

Lição 2: Manipulação de Vetores e Dataframes

Nesta lição, vamos ver dois tipos de estruturas de dados que serão usados mais frequentemente nas próximas tarefas – vetores e dataframes. Além disso, vamos aprender a usar algumas funções úteis para inspecionar esses tipos de objeto. Para outros tipos de estruturas de dados, você pode consultar a excelente página de Hadley Wickham: http://adv-r.had.co.nz/Data-structures.html.

Vetores são a estrutura de dados mais básica do R. São estruturas unidimensionais de dados do mesmo tipo (numéricos, caracteres etc.). Na Lição 1 – Fundamentos, criamos alguns vetores: `x`, `y`, `z`, `aleatorio`, `numeros`. Também carregamos o dataframe `dados`. Rode as linhas de comando a seguir para deixá-los novamente disponíveis. Defina como diretório de trabalho aquele que contém a planilha DadosRT.csv.

```
# Recriar vetores da Lição 1

x <- 4 + 9
y <- x + 10
z <- "gato"
aleatorio <- c(x, y, 230)
numeros <- c(20:39)

# Definir diretório de trabalho

#setwd()

# Carregar dados

dados <- read_csv("DadosRT.csv",
                  col_types = cols(.default = col_factor(),
                                   cont.precedente = col_character(),
                                   ocorrencia = col_character(),
                                   cont.seguinte = col_character(),
                                   IDADE = col_integer(),
                                   INDICE.SOCIO = col_double(),
                                   FREQUENCIA = col_double()
                                   )
                  )
```

Para esta lição, vamos precisar mais adiante das funções do pacote tidyverse. Carregue-o para deixá-lo disponível nesta sessão.

```
library(tidyverse)
```

Para acessar elementos específicos dentro de um vetor, usamos o operador de indexação `[]` com um número dentro dos colchetes. Veja o que dá a linha de comando `aleatorio[3]`.

```
aleatorio[3]
## [1] 230
```

O R dá como resultado o terceiro elemento do vetor `aleatorio`. Peça ao R agora que mostre qual é o 15º elemento do vetor `numeros`.

```
numeros[15]
## [1] 34
```

Você se lembra qual era o valor de `x`? Olhe a aba Environment se necessário. Qual será o resultado de `numeros[x]`?

- O resultado é imprevisível
- Vai ocorrer um erro, porque `x` não é um número
- O R mostrará o 13º elemento do vetor `numeros`

Vamos checar o resultado de `numeros[x]`. Digite esse comando no *script*.

```
numeros[x]
## [1] 32
```

E se quisermos pedir ao R que mostre o 1º e o 3º elementos do vetor `aleatorio`? Uma função vista na Lição 1 vai ser útil aqui. Qual é ela?

- `c()`
- `getwd()`
- `library()`
- `read_csv()`
- `setwd()`

Vamos testar! Peça ao R que mostre o 1º e 3º elementos do vetor `aleatorio` com `aleatorio[c(1, 3)]`.

```
aleatorio[c(1, 3)]
```

```
## [1]  13 230
```

Também é possível exibir todos os elementos de um vetor *exceto* um ou alguns, com o operador de subtração `-`. Teste `numeros[-2]`.

```
numeros[-2]
```

```
##  [1] 20 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
```

O R retorna todos os elementos do vetor `numeros`, exceto o segundo elemento.

Na lição anterior, também vimos outro tipo de estrutura de dados, um dataframe. Diferentemente de um vetor, o dataframe é um conjunto bidimensional de dados, com linhas e colunas. Outro modo de entender o dataframe é como um conjunto de vetores de mesma extensão (mesmo número de colunas ou mesmo número de linhas).

Na Lição 1, havíamos carregado uma planilha dentro de um objeto chamado `dados`. Ele está disponível novamente para esta sessão do R. Digite `dados` para visualizar o tibble desse dataframe.

```
dados
```

```
## # A tibble: 9,226 × 20
##    VD         PARTICIPANTE SEXO.GENERO IDADE FAIXA.ETARIA ESCOLARID
ADE
##    <fct>      <fct>        <fct>       <int> <fct>        <fct>
##  1 retroflexo IvanaB       feminino       30 1a           fundament
al
##  2 retroflexo IvanaB       feminino       30 1a           fundament
al
##  3 retroflexo IvanaB       feminino       30 1a           fundament
al
##  4 tepe       IvanaB       feminino       30 1a           fundament
al
##  5 retroflexo IvanaB       feminino       30 1a           fundament
al
##  6 retroflexo IvanaB       feminino       30 1a           fundament
al
##  7 retroflexo IvanaB       feminino       30 1a           fundament
al
##  8 retroflexo IvanaB       feminino       30 1a           fundament
al
##  9 tepe       IvanaB       feminino       30 1a           fundament
al
## 10 retroflexo IvanaB       feminino       30 1a           fundament
al
## # … with 9,216 more rows, and 14 more variables: REGIAO <fct>,
## #   INDICE.SOCIO <dbl>, ORIGEM.PAIS <fct>, CONT.FON.PREC <fct>,
```

```
## #   CONT.FON.SEG <fct>, TONICIDADE <fct>, POSICAO.R <fct>,
## #   CLASSE.MORFOLOGICA <fct>, FREQUENCIA <dbl>, ESTILO <fct>,
## #   ITEM.LEXICAL <fct>, cont.precedente <chr>, ocorrencia <chr>,
## #   cont.seguinte <chr>
```

Veja que o R exibiu apenas as primeiras linhas de `dados` e omitiu as restantes. Se quisermos visualizar o conjunto completo de dados, é melhor fazê-lo em outra aba. Aplique a função `View()` a `dados` e veja o resultado.

```
View(dados)
```

N.B.: Resultado aqui omitido.

O R abre uma nova aba em Source e permite visualizar a planilha como no Excel ou no Calc. Bastante útil, não? O R também permite saber certas propriedades deste dataframe. O número de colunas pode ser visto com a função `ncol()`, que toma o dataframe como argumento. Digite `ncol(dados)`.

```
ncol(dados)
```

```
## [1] 20
```

Veja agora o número de linhas aplicando a função `nrow()` a `dados`.

```
nrow(dados)
```

```
## [1] 9226
```

A função `str()` (=structure) fornece uma visão global sobre o conjunto de dados. Aplique-a a `dados` para ver o resultado.

```
str(dados)
```

```
## spec_tbl_df [9,226 × 20] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ VD                : Factor w/ 2 levels "retroflexo","tepe": 1 1
1 2 1 1 1 1 2 1 ...
##  $ PARTICIPANTE      : Factor w/ 118 levels "IvanaB","HeloisaS",..:
1 1 1 1 1 1 1 1 1 1 ...
##  $ SEXO.GENERO       : Factor w/ 2 levels "feminino","masculino": 1
1 1 1 1 1 1 1 1 1 ...
##  $ IDADE             : int [1:9226] 30 30 30 30 30 30 30 30 30 30 .
..
##  $ FAIXA.ETARIA      : Factor w/ 3 levels "1a","3a","2a": 1 1 1 1 1
1 1 1 1 1 ...
##  $ ESCOLARIDADE      : Factor w/ 3 levels "fundamental",..: 1 1 1 1
1 1 1 1 1 1 ...
##  $ REGIAO            : Factor w/ 2 levels "periferica","central": 1
1 1 1 1 1 1 1 1 1 ...
##  $ INDICE.SOCIO      : num [1:9226] 2 2 2 2 2 2 2 2 2 2 ...
##  $ ORIGEM.PAIS       : Factor w/ 5 levels "mista","SPcapital",..: 1
1 1 1 1 1 1 1 1 1 ...
```

```
##  $ CONT.FON.PREC     : Factor w/ 7 levels “a”,”o”,”e”,”0”,..: 1 2 3
3 1 1 4 1 4 5 ...
##  $ CONT.FON.SEG      : Factor w/ 19 levels “ts”,”n”,”g”,”v”,..: 1 2
3 4 5 6 7 7 5 8 ...
##  $ TONICIDADE        : Factor w/ 2 levels “tonica”,”atona”: 1 2 2 2
1 1 1 1 2 1 ...
##  $ POSICAO.R         : Factor w/ 2 levels “medial”,”final”: 1 1 1 1
2 2 1 1 1 1 ...
##  $ CLASSE.MORFOLOGICA: Factor w/ 6 levels “substantivo”,..: 1 1 1 1
1 1 1 1 2 1 ...
##  $ FREQUENCIA        : num [1:9226] 1.34 0.16 0.22 0.44 1.94 1.94 0
.35 0.03 5.98 0.16 ...
##  $ ESTILO            : Factor w/ 4 levels “conversacao”,..: 1 1 1 1
1 1 1 1 1 1 ...
##  $ ITEM.LEXICAL      : Factor w/ 1151 levels “parte”,”jornal”,..: 1
2 3 4 5 5 6 7 8 9 ...
##  $ cont.precedente   : chr [1:9226] “do CEU é daquela” “eu não gost
o de” “não (es)to(u) entendendo a” “o que sei... num” ...
##  $ ocorrencia        : chr [1:9226] “parte <R>“ “jornal <R>“ “pergu
nta <R>“ “serviço <T>“ ...
##  $ cont.seguinte     : chr [1:9226] “que as perua(s) ia” “D1: mas d
igo assim” “D1: o que que/” “a não ser <A>“ ...
##  - attr(*, “spec”)=
##   .. cols(
##   ..   .default = col_factor(),
##   ..   VD = col_factor(levels = NULL, ordered = FALSE, include_na =
FALSE),
##   ..   PARTICIPANTE = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   SEXO.GENERO = col_factor(levels = NULL, ordered = FALSE, inc
lude_na = FALSE),
##   ..   IDADE = col_integer(),
##   ..   FAIXA.ETARIA = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   ESCOLARIDADE = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   REGIAO = col_factor(levels = NULL, ordered = FALSE, include_
na = FALSE),
##   ..   INDICE.SOCIO = col_double(),
##   ..   ORIGEM.PAIS = col_factor(levels = NULL, ordered = FALSE, inc
lude_na = FALSE),
##   ..   CONT.FON.PREC = col_factor(levels = NULL, ordered = FALSE, i
nclude_na = FALSE),
##   ..   CONT.FON.SEG = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   TONICIDADE = col_factor(levels = NULL, ordered = FALSE, incl
ude_na = FALSE),
##   ..   POSICAO.R = col_factor(levels = NULL, ordered = FALSE, inclu
de_na = FALSE),
##   ..   CLASSE.MORFOLOGICA = col_factor(levels = NULL, ordered = FAL
SE, include_na = FALSE),
##   ..   FREQUENCIA = col_double(),
##   ..   ESTILO = col_factor(levels = NULL, ordered = FALSE, include_
na = FALSE),
```

```
##   ..   ITEM.LEXICAL = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   cont.precedente = col_character(),
##   ..   ocorrencia = col_character(),
##   ..   cont.seguinte = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

O resultado de `str()` informa que `dados` é um dataframe com 9.226 linhas (observações) e 20 colunas (variáveis). Em seguida, indica, para cada variável, seu nome (que segue o símbolo `$`), seu tipo (num, Factor with N levels etc.) e seus primeiros dados. Veja que a função `str()` faz as vezes de `ncol()`, `nrow()` e do tibble, de modo que vamos usá-la frequentemente nas próximas lições. Sempre que carregar um dataframe com `read_csv()`, vale a pena checar se o dataframe foi carregado corretamente com `str()`.

Também vale conhecer a função `summary()`. Aplique-a a `dados` para ver o resultado.

```
summary(dados)
```

```
##           VD         PARTICIPANTE     SEXO.GENERO       IDADE
##  retroflexo:2611   VeraD   : 108   feminino :4565   Min.   :19.00
##  tepe      :6615   RodrigoC: 106   masculino:4661   1st Qu.:31.00
##                    RogerioA: 106                    Median :47.00
##                    MarcoP  : 105                    Mean   :46.82
##                    LucasS  : 105                    3rd Qu.:61.00
##                    RenataC : 105                    Max.   :83.00
##                    (Other) :8591
##  FAIXA.ETARIA      ESCOLARIDADE          REGIAO      INDICE.SOCIO
##  1a:3051      fundamental:1098   periferica:4881   Min.   :1.600
##  3a:2919      superior   :4729   central   :4345   1st Qu.:2.700
##  2a:3256      medio      :3399                     Median :3.100
##                                                    Mean   :3.075
##                                                    3rd Qu.:3.500
##                                                    Max.   :4.500
##
##       ORIGEM.PAIS   CONT.FON.PREC  CONT.FON.SEG   TONICIDADE
##  mista      :3201   a:1983        t      :1513   tonica:4611
##  SPcapital  :1992   o:1709        k      :1382   atona :4615
##  interior   :2562   e:2088        m      :1065
##  nordeste   : 611   0: 696        d      : 748
##  estrangeira: 860   3: 787        #      : 725
##                     u:1369        g      : 653
##                     i: 594        (Other):3140
##   POSICAO.R      CLASSE.MORFOLOGICA   FREQUENCIA
##  medial:7382   substantivo:4826     Min.   :0.010
##  final :1844   conj.prep  : 969     1st Qu.:0.130
##                adverbio   :  37     Median :0.620
##                verbo      :1181     Mean   :1.034
```

```
##                adjetivo   :1530     3rd Qu.:0.930
##                morf.inf   : 683     Max.   :5.980
##
##          ESTILO      ITEM.LEXICAL  cont.precedente
##  conversacao:5900   porque : 552   Length:9226
##  palavras   :1985   por    : 300   Class :character
##  jornal     : 592   perto  : 240   Mode  :character
##  depoimento : 749   erguer : 200
##                     irmã   : 193
##                     lugar  : 179
##                     (Other):7562
##   ocorrencia        cont.seguinte
##  Length:9226        Length:9226
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

`summary()` também fornece uma visão global de um conjunto de dados, com medidas estatísticas apropriadas para cada tipo de variável. Para variáveis fatoriais, como VD (que significa "variável dependente" e representa as variantes de /r/ em coda), é apresentado o número de ocorrências de cada variante. Para variáveis numéricas, como IDADE, são apresentadas medidas como valor mínimo, 1º quartil, média etc. Tudo isso vai ser objeto das Lições 3 a 7.

Note, no entanto, que o R computou as frequências das variáveis fatoriais e certas medidas estatísticas para as variáveis numéricas porque, ao importar os dados com `read_csv()`, essas variáveis foram definidas com essas características por meio de `col_factor()`, `col_integer()` etc. As últimas três colunas, cont.precedente, ocorrencia e cont.seguinte, que foram definidas como `col_character()` – por não serem variáveis de fato –, não apresentam essas medidas estatísticas! Perceba, então, que as medidas geradas por `summary()` dependem da natureza de cada coluna/variável!

Mas e se quisermos acessar elementos específicos de um dataframe, do modo como fizemos para os vetores anteriormente? Para isso, também usamos os colchetes `[]`. Entretanto, como dataframes são estruturas bidimensionais, precisamos de dois índices para localizar um elemento – um para a linha, e outro para a coluna. Memorize essa ordem: [L, C]!

Por exemplo, para acessar o elemento da 5ª linha da 6ª coluna do conjunto `dados`, a linha de comando é `dados[5, 6]`. Note que não há espaço entre o nome do objeto e o colchete aberto, e que os índices de linha e de coluna são separados por vírgula. Rode essa linha de comando agora.

```
dados[5, 6]
```

```
## # A tibble: 1 × 1
##   ESCOLARIDADE
##   <fct>
## 1 fundamental
```

Acesse o elemento da 11ª linha da 4ª coluna do conjunto dados.

```
dados[11, 4]
```

```
## # A tibble: 1 × 1
##   IDADE
##   <int>
## 1    30
```

Acesse agora os elementos da 1ª até a 10ª linha da 9ª coluna de `dados`.

```
dados[1:10, 9]
```

```
## # A tibble: 10 × 1
##    ORIGEM.PAIS
##    <fct>
##  1 mista
##  2 mista
##  3 mista
##  4 mista
##  5 mista
##  6 mista
##  7 mista
##  8 mista
##  9 mista
## 10 mista
```

Para acessar todos os dados de uma linha, basta deixar o espaço para o índice de coluna vazio. Digite `dados[3, ]` para ver o resultado. Note que ainda assim é necessário usar a vírgula!

```
dados[3, ]
```

```
## # A tibble: 1 × 20
##   VD         PARTICIPANTE SEXO.GENERO IDADE FAIXA.ETARIA ESCOLARIDA
DE
##   <fct>      <fct>        <fct>       <int> <fct>        <fct>
## 1 retroflexo IvanaB       feminino       30 1a           fundamenta
l
```

```
## # … with 14 more variables: REGIAO <fct>, INDICE.SOCIO <dbl>,
## #   ORIGEM.PAIS <fct>, CONT.FON.PREC <fct>, CONT.FON.SEG <fct>,
## #   TONICIDADE <fct>, POSICAO.R <fct>, CLASSE.MORFOLOGICA <fct>,
## #   FREQUENCIA <dbl>, ESTILO <fct>, ITEM.LEXICAL <fct>,
## #   cont.precedente <chr>, ocorrencia <chr>, cont.seguinte <chr>
```

De modo semelhante, se deixarmos o espaço para o índice de linha vazio e preenchermos apenas a coluna, o resultado será todos os elementos da coluna respectiva. Digite `dados[, 10]` para ver os elementos da 10ª coluna do dataframe.

```
dados[, 10]
```

```
## # A tibble: 9,226 × 1
##    CONT.FON.PREC
##    <fct>
##  1 a
##  2 o
##  3 e
##  4 e
##  5 a
##  6 a
##  7 0
##  8 a
##  9 0
## 10 3
## # … with 9,216 more rows
```

No R, sempre há diversas maneiras para se chegar a um mesmo resultado. Outro modo de acessar os elementos de uma coluna de um dataframe é por meio do símbolo `$`, que já vimos acima. Digite `dados$CONT.FON.PREC` para ver os elementos da coluna CONT.FON.PREC.

```
dados$CONT.FON.PREC
```

N.B.: Resultado aqui omitido.

A principal diferença entre `[]` e `$` é que, no último caso, usamos o *nome* da variável para acessar seus elementos. Isso pode ser útil quando se sabe o nome da variável mas não se lembra em que coluna ela está (o que provavelmente vai ser o caso com seus dados).

Muitas vezes temos o interesse em acessar elementos específicos de um dataframe para criar subconjuntos de dados. O pacote dplyr, que faz parte do conjunto de pacotes do tidyverse – que carregamos mais acima –, permite criar novos dataframes de um subconjunto de linhas ou colunas, a partir de certos critérios. Digamos que você queira

criar um subconjunto apenas com os dados de /r/ em coda que ocorrem em sílabas tônicas. Na coluna TONICIDADE de `dados`, esses dados estão identificados por "tonica".

A função `filter()` toma como primeiro argumento o conjunto de dados, e outro argumento que especifica a condição a ser preenchida. Digite `dados_tonicas <- filter(dados, TONICIDADE == "tonica")`. (Note que o sinal de igual, em R, é duplo: ==).

```
dados_tonicas <- filter(dados, TONICIDADE == "tonica")
```

Crie agora o subconjunto `dados_atonas`, que contém os dados de /r/ em coda que ocorrem em sílabas átonas (identificados por "atona").

```
dados_atonas <- filter(dados, TONICIDADE == "atona")
```

Em outros casos, pode ser que você queira especificar quais variantes não devem entrar no conjunto de dados. Imagine que, na análise, você decida retirar os dados de /r/ que ocorrem em verbos do infinitivo, como em "andar", "comer", uma vez que, nesse contexto, o /r/ é quase sempre apagado. A variável de interesse aqui é CLASSE.MORFOLOGICA e o código para verbos do infinitivo é "morf.inf". Vamos guardar o resultado em um novo dataframe chamado `dados_semInf`. E a informação mais importante de que você precisa para realizar essa operação: o símbolo para "não é igual a", no R, é `!=`.

```
dados_semInf <- filter(dados, CLASSE.MORFOLOGICA != "morf.inf")
```

Crie um novo subconjunto de dados que contém apenas os dados advindos de leituras (depoimento, jornal, palavras), ou seja, sem os dados "conversacao". A variável se chama ESTILO. Guarde o resultado em um novo dataframe chamado `dados_leitura`.

```
dados_leitura <- filter(dados, ESTILO != "conversacao")
```

Agora crie um subconjunto de dados, chamado `dados_menor2`, de palavras cuja FREQUENCIA é menor do que 2% do corpus. O operador lógico do R para "menor que" é `<`. Digite `dados_menor2 <- filter(dados, FREQUENCIA < 2)`.

```
dados_menor2 <- filter(dados, FREQUENCIA < 2)
```

Mais uma! Se temos interesse em palavras cuja frequência é *menor ou igual* a 2%, o operador lógico pertinente é <=. Crie um novo subconjunto, com essa condição, chamado dados_menor_igual2.

```
dados_menor_igual2 <- filter(dados, FREQUENCIA <= 2)
```

Para criar o subconjunto de /r/ em coda em sílabas tônicas, usamos as aspas para especificar "tonica". Para criar o subconjunto de palavras menos frequentes do que 2%, não usamos as aspas. Por quê?

- Porque queremos que o R entenda "2" como um caractere
- Porque "tonica" é uma sequência de caracteres e 2 é um elemento numérico
- Porque se digitarmos tonica sem as aspas, o R tratará a variante como numérica

Podemos ser ainda mais restritivos e criar um subconjunto de dados de /r/ em coda em sílabas tônicas que ocorrem em final de palavra. O operador lógico no R para juntar duas condições é &. Digite dados_tonica_final <- filter(dados, TONICIDADE == "tonica" & POSICAO.R == "final").

```
dados_tonica_final <- filter(dados, TONICIDADE == "tonica" & POSICAO.R
== "final")
```

Podemos também ser *menos* restritivos e criar um subconjunto que contenha os dados de /r/ em coda que estejam em sílabas tônicas *ou* em sílaba final. Em relação ao conjunto dados_tonica_final, esse novo conjunto de dados (sílabas tônicas *ou* posição final)...

- Provavelmente contém um número maior de dados do que dados_tonica_final
- Provavelmente contém o mesmo número de dados de dados_tonica_final
- Provavelmente contém um número menor de dados do que dados_tonica_final

A razão para que um novo subconjunto que tenha os dados em sílabas tônicas *ou* em posição final seja provavelmente maior é o fato de que o operador lógico "ou" retorna um resultado verdadeiro se qualquer uma das duas condições é preenchida e, portanto, um maior número de observações potencialmente preencherá o critério. No caso do operador lógico &, o resultado só é verdadeiro se ambas as condições forem verdadeiras,

o que faz com que a probabilidade de haver observações que preenchem ambos os critérios seja menor.

Vamos então criar o subconjunto de dados de /r/ em sílabas tônicas *ou* em sílabas finais. O operador lógico "ou", no R, é o símbolo |. Chame esse novo conjunto de dados_tonica_ou_final.

```
dados_tonica_ou_final <- filter(dados, TONICIDADE == "tonica" | POSICAO.R == "final")
```

Usamos até agora a função `filter()`, do pacote dplyr/tidyverse, para fazer subconjuntos de dados restringindo o número de linhas (=ocorrências) do dataframe. Também é possível selecionar as colunas de um conjunto de dados para criar um novo dataframe, por meio da função `select()`.

A função `select()`, de modo semelhante à função `filter()`, também toma como primeiro argumento o dataframe original. Os demais argumentos são as colunas/variáveis que você deseja incluir. Crie um novo df chamado `dados_varsociais1`, com as variáveis SEXO.GENERO, IDADE, ESCOLARIDADE e REGIAO.

```
dados_varsociais1 <- select(dados, SEXO.GENERO, IDADE, ESCOLARIDADE, REGIAO)
```

Também é possível usar o operador : para selecionar desde uma coluna até outra. Crie um novo dataframe, chamado `dados_varsociais2`, que contém as colunas a partir de SEXO.GENERO até REGIAO.

```
dados_varsociais2 <- select(dados, SEXO.GENERO:REGIAO)
```

Você pode comparar os dois dfs em Environment: enquanto `dados_varsociais1` tem quatro colunas, `dados_varsociais2` tem uma coluna a mais, pois também inclui a variável FAIXA.ETARIA. Se quiser, veja novamente o dataframe `dados` completo, aberto em uma das janelas em Source.

Nesta lição, você aprendeu uma série de funções para manipular conjuntos de dados no R, seja um vetor, seja um dataframe: `ncol()`, `nrow()`, `str()`, `summary()`, `filter()`, `select()` – as duas últimas funções, do pacote dplyr. Aprendeu também alguns operadores lógicos do R: ==, !=, <, <=, &, |.

Você pode estar com a sensação de que entendeu o conteúdo, mas que não vai se lembrar de usar a função adequada quando realmente precisar... A solução para isso é praticar! Faça os exercícios correspondentes a esta lição!

Para saber mais

Para saber mais sobre as funções do pacote dplyr, recomendo visitar a página https://dplyr.tidyverse.org/. Se quiser praticar mais, recomendo que você faça as lições 3 ("Sequences of Numbers"), 4 ("Vectors"), 6 ("Subsetting Vectors") e 7 ("Matrices and Data Frames") do curso R Programming, do swirl. Recomendo também a leitura do capítulo 2 de Gries (2019).

Exercícios

1. Carregue o pacote tidyverse na memória do R.
2. Defina o diretório de trabalho como aquele em que se encontra a planilha DadosRT.csv.
3. Carregue o conjunto de dados da planilha DadosRT num dataframe chamado `dadosR`. Use a função `read_csv()` para isso, e especifique as colunas como "factor", exceto IDADE, que deve ser especificada como "integer", e INDICE.SOCIO e FREQUENCIA, que devem ser especificadas como "double".
4. Inspecione a estrutura do dataframe `dadosR` com a função `str()`.
5. A função `str()` fornece uma visão global das propriedades de um dataframe e, como tal, permite checar rapidamente se um conjunto de dados foi corretamente carregado e se sua planilha não tem problemas. Qual é o número de linhas no dataframe `dadosR`?
6. Qual é o número de colunas no dataframe `dadosR`?
7. Quantos fatores têm as variáveis FAIXA.ETARIA e ORIGEM.PAIS, respectivamente?
    a. 3 e 5
    b. 2 e 105

c. 7 e 15

8. Que tipo de variável é IDADE?
    a. Factor
    b. int
    c. num
9. As três últimas colunas de `dadosR` mostram a ocorrência da palavra com /r/ em coda com algumas palavras do contexto precedente e seguinte. Elas servem como base para a codificação das demais colunas, mas não são variáveis em si. Crie um novo dataframe, chamado `dadosR2`, com a exclusão dessas três colunas.
10. As funções `head()` e `tail()` mostram as primeiras e as últimas ocorrências de um objeto. Elas admitem mais um argumento, que é o número de elementos que se deseja visualizar. Veja a documentação da função `head()` na Ajuda.
11. O número *default* da função `head()` é 6, mas caso você queira visualizar as 15 primeiras linhas, basta incluir 15 como segundo argumento da função. Aplique a função `head()` a `dadosR2` para visualizar as 15 primeiras linhas desse dataframe.
12. Visualize o elemento que está na 45ª linha da 10ª coluna de `dadosR2`.
13. Visualize os dados da 5ª coluna de `dadosR2`.
14. Visualize os dados da variável TONICIDADE de `dadosR2`.
15. Visualize os 6 primeiros dados da variável TONICIDADE de `dadosR2`.
16. Visualize os 20 primeiros dados da variável TONICIDADE de `dadosR2`.
17. Guarde os 20 primeiros dados da variável TONICIDADE de `dadosR2` em um objeto chamado `tonicidade20`.
18. Que tipo de estrutura de dado é `tonicidade20`?
    a. Um dataframe
    b. Um vetor
    c. Um vetor e um dataframe

19. Acesse o 10º elemento do vetor `tonicidade20`.
20. Acesse o 5º e o 3º elementos do vetor `tonicidade20`.
21. Acesse o intervalo do 5º ao 10º elemento do vetor `tonicidade20`.
22. Aplique a função `length()` para saber quantos dados há na coluna CONT.FON.SEG de `dadosR2`.
23. Na prática, a função `length()` aplicada a uma coluna de um dataframe fornece o número de observações/linhas, uma vez que todas as colunas de um dataframe têm a mesma extensão. Quais outras funções também informam o número de observações/linhas de um dataframe?
    a. `head()` e `str()`
    b. `nrow()` e `str()`
    c. `nrow()` e `summary()`
    d. `str()` e `summary()`
24. Na sequência, serão criados vários subconjuntos de dados com a função `filter()`. Use o dataframe `dadosR2` em todos os casos. Crie um novo dataframe chamado `dados_1a` que contém todos os dados dos falantes de 1ª faixa etária. A variável relevante se chama FAIXA.ETARIA e a variante é "1a".
25. Crie um subconjunto de dados chamado `dados_1a2a` que contém todos os dados dos falantes de 1ª e 2ª faixas etárias. Use um operador lógico nessa linha de comando.
26. Crie um subconjunto de dados chamado `dados_menor45` com os dados dos falantes que têm menos de 45 anos. Se precisar olhar o conjunto de dados, use a função `View()` ou `str()` para visualizar os nomes relevantes de variáveis e variantes.
27. Crie um subconjunto de dados chamado `dados_menor_igual45` com os dados dos falantes que têm 45 anos ou menos.
28. Crie um subconjunto de dados chamado `dados_maior_igual46` com os dados dos falantes que têm 46 anos ou mais.

29. Crie um subconjunto de dados chamado `dados_1a_central` com os dados dos falantes de primeira faixa etária e que vivem na região central da cidade.
30. Crie um subconjunto de dados chamado `dados_1a_ou_central` com os dados dos falantes de primeira faixa etária *ou* que vivem na região central da cidade.
31. Crie um subconjunto de dados com os critérios que desejar e guarde-o em um objeto. Use a função `filter()` para isso. Cheque na aba Environment se o objeto foi criado corretamente.
32. Crie um novo dataframe chamado `dadosR2_varsociais`, que contém as colunas VD, PARTICIPANTE, SEXO.GENERO, IDADE, FAIXA.ETARIA, ESCOLARIDADE, REGIAO, INDICE.SOCIO e ORIGEM.PAIS.

Lição 3: Tipos de Variáveis

Para esta lição, vamos usar novamente o conjunto de dados DadosRT.csv. Então carregue inicialmente o pacote tidyverse, para ter acesso à função `read_csv()`.

```
library(tidyverse)
```

Defina como diretório de trabalho aquele que contém, em seu computador, o arquivo DadosRT.csv.

```
setwd("~/Dropbox/_R/swirl/Introducao_a_Estatistica_para_Linguistas/data")
```

E rode as linhas de comando no *script* com a função `read_csv()`, para carregar a planilha de dados na memória desta sessão.

```
dados <- read_csv("DadosRT.csv",
                  col_types = cols(.default = col_factor(),
                                   cont.precedente = col_character(),
                                   ocorrencia = col_character(),
                                   cont.seguinte = col_character(),
                                   IDADE = col_integer(),
                                   INDICE.SOCIO = col_double(),
                                   FREQUENCIA = col_double()
                                   )
                  )
```

O R classifica variáveis em seis tipos diferentes. Para nossos propósitos, quatro deles são mais importantes: int (=integer), numeric, character e factor. Na Lição 1, vimos vetores numéricos (como `aleatorio` e `x`) e vetores de caracteres (como `z`). Ao carregar os dados da planilha DadosRT.csv, especificamos as variáveis como factor, character, integer e double. As variáveis int e factor relacionam-se com as variáveis numéricas e de caracteres, mas têm certas especificidades.

Aplique a função `str()` a `dados` para que possamos examinar as propriedades de cada variável.

```
str(dados)
```

```
## spec_tbl_df [9,226 × 20] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ VD                : Factor w/ 2 levels "retroflexo","tepe": 1 1
1 2 1 1 1 1 2 1 ...
##  $ PARTICIPANTE      : Factor w/ 118 levels "IvanaB","HeloisaS",..:
1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ SEXO.GENERO       : Factor w/ 2 levels “feminino”,”masculino”: 1
1 1 1 1 1 1 1 1 1 ...
##  $ IDADE             : int [1:9226] 30 30 30 30 30 30 30 30 30 30 .
..
##  $ FAIXA.ETARIA      : Factor w/ 3 levels “1a”,”3a”,”2a”: 1 1 1 1 1
1 1 1 1 1 ...
##  $ ESCOLARIDADE      : Factor w/ 3 levels “fundamental”,..: 1 1 1 1
1 1 1 1 1 1 ...
##  $ REGIAO            : Factor w/ 2 levels “periferica”,”central”: 1
1 1 1 1 1 1 1 1 1 ...
##  $ INDICE.SOCIO      : num [1:9226] 2 2 2 2 2 2 2 2 2 2 ...
##  $ ORIGEM.PAIS       : Factor w/ 5 levels “mista”,”SPcapital”,..: 1
1 1 1 1 1 1 1 1 1 ...
##  $ CONT.FON.PREC     : Factor w/ 7 levels “a”,”o”,”e”,”0”,..: 1 2 3
3 1 1 4 1 4 5 ...
##  $ CONT.FON.SEG      : Factor w/ 19 levels “ts”,”n”,”g”,”v”,..: 1 2
3 4 5 6 7 7 5 8 ...
##  $ TONICIDADE        : Factor w/ 2 levels “tonica”,”atona”: 1 2 2 2
1 1 1 1 2 1 ...
##  $ POSICAO.R         : Factor w/ 2 levels “medial”,”final”: 1 1 1 1
2 2 1 1 1 1 ...
##  $ CLASSE.MORFOLOGICA: Factor w/ 6 levels “substantivo”,..: 1 1 1 1
1 1 1 1 2 1 ...
##  $ FREQUENCIA        : num [1:9226] 1.34 0.16 0.22 0.44 1.94 1.94 0
.35 0.03 5.98 0.16 ...
##  $ ESTILO            : Factor w/ 4 levels “conversacao”,..: 1 1 1 1
1 1 1 1 1 1 ...
##  $ ITEM.LEXICAL      : Factor w/ 1151 levels “parte”,”jornal”,..: 1
2 3 4 5 5 6 7 8 9 ...
##  $ cont.precedente   : chr [1:9226] “do CEU é daquela” “eu não gost
o de” “não (es)to(u) entendendo a” “o que sei... num” ...
##  $ ocorrencia        : chr [1:9226] “parte <R>“ “jornal <R>“ “pergu
nta <R>“ “serviço <T>“ ...
##  $ cont.seguinte     : chr [1:9226] “que as perua(s) ia” “D1: mas d
igo assim” “D1: o que que/” “a não ser <A>“ ...
##  - attr(*, “spec”)=
##   .. cols(
##   ..   .default = col_factor(),
##   ..   VD = col_factor(levels = NULL, ordered = FALSE, include_na =
FALSE),
##   ..   PARTICIPANTE = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   SEXO.GENERO = col_factor(levels = NULL, ordered = FALSE, inc
lude_na = FALSE),
##   ..   IDADE = col_integer(),
##   ..   FAIXA.ETARIA = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   ESCOLARIDADE = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   REGIAO = col_factor(levels = NULL, ordered = FALSE, include_
na = FALSE),
##   ..   INDICE.SOCIO = col_double(),
##   ..   ORIGEM.PAIS = col_factor(levels = NULL, ordered = FALSE, inc
lude_na = FALSE),
```

```
##   ..   CONT.FON.PREC = col_factor(levels = NULL, ordered = FALSE, i
nclude_na = FALSE),
##   ..   CONT.FON.SEG = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   TONICIDADE = col_factor(levels = NULL, ordered = FALSE, incl
ude_na = FALSE),
##   ..   POSICAO.R = col_factor(levels = NULL, ordered = FALSE, inclu
de_na = FALSE),
##   ..   CLASSE.MORFOLOGICA = col_factor(levels = NULL, ordered = FAL
SE, include_na = FALSE),
##   ..   FREQUENCIA = col_double(),
##   ..   ESTILO = col_factor(levels = NULL, ordered = FALSE, include_
na = FALSE),
##   ..   ITEM.LEXICAL = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   cont.precedente = col_character(),
##   ..   ocorrencia = col_character(),
##   ..   cont.seguinte = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Como já vimos, por meio da função `str()`, o R mostra a estrutura de um objeto. Para um dataframe, depois de mostrar o número de linhas (=observações) e colunas (=variáveis), o R lista as variáveis identificadas por `$` e, para cada uma, reporta como a variável está classificada.

Como a variável IDADE (que indica a idade de cada falante em anos) está classificada?

- int
- num
- factor

Como a variável INDICE.SOCIO (que indica a categorização dos falantes dentro de um índice que vai de 0 a 5) está classificada?

- int
- num
- factor

Como a variável FREQUENCIA (que indica a proporção de determinado item lexical no *corpus*) está classificada?

- int

- num
- factor

IDADE está classificada como 'int' (= números inteiros), enquanto INDICE.SOCIO e FREQUENCIA estão classificadas como 'num'. A diferença do segundo para o primeiro é o fato de que 'num' admite valores numéricos com casas decimais. Fomos nós que informamos ao R que essas variáveis são numéricas no momento de carregar a planilha no dataframe `dados`. No segundo argumento de `read_csv()`, col_types, indicamos a variável IDADE como `col_integer()`, e INDICE.SOCIO e FREQUENCIA como `col_double()`. `double` especifica "dupla precisão", daí a leitura das casas decimais.

Como está classificada a variável FAIXA.ETARIA (que separa os falantes em três categorias: "1a", de 20 a 34 anos; "2a", de 35 a 59 anos; e "3a", de 60 ou mais anos)?

- int
- num
- factor

A variável FAIXA.ETARIA é um reagrupamento dos falantes de acordo com sua IDADE, em três categorias. No entanto, apesar de falarmos em "1a", "2a" ou "3a" faixas etárias, não se trata de uma variável verdadeiramente numérica. Por exemplo, um falante de 36 anos – da 2ª faixa etária – não tem o dobro de idade de um falante de 25 – da 1ª faixa etária, de modo que a relação entre 1 e 2 não é quantitativa, mas *qualitativa*. Contudo, o R não sabe disso. A variável FAIXA.ETARIA poderia ter sido codificada como "A", "B" e "C", ou como "1", "2" e "3". O R classificou FAIXA.ETARIA como um fator porque assim especificamos na função `read_csv()` – usando `col_factor()` com opção *default* para as colunas. Factor é um tipo específico de vetores de caracteres que têm uma propriedade adicional: fatores têm níveis que normalmente ocorrem ao menos uma vez.

Acesse os elementos da coluna FAIXA.ETARIA, digitando `dados$FAIXA.ETARIA`.

```
dados$FAIXA.ETARIA
```

N.B.: Resultado aqui omitido.

Veja que o R não só mostra os elementos dessa coluna, mas também indica, ao final, quais são os níveis (=levels) dessa variável: 1a, 3a e 2a. Acesse agora apenas o primeiro elemento da coluna FAIXA.ETARIA.

```
dados$FAIXA.ETARIA[1]
```

```
## [1] 1a
## Levels: 1a 3a 2a
```

Veja que, mesmo mostrando um único elemento, o R mostra quais são todos os níveis possíveis da variável, e isso porque a variável FAIXA.ETARIA está classificada como factor, e não como character.

Com a função `read_csv()` do pacote tidyverse, é o próprio usuário que define o tipo de cada variável de seu conjunto de dados. A definição adequada para cada variável é importante porque sua natureza determina que tipos de operações e análises podem ser feitas sobre cada uma delas. Por exemplo, para variáveis numéricas (int e num), é possível calcular médias e medianas; para variáveis fatoriais, é possível calcular frequências (como vimos com a função `summary()` na última lição). O R não sabe quais operações são adequadas para cada tipo de dado – cabe a você determinar isso.

Você deve ter notado que o R está mostrando os níveis da variável FAIXA.ETARIA numa ordem "não lógica": 1a, 3a, 2a. Quando carregamos variáveis como factor com a função `read_csv()`, o *default* é que elas sejam ordenadas na mesma sequência em que aparecem na planilha. Contudo, a variável FAIXA.ETARIA, além de ser uma variável nominal/qualitativa (não numérica), é também uma variável ordinal: seus níveis se organizam hierarquicamente.

Novamente, esta é uma informação que o R não tem, pois somente o pesquisador pode determinar a natureza de uma variável.

Das variáveis de `dados`, quais outras também são ordinais?

- ESCOLARIDADE e FREQUENCIA
- REGIAO e CONT.FON.PREC
- ORIGEM.PAIS e CLASSE.MORFOLOGICA

Você pode ter achado estranho FREQUENCIA ser uma variável ordinal, uma vez que ela é numérica. Entretanto, toda variável numérica é também ordinal – embora nem toda variável ordinal seja numérica! E toda variável ordinal também pode ser nominal – embora nem toda variável nominal seja ordinal!

A ordem dos níveis de uma variável fatorial vai ser importante adiante na hora de plotar gráficos e ver o resultado de modelos estatísticos. Então, logo após importar os dados de uma planilha, é importante checar como o R está organizando os níveis dessas variáveis e, se necessário, reordená-los.

Um modo de checar os níveis de uma variável (além do que fizemos mais acima) é aplicar a função `levels()` a um vetor. Aplique essa função à coluna FAIXA.ETARIA de `dados`.

```
levels(dados$FAIXA.ETARIA)
```

```
## [1] "1a" "3a" "2a"
```

Entretanto, quando se tem um dataframe complexo, como é `dados`, pode ser muito trabalhoso aplicar essa função coluna a coluna. Um modo mais fácil de fazer isso para múltiplas colunas é aplicando a função `lapply()`, que toma como primeiro argumento o objeto a ser manipulado e como segundo argumento o nome da função que se quer aplicar. Aplique então a função `lapply()` a `dados` para executar a função levels em todas as suas colunas.

```
lapply(dados, levels)
```

```
## $VD
## [1] "retroflexo" "tepe"
##
## $PARTICIPANTE
##   [1] "IvanaB"      "HeloisaS"    "DercyF"      "SergioP"
##   [5] "AmandaA"     "PamelaR"     "PatriciaT"   "GiovanaA"
##   [9] "EdnaS"       "InesC"       "GilsonS"     "AdolfoF"
## [...]
##
## $SEXO.GENERO
## [1] "feminino"  "masculino"
##
## $IDADE
## NULL
##
## $FAIXA.ETARIA
```

```
## [1] “1a” “3a” “2a”
##
## $ESCOLARIDADE
## [1] “fundamental” “superior”    “medio”
##
## $REGIAO
## [1] “periferica” “central”
##
## $INDICE.SOCIO
## NULL
##
## $ORIGEM.PAIS
## [1] “mista”       “SPcapital”   “interior”    “nordeste”
## [5] “estrangeira”
##
## $CONT.FON.PREC
## [1] “a” “o” “e” “0” “3” “u” “i”
##
## $CONT.FON.SEG
##  [1] “ts” “n”  “g”  “v”  “k”  “dz” “t”  “s”  “d”  “#”  “m”  “l”  “p
”
## [14] “f”  “j”  “b”  “z”  “h”  “x”
##
## $TONICIDADE
## [1] “tonica” “atona”
##
## $POSICAO.R
## [1] “medial” “final”
##
## $CLASSE.MORFOLOGICA
## [1] “substantivo” “conj.prep”   “adverbio”    “verbo”
## [5] “adjetivo”    “morf.inf”
##
## $FREQUENCIA
## NULL
##
## $ESTILO
## [1] “conversacao” “palavras”    “jornal”      “depoimento”
##
## $ITEM.LEXICAL
##    [1] “parte”            “jornal”           “pergunta”
##    [4] “serviço”          “lugar”            “porta”
##    [7] “Marta”            “porque”           “comércio”
##   [10] “absurdo”          “melhor”           “suportei”
##   [13] “perdia”           “quarta”           “concordo”
## [...]
##
## $cont.precedente
## NULL
##
## $ocorrencia
## NULL
##
```

```
## $cont.seguinte
## NULL
```

N.B.: Resultado aqui abreviado.

O resultado é extenso, pois trata-se de um dataframe com 20 colunas! Como já sabíamos, a variável FAIXA.ETARIA não está ordenada de modo "lógico". Para reordenar os níveis de uma variável fatorial, podemos usar a função `fct_relevel()`, do pacote forcats, que faz parte do conjunto de pacotes do tidyverse (já carregado no início da lição). Aplique essa função à coluna FAIXA.ETARIA, usando sort como segundo argumento, para colocarmos os níveis em ordem alfabética: 1a, 2a e 3a. Além disso, guarde o resultado da operação novamente em `dados$FAIXA.ETARIA`, para que o dataframe seja atualizado.

```
dados$FAIXA.ETARIA <- fct_relevel(dados$FAIXA.ETARIA, sort)
```

Quer saber se o R fez a devida modificação nos níveis de FAIXA.ETARIA? Cheque novamente com a função `levels()`, voltando a essa linha de comando com a flecha para cima.

```
levels(dados$FAIXA.ETARIA)
## [1] "1a" "2a" "3a"
```

Veja agora, no resultado de `lapply()`, a ordem dos níveis de ESCOLARIDADE. Eles estão incorretos, não é? Por coincidência, a ordem "lógica" – fundamental, medio, superior – também segue a ordem alfabética, de modo que também podemos aplicar `fct_relevel()` com sort para reordená-los. Faça isso agora, guardando o resultado da linha de comando em `dados$ESCOLARIDADE`.

```
dados$ESCOLARIDADE <- fct_relevel(dados$ESCOLARIDADE, sort)
```

Reveja agora os níveis da variável VD no resultado de `lapply()` mais acima. Os níveis estão organizados como "retroflexo" e "tepe", a ordem em que aparecem na planilha. Bem mais adiante, na Lição 14, veremos que os resultados de modelos de regressão logística são fornecidos de acordo com o segundo nível da variável resposta – que, no momento, é o tepe! Digamos que um pesquisador tenha interesse em visualizar

os resultados de acordo com a variante retroflexa. Para obter esse resultado, é necessário mudar a ordem dos níveis dessa variável.

Nesse caso, os níveis já estão (coincidentemente) em ordem alfabética, de modo que não faz sentido usar o argumento sort. Outro modo de aplicar a função `fct_relevel()` é especificando, como segundo argumento, qual é o nível que se quer como primeiro. O nível especificado deve vir entre aspas. Faça isso agora, guardando o resultado em `dados$VD`.

```
dados$VD <- fct_relevel(dados$VD, "tepe")
```

Faça a mesma operação que fizemos acima com a variável ORIGEM.PAIS, colocando "SPcapital" como primeiro nível dessa variável. Não se esqueça de guardar o resultado no dataframe!

```
dados$ORIGEM.PAIS <- fct_relevel(dados$ORIGEM.PAIS, "SPcapital")
```

No caso da variável VD, só há duas variantes/dois níveis. Em ORIGEM.PAIS, há vários outros níveis – mista, interior, nordeste, estrangeira – mas, ao colocar SPcapital como primeiro nível, estamos indicando que a ordem dos demais não importa, e que – por motivos que ficarão mais claros na Lição 14 – interessa apenas ter "SPcapital" como referência.

Em outros casos, no entanto, podemos querer redefinir a ordem de todos os níveis. Tomemos como exemplo a variável CONT.FON.PREC, que codifica qual vogal ocorreu antes do /r/ em coda. Seus níveis são as sete vogais orais do português brasileiro: a e ɛ i o ɔ u. As vogais /ɛ/ e /ɔ/ foram codificadas como "3" e como "0", respectivamente, para evitar o uso de caracteres especiais. Embora CONT.FON.PREC não seja uma variável ordinal, pode fazer mais sentido ordenar as vogais de acordo com algum critério fonológico, como o traço de anterioridade – de [+anterior] a [-anterior], de modo que seus níveis sejam i e 3 a 0 o u.

Aqui, não podemos usar o argumento sort, tampouco basta especificar apenas o primeiro nível da variável. Para casos como esse, podemos especificar todos os níveis como diferentes argumentos da função `fct_relevel()`, como a linha de comando já pronta no *script*. Rode-a agora.

```
dados$CONT.FON.PREC <- fct_relevel(dados$CONT.FON.PREC,
                                   "i", "e", "3", "a", "0", "o", "u")
```

Os níveis da variável ESTILO são "conversacao", "palavras", "jornal" e "depoimento". Eles codificam de que ponto da entrevista sociolinguística o dado foi extraído, de acordo com a hipótese laboviana de "grau de atenção à fala". Do modo como o roteiro da entrevista foi elaborado, espera-se que o falante esteja prestando menos atenção durante a conversação com o documentador, e que preste progressivamente mais atenção em três estilos de leitura: depoimento – que é a leitura de um texto informal –, jornal – a leitura de uma notícia – e palavras – a leitura de palavras isoladas.

Aqui, coincidentemente, a ordem de "grau de atenção" é a mesma que a ordem alfabética, o que significa que poderíamos usar o argumento sort. No entanto, como nem sempre temos essa sorte, vale aplicar a função `fct_relevel()` com especificação explícita dos níveis, apenas para treino. Digite a linha de comando com os níveis na ordem desejada, sem esquecer de guardar o resultado.

```
dados$ESTILO <- fct_relevel(dados$ESTILO,
                            "conversacao", "depoimento", "jornal",
                            "palavras")
```

Com a função `fct_relevel()`, estamos reorganizando os níveis das variáveis em diferentes linhas de comando. No entanto, também é possível fazer isso no momento da importação de dados, com a função `read_csv()`. Examine a linha de comando abaixo. Ela é semelhante ao comando que usamos mais acima, mas, desta vez, também estamos especificando a ordem dos níveis de FAIXA.ETARIA com o argumento levels da função `col_factor()`. Rode essa linha de comando.

```
dados <- read_csv("DadosRT.csv",
                  col_types = cols(.default = col_factor(),
                                   cont.precedente = col_character(),
                                   ocorrencia = col_character(),
                                   cont.seguinte = col_character(),
                                   IDADE = col_integer(),
                                   INDICE.SOCIO = col_double(),
                                   FREQUENCIA = col_double(),
                                   FAIXA.ETARIA = col_factor(levels =
c("1a",  "2a", "3a"))
                                   )
                  )
```

E cheque a ordem dos níveis da variável FAIXA.ETARIA com a função `levels()`.

```
levels(dados$FAIXA.ETARIA)

## [1] “1a” “2a” “3a”
```

Nesse último exemplo, redefinimos apenas os níveis de FAIXA.ETARIA, mas é possível fazer isso com todas as variáveis fatoriais simplesmente acrescentando novos argumentos na função `cols()`. Volte, então, à mesma linha de comando com `read_csv()`. Ao final da linha com redefinição dos níveis de FAIXA.ETARIA, inclua uma vírgula (para poder acrescentar um novo argumento) e pressione ENTER para mudar de linha. Depois, redefina os níveis de ESCOLARIDADE para “fundamental”, “medio”, “superior”.

```
dados <- read_csv(“DadosRT.csv”,
                  col_types = cols(.default = col_factor(),
                                   cont.precedente = col_character(),
                                   ocorrencia = col_character(),
                                   cont.seguinte = col_character(),
                                   IDADE = col_integer(),
                                   INDICE.SOCIO = col_double(),
                                   FREQUENCIA = col_double(),
                                   FAIXA.ETARIA = col_factor(levels =
c(“1a”,  “2a”, “3a”)),
                                   ESCOLARIDADE = col_factor(levels =
c(“fundamental”, “medio”, “superior”))
                                   )
                  )
```

Pode parecer trabalhoso redefinir os níveis de todas as variáveis fatoriais desse modo, em uma única linha de comando, em vez de fazê-lo passo a passo, como fizemos com `fct_relevel()`. Um dos motivos disso é que, para produzi-la, você precisa saber quais são todas as variáveis fatoriais de seus dados, quais são os níveis de cada variável e em qual ordem você os quer. Entretanto, a principal dificuldade que você está sentindo aqui se deve ao fato de que este não é o *seu* conjunto de dados. Quando estiver trabalhando com sua própria planilha, você certamente saberá quais são as variáveis e suas respectivas variantes!

Além disso, essa é a grande vantagem de se trabalhar com *scripts*: não é necessário digitar uma linha de comando completa de primeira. Você pode, por exemplo, importar primeiramente os dados e checar os níveis com `lapply()`, como fizemos mais acima. Ao

verificar que os níveis de algumas variáveis precisam ser redefinidos, você pode voltar à linha de comando já digitada, acrescentar um novo argumento e rodá-la novamente. No fundo, o trabalho de digitar mais um argumento em `read_csv()` é o mesmo de digitar várias linhas de comando separadas com `fct_relevel()`!

Para aproveitar que estamos no mundo do pacote forcats, vale a pena apresentar uma função para amalgamar os níveis de uma variável fatorial, algo que o pesquisador pode querer fazer por motivos vários: porque não há dados suficientes para um dos níveis, porque não há diferença relevante entre certos níveis, porque faz sentido dentro da teoria linguística.

Para exemplificar, vamos usar a variável CONT.FON.SEG. De modo semelhante a CONT.FON.PREC, a variável codifica o contexto fonológico de ocorrência de /r/ em coda – neste caso, qual é a consoante que vem em seguida, ou ainda se é uma pausa. Essa variável foi originalmente codificada de modo detalhado, pois é fácil juntar fatores, mas dá mais trabalho separá-los (seria necessário recodificá-los um a um!). Seguindo a hipótese que se quer testar, o contexto fônico seguinte pode ser reorganizado de acordo com o ponto de articulação, ou modo de articulação, ou a sonoridade do segmento, ou o Ponto de C etc.

Para juntar fatores em um mesmo nível, vamos usar a função `fct_collapse()`, que pode ser vista no *script*. Assim como a função `fct_relevel()`, o primeiro argumento é o vetor/coluna que se quer modificar. Os demais argumentos devem especificar, um a um, o nome do novo nível à esquerda do sinal de igual, e os valores a amalgamar à direita do sinal de igual. Note o uso da função `c()` para juntar os fatores; quando há apenas um fator, como no caso da categoria `pausa`, não é necessário aplicar a função de concatenação. Isso também indica que a função pode ser usada para renomear os níveis de uma variável. Observe também que, em vez de guardar o resultado num vetor de mesmo nome, estamos criando uma nova coluna, chamada `CONT.FON.SEG_PC`, e inserindo-a no dataframe. Isso pode ser interessante para preservar a variável original. Rode então essa linha de comando, que já está pronta.

```
dados$CONT.FON.SEG_PC <- fct_collapse(dados$CONT.FON.SEG,
                                      coronal = c("ts", "dz", "t", "d"
, "n", "s", "z", "x", "j", "l"),
                                      dorsal = c("k", "g", "h"),
                                      labial = c("f", "v", "p", "b", "
m"),
                                      pausa = "#")
```

Você pode ver, em Environment, que o dataframe `dados`, antes com 20 colunas, agora tem 21. A variável CONT.FON.SEG_PC foi adicionada como última coluna do dataframe.

Agora é a sua vez! Vamos criar uma nova variável a partir de CONT.FON.PREC, amalgamando as vogais precedentes em dois níveis, definidos pela altura da vogal. A estrutura desse comando está no *script*, mas você deve completá-lo para que o R possa interpretá-lo corretamente. A nova variável já está definida: `dados$CONT.FON.PREC_altura`. Insira nos pontos devidos as seguintes informações: (i) o vetor original cujos níveis serão amalgamados; (ii) o nível alta, com os fatores i, e, u e o (nessa ordem); e (iii) o nível baixa, com os fatores 3, a e 0 (nessa ordem).

```
dados$CONT.FON.PREC_altura <- fct_collapse(dados$CONT.FON.PREC,
                                           alta = c("i", "e", "u", "o"
),
                                           baixa = c("3", "a", "0")
                                           )
```

Nesta lição, vimos que o R classifica as variáveis de um dataframe em int, num, chr ou factor. A verdadeira natureza de cada variável é determinada pelo pesquisador e deve ser levada em conta no momento de codificação e importação dos dados ao R. Também é possível redefinir os níveis de uma variável fatorial usando a função `fct_relevel()`, e juntar fatores com a função `fct_collapse()`, ambas do pacote forcats/tidyverse.

Para saber mais

Recomendo a leitura de Wickham (2014) e as páginas 28–31 de Gries (2019) sobre a coleta e a organização de dados.

Exercícios

A planilha de dados é normalmente preparada fora do ambiente do R, em programas como o Excel ou o Calc. O modo como cada variável é codificada na planilha e importada ao R depende de escolhas do pesquisador e da natureza das variáveis sob análise. Com isso em mente, responda as questões a seguir.

1. Qual é o melhor modo de codificar a variável IDADE, a ser lida pelo R como uma variável como numérica/int?
    a. 20 anos, 21 anos, 22 anos...
    b. 20, 21, 22...
    c. vinte, vinte e um, vinte e dois...
2. Caso um pesquisador codifique a variável CLASSE.SOCIAL como baixa, media.baixa, media etc., como a função `read_csv()` vai ler essa coluna, sem qualquer especificação adicional?
    a. como character
    b. como int
    c. como num
3. Como a variável CLASSE.SOCIAL codificada como baixa, media.baixa, media etc. deve ser especificada em `col_types()`, dentro de `read_csv()`?
    a. `col_character()`
    b. `col_double()`
    c. `col_factor()`
    d. `col_integer()`
4. Qual é a ordem em que `read_csv()`, do pacote tidyverse, colocará os níveis de CLASSE.SOCIAL, se codificada como baixa, media.baixa, media, media.alta, alta?
    a. na ordem em que aparecem na planilha

   b. na ordem lógica – baixa, media.baixa, media, media.alta, alta
   c. na ordem alfabética – alta, baixa, media, media.alta, media.baixa

5. Rode a linha de comando a seguir e visualize o vetor CLASSE.SOCIAL.

```
CLASSE.SOCIAL <- as.factor(c("alta", "baixa",
                             "media", "media_baixa", "media_alta"))
```

6. Reorganize os níveis da variável CLASSE.SOCIAL numa ordem "lógica", de classe mais baixa para classe mais alta. Use a função `fct_relevel()`. Não é necessário guardar o resultado.
7. Reorganize os níveis da variável CLASSE.SOCIAL, juntando os fatores relativos à classe média (media_baixa, media e media_alta) em um único fator chamado "media". Guarde o resultado em um vetor chamado CLASSE.SOCIAL2.
8. Reorganize os níveis da variável CLASSE.SOCIAL2 na ordem lógica ascendente: baixa, media e alta. Não é necessário guardar o resultado.
9. Costuma-se classificar variáveis em 3 tipos: nominais (ou categóricas); ordinais; e numéricas (Gries, 2019, p.25–26). O R as classifica de modo diferente (int, num, double, factor), mas há uma relação entre elas. Escolha a opção que *não* faz uma relação correta.
   a. categórica – factor
   b. categórica – num
   c. numérica – double
   d. numérica – int
   e. numérica – num
   f. ordinal – factor, int ou num
10. Que tipo de variável é CLASSE.MORFOLOGICA (codificada como substantivo, adjetivo, verbo etc.)?
   a. nominal
   b. numérica
   c. ordinal

11. Como a variável CLASSE.MORFOLOGICA (codificada como "substantivo", "adjetivo", "verbo" etc.) deve ser especificada em `col_types()`, dentro de `read_csv()`?
    a. `col_character()`
    b. `col_double()`
    c. `col_factor()`
    d. `col_integer()`
12. Qual das categorizações a seguir não se aplica a FREQUENCIA.LEXICAL, codificada como muito.frequente, frequente, pouco.frequente, infrequente?
    a. categórica
    b. nominal
    c. numérica
    d. ordinal
13. Como a variável FREQUENCIA.LEXICAL (codificada como muito.frequente, frequente, pouco.frequente, infrequente) deve ser especificada em `col_types()`, dentro de `read_csv()`?
    a. `col_character()`
    b. `col_double()`
    c. `col_factor()`
    d. `col_integer()`
14. Qual das categorizações a seguir não se aplica a FREQUENCIA.LEXICAL, codificada como o número de ocorrências por mil palavras (0,01, 0,02, 0,03, 0,5, 1,0 etc.)?
    a. nominal
    b. numérica
    c. ordinal
15. Como a variável FREQUENCIA.LEXICAL, codificada como o número de ocorrências por mil palavras (0,01, 0,02, 0,03, 0,5, 1,0 etc.) deve ser especificada em `col_types()`, dentro de `read_csv()`?
    a. `col_character()`
    b. `col_double()`
    c. `col_factor()`

   d. `col_integer()`

16. Qual das categorizações a seguir não se aplica a F1, codificada como as medições de F1 em Hertz de diferentes vogais (300, 325, 450 etc.)?
    a. nominal
    b. numérica
    c. ordinal
17. Como a variável F1, codificada como as medições de F1 em Hertz de diferentes vogais (300, 325, 450 etc.), deve ser especificada em `col_types()`, dentro de `read_csv()`?
    a. `col_character()`
    b. `col_factor()`
    c. `col_integer()`
18. Defina como diretório de trabalho aquele que contém a planilha DadosRT.csv em seu computador.
19. Carregue o pacote tidyverse.
20. Use a função `read_csv()` para carregar a planilha DadosRT.csv em um dataframe chamado `dadosRT`. Ao importar os dados: (i) especifique o *default* das variáveis como `col_factor()`; (ii) especifique a variável IDADE como `col_integer()`; (iii) especifique as variáveis INDICE.SOCIO e FREQUENCIA como `col_double()`; e defina a ordem dos níveis da variável CONT.FON.PREC como "i", "e", "3", "a", "0", "o" e "u".
21. Aplique a função `levels()` para visualizar os níveis da variável POSICAO.R do dataframe `dadosRT`.
22. Reorganize os níveis da variável POSICAO.R na ordem "final" e "medial". Não é necessário guardar o resultado.
23. Aplique a função `levels()` para visualizar os níveis da variável TONICIDADE do dataframe `dadosRT`.
24. Reorganize os níveis da variável TONICIDADE na ordem "atona" e "tonica". Não é necessário guardar o resultado.

25. Aplique a função `levels()` para visualizar os níveis da variável ORIGEM.PAIS do dataframe `dadosRT`.
26. Reorganize os níveis da variável ORIGEM.PAIS na ordem de maior para menor proximidade da cidade de São Paulo, com o nível "mista" (= pai e mãe nascidos em locais distintos) por último. Não é necessário guardar o resultado.

Lição 4: Variáveis Nominais: Tabelas

Nesta lição, vamos aprender a verificar a distribuição de dados de variáveis nominais (também chamadas de categóricas ou qualitativas), por meio de tabelas.

Uma vez com uma planilha de dados em mãos, o primeiro passo de uma análise estatística é tabular e visualizar os dados. Isso mesmo! Não é bom sair correndo para aplicar testes estatísticos sem ter uma ideia de como se distribuem seus dados. A feitura de gráficos é objeto da próxima lição.

Normalmente se pensa que fazer tabelas, figuras e gráficos é um dos últimos passos da pesquisa, quando se está prestes a fazer uma comunicação num congresso, apresentar um pôster ou escrever um artigo... mas adquira o bom hábito de produzi-los assim que tiver os dados organizados em uma planilha. A Estatística Descritiva é o primeiro passo de qualquer boa análise.

Também já é bom colocar em prática os bons hábitos de gerenciamento do fluxo de trabalho. Primeiro, carregue o pacote tidyverse, que será necessário para as funções que vamos executar.

```
library(tidyverse)
```

Nesta lição, vamos trabalhar com um arquivo de dados famoso – do estudo de Labov nas lojas de departamento de Nova Iorque. (Se você não conhece este trabalho, leia o capítulo 2 de Padrões Sociolinguísticos, de William Labov.) Defina como diretório de trabalho aquele que contém o arquivo LabovDS.csv.

```
setwd("~/Dropbox/_R/swirl/Introducao_a_Estatistica_para_Linguistas/data")
```

N.B.: Defina como diretório de trabalho aquele que contém o arquivo LabovDS.csv.

Agora carregue os dados da planilha LabovDS.csv em um dataframe chamado `ds` (para department store). Use a função `read_csv()`, definindo todas as colunas como factor. Não vai ser necessário redefinir os níveis de nenhuma variável.

```
ds <- read_csv("LabovDS.csv",
               col_types = cols(.default = col_factor()
                                )
               )
```

Sempre que carregar um arquivo de dados, é importante verificar se o arquivo foi lido corretamente. Aplique a função `str()` ao dataframe `ds` para fazer essa checagem e também para ter um primeiro contato com a estrutura dos dados.

```
str(ds)
```

```
## spec_tbl_df [759 × 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ r       : Factor w/ 3 levels "r1","r0","d": 1 1 1 1 1 1 1 1 1 1
...
##  $ store   : Factor w/ 3 levels "Saks","Macys",..: 1 1 1 1 1 1 1 1
1 1 ...
##  $ emphasis: Factor w/ 2 levels "casual","emphatic": 1 1 1 1 1 1 1
1 1 1 ...
##  $ word    : Factor w/ 2 levels "fouRth","flooR": 1 1 1 1 1 1 1 1 1
1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   .default = col_factor(),
##   ..   r = col_factor(levels = NULL, ordered = FALSE, include_na =
FALSE),
##   ..   store = col_factor(levels = NULL, ordered = FALSE, include_n
a = FALSE),
##   ..   emphasis = col_factor(levels = NULL, ordered = FALSE, includ
e_na = FALSE),
##   ..   word = col_factor(levels = NULL, ordered = FALSE, include_na
= FALSE)
##   .. )
##  - attr(*, "problems")=<externalptr>
```

O R nos informa que `ds` é um dataframe com 759 ocorrências de 4 variáveis: r, store, emphasis e word. Nesse estudo, Labov analisou a pronúncia variável de /r/ pós-vocálico – que pode ser realizado (r1) ou apagado (r0) –, em 3 lojas de departamento de Nova Iorque (Saks, Macy's e S. Klein), em dois contextos linguísticos – meio (fourth) e fim de palavra (floor) –, e em dois graus de ênfase (casual ou enfático). Os dados de /r/ codificados como "d" se referem a realizações duvidosas, em que ele não conseguiu determinar se o /r/ havia sido realizado ou apagado.

Seu método foi extremamente engenhoso: Labov se aproximava de um funcionário da loja e perguntava onde ficava um determinado item (p.ex., os sapatos femininos), cuja resposta ele já sabia ser "fourth floor" (no quarto andar). Em seguida,

ele fingia não ter entendido o que a pessoa havia dito e pedia para que repetisse, ao que se esperava que o falante respondesse de modo mais claro ou enfático. Assim ele obtinha 4 ocorrências de /r/ pós-vocálico por falante, anotava as respostas em seu caderninho e seguia para fazer o mesmo com outro funcionário. Labov colheu dados de 68 pessoas na Saks, 125 na Macy's e 71 na S. Klein.

Um pesquisador honesto presta conta de todos os seus dados! Se Labov colheu 4 dados de 68 + 125 + 71 pessoas, qual deve ser o total de dados? Faça esta conta agora.

```
(68 + 125 + 71) * 4
```

```
## [1] 1056
```

Exato! Deveria haver 1056 dados, mas há 759 no total. Faltam, portanto, 297 dados. Labov explica em seu trabalho o que ocorreu: em alguns casos, principalmente na repetição, o falante não produziu "fourth floor", mas simplesmente "fourth". Esse tipo de coisa acontece. Pesquisas dificilmente tendem a ocorrer exatamente como planejado ou previsto, e os percalços e a solução encontrada também devem ser reportados em suas publicações.

Estamos prontos, então, para começar a analisar esses dados. Um primeiro interesse é verificar quantos dados há para cada variante de cada variável. No tidyverse, isso é feito com a função `count()`. No *script*, temos agora um novo símbolo, `%>%`, que é chamado de pipe. Esse símbolo, que será bastante usado junto a funções do pacote tidyverse, pode ser glosado como: "pegue o resultado da operação anterior e execute o que vem em seguida". Neste caso, estamos dizendo ao R para pegar o dataframe `ds` e, com ele, fazer a contagem dos dados da variável r (quantos r1-realizações, r0-apagamentos e d-dados duvidosos). Execute-o agora para ver o resultado.

```
ds %>%
  count(r)
```

```
## # A tibble: 3 × 2
##   r         n
##   <fct> <int>
## 1 r1      231
## 2 r0      499
## 3 d        29
```

O R nos fornece o *output* no formato de um dataframe visualizado como tibble, informando que houve 231 ocorrências de r1, 499 de r0 e 29 de d. Esses números são chamados de *frequências*. Cabe aqui chamar a atenção para o fato de que, no uso comum, normalmente se emprega o termo "frequência" para se referir a proporções, que são coisas distintas (e vamos ver logo adiante). O uso técnico e correto do termo "frequência" é este: quantas vezes alguma variante ocorreu.

A cada tabela, figura, teste estatístico..., cabe agora ao pesquisador verificar se os resultados estão de acordo com as expectativas, com a teoria, com os modelos testados etc. Pare para pensar um pouco o que nos diz a tabela de distribuição de dados de /r/.

Evidentemente, não há uma única resposta certa para isso, mas minimamente vale a pena notar duas coisas: (i) o número de ocorrências de dados duvidosos é bastante pequeno, o que dá maior confiança a quaisquer outros resultados a que se vai chegar – imagine se mais da metade dos dados fossem duvidosos!; (ii) o número de ocorrências de apagamento, no inglês novaiorquino na década de 1960 para esse tipo de falante (funcionários de lojas de departamento), era relativamente bem mais frequente do que sua realização – mais do que o dobro!

Visto que há um número pequeno de ocorrências de d, e que o foco do estudo é sobre a realização vs. apagamento de /r/, podemos descartar os dados duvidosos e criar um novo subconjunto de dados. Vimos como fazer isso na Lição 2 - Manipulação de Vetores e Dataframes. Você se lembra como? Qual função vamos usar?

- `c()`
- `file.create()`
- `filter()`
- `getwd()`
- `read_csv()`

Vamos criar um novo conjunto de dados, chamado `ds2`, que contém apenas os dados de "r0" e "r1". Vamos aproveitar e usar o pipe `%>%`, que acabamos de aprender. Digite `ds2 <- ds %>% filter(r != "d")`.

```
ds2 <- ds %>%
  filter(r != "d")
```

Cheque se o novo dataframe foi criado corretamente. Primeiro, em Environment, veja se aparece `ds2`, que deve conter 29 dados a menos do que `ds`. Em seguida, aplique a função `str()` a `ds2`. (Sim, isso é algo que deve ser feito sempre!)

```
str(ds2)

## spec_tbl_df [730 × 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ r       : Factor w/ 3 levels "r1","r0","d": 1 1 1 1 1 1 1 1 1 1
...
##  $ store   : Factor w/ 3 levels "Saks","Macys",..: 1 1 1 1 1 1 1 1
1 1 ...
##  $ emphasis: Factor w/ 2 levels "casual","emphatic": 1 1 1 1 1 1 1
1 1 1 ...
##  $ word    : Factor w/ 2 levels "fouRth","flooR": 1 1 1 1 1 1 1 1 1
1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   .default = col_factor(),
##   ..   r = col_factor(levels = NULL, ordered = FALSE, include_na =
FALSE),
##   ..   store = col_factor(levels = NULL, ordered = FALSE, include_n
a = FALSE),
##   ..   emphasis = col_factor(levels = NULL, ordered = FALSE, includ
e_na = FALSE),
##   ..   word = col_factor(levels = NULL, ordered = FALSE, include_na
= FALSE)
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Opa! Tem um problema! Apesar de excluirmos os dados duvidosos da planilha, "d" continua sendo um nível dessa variável fatorial. Isso acontece porque é possível haver um nível com zero dados. Mas tem um modo fácil de excluí-lo: volte à linha de comando em que excluímos os dados duvidosos, inclua novo `%>%` após o comando com `filter()`, e digite na linha seguinte `droplevels()`. Isso fará com que todos os níveis sem dados sejam excluídos.

```
ds2 <- ds %>%
  filter(r != "d") %>%
  droplevels()
```

Vamos seguir agora com o dataframe `ds2`, ok? Reveja a distribuição de dados de r no novo conjunto de dados, por meio da função `count()`.

```
ds2 %>%
  count(r)

## # A tibble: 2 × 2
##   r         n
##   <fct> <int>
## 1 r1      231
## 2 r0      499
```

Para calcular proporções no tidyverse, é necessário calcular primeiramente as frequências, como acabamos de fazer. Às frequências, vamos aplicar a função `mutate()` que, genericamente, faz transformações nos dados. Nesse caso, usamos a função `prop.table()` sobre a coluna n para computar as proporções. Identifique esses passos na linha de comando no *script*, que já está pronta, e rode-a em seguida. Observe o uso do pipe, que toma o resultado de cada operação para executar a seguinte.

```
ds2 %>%
  count(r) %>%
  mutate(prop = prop.table(n))

## # A tibble: 2 × 3
##   r         n  prop
##   <fct> <int> <dbl>
## 1 r1      231 0.316
## 2 r0      499 0.684
```

A proporção de r1 foi de 32% e de r0 foi de 68%. Vamos agora calcular a frequência dos dados da variável store.

```
ds2 %>%
  count(store)

## # A tibble: 3 × 2
##   store     n
##   <fct> <int>
## 1 Saks    178
## 2 Macys   336
## 3 Klein   216
```

Vejamos os resultados. Para store, o R informa que houve, no total, 178 ocorrências de /r/ na Saks, 336 na Macy's e 216 na S. Klein. O que isso quer dizer? Na verdade, isso nos diz muito pouco, para além de uma pista de onde estão os 297 dados não produzidos. Visto que mais dados foram coletados na Macy's (125 pessoas), era mesmo de se esperar que houvesse mais dados dessa loja...

As distribuições de dados das variáveis store, emphasis e word não fazem sentido sem levar em conta aquilo que é o foco do estudo: a pronúncia de /r/ pós-vocálico. A pronúncia de /r/ é a *variável dependente* (VD), e todas as demais são *independentes* (VIs). O que nos interessa é conhecer a distribuição de dados das VIs em relação à VD.

Para ver a distribuição dos dados entre duas variáveis, também usamos a função `count()`, com a adição de mais um argumento. Compute então as frequências de dados por store e r, nessa ordem.

```
ds2 %>%
  count(store, r)

## # A tibble: 6 × 3
##   store r         n
##   <fct> <fct> <int>
## 1 Saks  r1       85
## 2 Saks  r0       93
## 3 Macys r1      125
## 4 Macys r0      211
## 5 Klein r1       21
## 6 Klein r0      195
```

Em qual das lojas houve maior ocorrência de apagamentos de /r/? Evidentemente, as simples frequências (93, 211 e 195) podem ser enganadoras, uma vez que os totais para cada loja são diferentes. Queremos, então, visualizar a distribuição em proporções, que indicam o quanto cada frequência representa do total.

Para computar as proporções, o comando será semelhante ao que fizemos acima para a variável r. No entanto, como agora temos duas variáveis, é necessário informar ao R com base em qual variável será calculada essa medida estatística, por meio da função `group_by()`. Nela, estamos dizendo ao R para calcular as proporções por store. Na linha de comando no *script*, essa função foi incluída após `count()` e antes de `mutate()`. Após se certificar de que entendeu essa linha de comando, rode-a com CTRL + ENTER.

```
ds2 %>%
  count(store, r) %>%
  group_by(store) %>%
  mutate(prop = prop.table(n))

## # A tibble: 6 × 4
## # Groups:   store [3]
##   store r         n   prop
##   <fct> <fct> <int>  <dbl>
```

```
## 1 Saks  r1       85 0.478
## 2 Saks  r0       93 0.522
## 3 Macys r1      125 0.372
## 4 Macys r0      211 0.628
## 5 Klein r1       21 0.0972
## 6 Klein r0      195 0.903
```

Na última linha de comando, agrupamos os dados por store, de modo que, em cada loja, os dados de r1 e r0 somam 100%. Para contrastar, rode a próxima linha de comando, que agrupa os dados pela variável r.

```
# um comando que o R roda, mas não faz sentido

ds2 %>%
  count(r, store) %>%
  group_by(r) %>%
  mutate(prop = prop.table(n))

## # A tibble: 6 × 4
## # Groups:   r [2]
##   r     store     n   prop
##   <fct> <fct> <int>  <dbl>
## 1 r1    Saks     85 0.368
## 2 r1    Macys   125 0.541
## 3 r1    Klein    21 0.0909
## 4 r0    Saks     93 0.186
## 5 r0    Macys   211 0.423
## 6 r0    Klein   195 0.391
```

Desta vez, são os dados das lojas – Saks, Macy's, S. Klein – que somam 100% ou para r1 ou para r0. Mas essa última tabela não faz sentido! Ao tabular os dados de cada loja quanto ao uso da variável r, queremos saber se há diferenças entre os locais; cada uma delas, portanto, deve ser tomada com uma totalidade, dentro da qual se analisa a distribuição dos dados. Daí sim eles podem ser comparados. Do cômputo correto, depreendemos que r0 é mais frequente que r1 em todas as lojas, e que a proporção de apagamento na S. Klein (90,3%) é muito maior do que na Macy's (62,8%) e na Saks (52,2%).

Essa última operação foi feita a fim de chamar a atenção para a devida escolha da variável pela qual os dados devem ser agrupados. O R não sabe o que é store e r, e não tem como decidir por você! Veja que a escolha da variável incorreta levará a medidas estatísticas completamente diferentes!

Visualize agora a distribuição dos dados de frequência e proporções de emphasis pela VD.

```
ds2 %>%
  count(emphasis, r) %>%
  group_by(emphasis) %>%
  mutate(prop = prop.table(n))
```

```
## # A tibble: 4 × 4
## # Groups:   emphasis [2]
##   emphasis r         n  prop
##   <fct>    <fct> <int> <dbl>
## 1 casual   r1      137 0.298
## 2 casual   r0      322 0.702
## 3 emphatic r1       94 0.347
## 4 emphatic r0      177 0.653
```

Da tabela acima, em qual estilo de fala ocorreu mais apagamento (r0)?

- casual
- enfático

Compute a frequência e a proporção dos dados de word pela VD.

```
ds2 %>%
  count(word, r) %>%
  group_by(word) %>%
  mutate(prop = prop.table(n))
```

```
## # A tibble: 4 × 4
## # Groups:   word [2]
##   word   r         n  prop
##   <fct>  <fct> <int> <dbl>
## 1 fouRth r1       88 0.230
## 2 fouRth r0      295 0.770
## 3 flooR  r1      143 0.412
## 4 flooR  r0      204 0.588
```

Da tabela acima, vemos que a proporção de r0 na palavra “fourth” é maior do que na palavra “floor”. Também já vimos que houve relativamente mais apagamento no estilo casual (vs. enfático) e na S. Klein (vs. Macy’s e Saks). Em qual das variáveis a diferença entre proporções parece ser maior?

- emphasis
- store
- word

Na questão acima, usei o termo "parece" pois, para determinar o grau dessas diferenças, precisaremos de testes estatísticos que serão objeto de lições futuras. Essas distribuições, no entanto, já são um ótimo começo para saber o que está acontecendo nos dados.

Ao usar as funções do tidyverse, os resultados são fornecidos no formato de dataframes. Todas as distribuições de dados que vimos acima são nesse formato. Vamos ver agora outro modo de fazer a mesma coisa: visualizar frequências e distribuições de dados, mas dessa vez por meio de tabelas. Ao final, vamos comparar os dois modos – ambos serão úteis nas próximas lições.

A instalação base do R tem uma função chamada `table()`, que serve, justamente, para tabular dados. Vamos usá-la junto à função `with()`, para que o nome do dataframe não tenha que ser digitado muitas vezes. Rode a linha de comando neste ponto do *script*.

```
freq.r <- with(ds2, table(r))
```

Visualize a tabela digitando `freq.r`.

```
freq.r
```

```
## r
##  r1  r0
## 231 499
```

Para fazer a tabela de proporções, como vimos mais acima, precisamos da tabela de frequências. Aplique então a função `prop.table()` à tabela `freq.r`, e guarde o resultado em um objeto chamado `prop.r`.

```
prop.r <- prop.table(freq.r)
```

E visualize `prop.r`.

```
prop.r
```

```
## r
##        r1        r0
## 0.3164384 0.6835616
```

Crie um objeto chamado `freq.store` com a distribuição de dados de loja (store) pela VD (r). Para tanto, inclua as duas variáveis como argumentos de `table()`, deixando a VD ao final.

```
freq.store <- with(ds2, table(store, r))
```

Visualize `freq.store`.

```
freq.store
```

```
##        r
## store    r1  r0
##   Saks   85  93
##   Macys 125 211
##   Klein  21 195
```

Ao colocar a VD como segundo argumento de `table()`, ela aparece nas colunas (e a VI nas linhas), que é o modo mais comum de apresentar distribuição de dados de VIs por uma VD. Embora, em princípio, não faça diferença de qual modo você vai visualizar a distribuição dos dados, recomendo que você se acostume com essa convenção.

Assim como no tidyverse, para fazer a tabela de proporções quando se tabulam duas variáveis, podemos especificar para qual das variáveis as variantes somarão 100% – a da linha, a da coluna, ou ainda a tabela toda. Na função `prop.table()`, isso é informado como um novo argumento, pela convenção 1 = linha, 2 = coluna, e nada (= default) = tabela. Pela tabela de frequências acima, qual opção faz mais sentido aplicar?

- 1 = linha
- 2 = coluna
- nada = default

Crie então uma tabela de proporções chamada `prop.store`, usando a tabela `freq.store` e 1 como segundo argumento de `prop.table()`.

```
prop.store <- prop.table(freq.store, 1)
```

E visualize `prop.store`.

```
prop.store
```

```
##        r
## store           r1         r0
##   Saks  0.47752809 0.52247191
##   Macys 0.37202381 0.62797619
##   Klein 0.09722222 0.90277778
```

Note que, aqui também, a escolha equivocada da variável pela qual as proporções serão computadas pode levar a medições completamente diferentes. Neste caso, é

necessário saber qual variável ocupa a linha e qual ocupa a coluna para bem escolher 1 ou 2 em `prop.table()`.

Tabelas são semelhantes aos dataframes, pois também têm linhas e colunas. Assim, é possível visualizar elementos específicos de uma tabela por meio dos colchetes. Verifique qual é o elemento na 2ª linha da 1ª coluna de `freq.store`.

```
freq.store[2, 1]
```

```
## [1] 125
```

Também é possível visualizar os valores totais das linhas e das colunas de uma tabela aplicando a função `addmargins()`. Digite `addmargins(freq.store)` para ver o resultado.

```
addmargins(freq.store)
```

```
##        r
## store    r1  r0 Sum
##   Saks   85  93 178
##   Macys 125 211 336
##   Klein  21 195 216
##   Sum   231 499 730
```

Você deve ter notado que a visualização de frequências e proporções por meio de tabelas é diferente daquela no formato dataframe. Para além dessa diferença estética, há consequências práticas de escolher computar essas medidas por meio das funções do tidyverse, como fizemos primeiro, ou por meio das funções da instalação base do R.

A primeira delas é que, no tidyverse, é possível computar frequências e proporções (além de outras medidas) por meio de uma única linha de comando, usando `%>%`, e, na instalação base, isso precisa ser feito passo a passo. Para um usuário iniciante em R, a segunda opção pode parecer mais fácil, uma vez que é possível ter maior controle e visualizar cada etapa do que se está fazendo. Entretanto, como já vimos em lições anteriores, uma linha de comando não precisa ser digitada por completo logo de primeira. Você também pode digitar parte da linha de comando (desde que já seja inteligível ao R), ver o resultado, e completá-la posteriormente.

Uma segunda diferença é que as funções do tidyverse geraram o *output* no formato de dataframe, e as funções da instalação base geraram tabelas. Essa diferença é

relevante a depender do que se pretende fazer posteriormente com esses dados. Por exemplo, para fazer um gráfico de barras no tidyverse (com o pacote ggplot2), você vai precisar de um dataframe (ver Lições 5 e 7); para fazer um teste de qui-quadrado, você vai precisar de uma tabela (ver Lição 9). É importante, então, conhecer os dois modos de computar frequência e proporções!

Nesta lição, vimos, por exemplo, que houve proporcionalmente mais ocorrências de apagamento de /r/ na S. Klein, seguida da Macy's, e por último na Saks. (Leia o estudo de Labov 1972 para ver sua interpretação desses resultados!). A diferença entre as lojas já fica bastante clara por meio dos números, mas isso pode ficar ainda mais claro se mostrado por uma figura – um objeto gráfico –, em vez de uma tabela, que é textual. A feitura de gráficos é o assunto da próxima lição.

Para saber mais

Recomendo a leitura do capítulo 4 de Dalgaard (2008) sobre Estatística Descritiva.

Exercícios

Para esta lição, vamos usar o arquivo de dados LabovDS.csv, com a exclusão de dados duvidosos "d".

1. Defina como diretório de trabalho aquele que, em seu computador, contém a planilha LabovDS.csv.
2. Carregue o pacote tidyverse.
3. Carregue os dados da planilha em um dataframe chamado `ds`, especificando que todas as colunas são do tipo factor.
4. Cheque o dataframe `ds` por meio da função `str()`.
5. Exclua os dados 'd' por meio da função `filter()`. Guarde o novo dataframe em um objeto chamado `ds2` e exclua o nível 'd' do dataframe.
6. Visualize os níveis da variável store do dataframe `ds2`.

7. Reorganize os níveis da variável store em ordem inversa: Klein, Macys, Saks. Guarde o resultado na coluna correspondente do dataframe.
8. A partir de `ds2`, faça uma tabela de frequências da variável emphasis pela variável dependente r com as funções da instalação base do R. *Não* se preocupe em guardar o resultado em um objeto.
9. A partir de `ds2`, faça a tabela de frequências da variável word pela variável dependente r com as funções da instalação base do R. *Não* se preocupe em guardar o resultado em um objeto.
10. A partir de `ds2`, faça a tabela de frequências da variável store pela variável dependente r com as funções da instalação base do R. *Não* se preocupe em guardar o resultado em um objeto.
11. A partir de `ds2`, faça a tabela de frequências da variável store pela variável dependente r com as funções da instalação base do R, e adicione os totais das linhas e das colunas. Faça isso em uma única linha de comando!
12. A partir de `ds2`, faça a tabela de proporções por linha da variável emphasis pela variável dependente r. Faça isso em uma única linha de comando.
13. A partir de `ds2`, faça a tabela de proporções por linha da variável word pela variável dependente r. Faça isso em uma única linha de comando.
14. A partir de `ds2`, faça a tabela de proporções por linha da variável store pela variável dependente r. Faça isso em uma única linha de comando.
15. A partir de `ds2`, crie um dataframe, chamado `freq.prop.store`, com as frequências e proporções da variável store por r, usando as funções do tidyverse. Nomeie a coluna com proporções como proporcao.
16. Visualize `freq.prop.store`.
17. Acesse a coluna proporcao do dataframe `freq.prop.store`.
18. A partir de `freq.prop.store`, acesse apenas as frequências e as proporções, e somente da loja Saks. Use os colchetes [ ].
19. A partir de `freq.prop.store`, acesse as frequências e as proporções somente da loja Saks, usando o pipe `%>%` e as funções `filter()` e `select()`.

20. A partir de `freq.prop.store`, crie um novo dataframe chamado `freq.prop.store2`, que contém as proporções abaixo de 60%.
21. A partir de `freq.prop.store`, crie um novo dataframe chamado `freq.prop.store3`, que contém as frequências acima de 90.
22. Compute as frequências e as proporções de r1 e r0 por palavra (word) apenas da loja Saks, usando as funções do tidyverse. Para tanto, use o pipe `%>%` e as funções `filter()`, `count()`, `group_by()` e `mutate()`. Nomeie a coluna de proporções como prop.
23. Compute as frequências e as proporções de r1 e r0 por ênfase (emphasis), usando as funções do tidyverse. Nomeie a coluna de proporções como prop. As proporções devem aparecer como valores entre 1 e 100%.
24. Veja o dataframe que você acabou de criar com as frequências e proporções da VD r pela VI emphasis. Que tipo de variável é n?
    a. double
    b. factor
    c. int
25. Veja o dataframe que você acabou de criar com as frequências e proporções da VD r pela VI emphasis. Que tipo de variável é prop?
    a. double
    b. factor
    c. int
26. Veja o dataframe que você acabou de criar com as frequências e proporções da VD r pela VI emphasis. Que tipo de variável é emphasis?
    a. double
    b. factor
    c. int

Lição 5: Variáveis Nominais: Gráficos

Na última lição, computamos frequências e proporções a partir dos dados de Labov em seu estudo nas lojas de departamento em Nova Iorque. Para recarregar os dataframes `ds2` e `df.store` nesta sessão, caso já não o tenha em Environment, rode as linhas de comando a seguir.

```
# Definir diretório de trabalho
#setwd()

library(tidyverse)

# Importar planilha

ds <- read_csv("LabovDS.csv",
               col_types = cols(.default = col_factor())
               )

# Excluir dados duvidosos `d`

ds2 <- ds %>%
  filter(r != "d") %>%
  droplevels()

# Tabular frequências e proporções da variável store pela variável r

df.store <- ds2 %>%
  count(store, r) %>%
  group_by(store) %>%
  mutate(prop = prop.table(n))
```

No tidyverse, gráficos são feitos com o pacote ggplot2. Carregue então o pacote tidyverse para deixá-lo disponível nesta sessão.

```
library(tidyverse)
```

Carregue também o pacote RColorBrewer, que oferece várias paletas de cores.

```
library(RColorBrewer)
```

O ggplot2 é um pacote dedicado à visualização de dados, elaborado por Hadley Wickham. Trata-se de uma implementação da Gramática de Gráficos (daí os dois "g" do nome do pacote) de Leland Wilkinson, uma abordagem da visualização de dados que a entende como uma gramática, com estruturas e regras. Um gráfico normalmente é

composto de atributos estéticos (cores, formas, tamanhos) de objetos geométricos (linhas, pontos, barras etc.) em um sistema de coordenadas. Uma gramática de gráficos nos auxilia a dispor esses elementos de modo significativo. Todos os gráficos com ggplot2 são compostos de um conjunto de dados, dispostos em um dataframe, e de um mapeamento, que descreve como as variáveis do dataframe são mapeados a atributos estéticos.

Nesta lição, vamos trabalhar sobre o dataframe `df.store`. Inspecione-o agora.

```
df.store
```

```
## # A tibble: 6 × 4
## # Groups:   store [3]
##   store r         n   prop
##   <fct> <fct> <int>  <dbl>
## 1 Saks  r1       85 0.478
## 2 Saks  r0       93 0.522
## 3 Macys r1      125 0.372
## 4 Macys r0      211 0.628
## 5 Klein r1       21 0.0972
## 6 Klein r0      195 0.903
```

Um tipo de gráfico básico e apropriado para mostrar distribuições de variáveis *nominais* – como é o caso da VD /r/ pós-vocálico – é o gráfico de barras. No *script*, temos a estrutura de um gráfico de barras. No momento, há apenas três linhas sem #, que serão lidas pelo R ao rodar esse comando – aquelas com # serão ignoradas. Rode-a agora, para ver o resultado (Figura 5.1), que será comentado na sequência. Se, a qualquer momento, aparecer a mensagem de erro "Error in plot.new() : figure margins too large", aumente a janela para Plots.

```
ggplot(df.store, aes(x = store, y = prop, fill = r)) +
  geom_bar(stat = "identity", color = "black") +
#  ggtitle("") +
#  labs(x = "", y = "", fill = "") +
#  scale_x_discrete() +
#  scale_fill_brewer(palette = "", labels = ) +
  theme_bw()
```

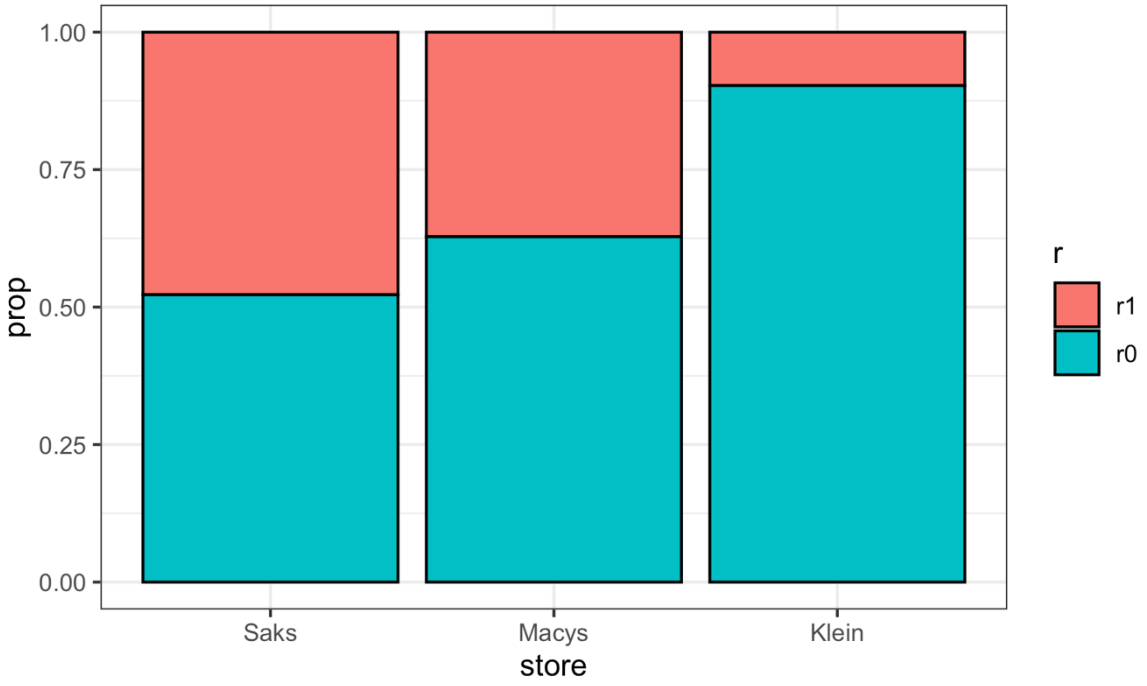

Figura 5.1: Gráfico de barras simples com ggplot. Fonte: própria.

A linha de comando definiu o conjunto de dados (`df.store`) e os parâmetros estéticos (aes) com a função `ggplot()`, declarou que o gráfico é de barras com `geom_bar()`, e estabeleceu um tema com `theme_bw()`. Os dois primeiros são o mínimo necessário para plotar um gráfico no ggplot2. Vejamos os argumentos dessas funções com mais detalhes.

Na função `ggplot()`, o primeiro argumento é o conjunto de dados, que sempre deve ter o formato de um dataframe. O segundo argumento, `aes()`, define a estética do gráfico. Aqui, especificamos que a variável store deve ocupar o eixo x, que a variável prop deve ocupar o eixo y, e que as barras (definidas na segunda linha) devem ser preenchidas separando-se as variantes da variável r.

No final da primeira linha há um sinal de mais +. Na Gramática de Gráficos, a visualização de dados é construída camada a camada, de modo semelhante a um pintor que trabalha sobre uma tela. Sempre que quiser acrescentar uma nova camada, adicione um + antes da próxima função.

A segunda linha declara qual forma gráfica os dados mapeados em `aes()` devem tomar. Para barras, usamos `geom_bar()`. Outras formas são `geom_line()`, `geom_histogram()`, `geom_point()`, `geom_area()`, `geom_polygon()` etc. Pode ter certeza de que, não importa qual gráfico você queira fazer, há uma geometria correspondente!

Dentro de `geom_bar()`, especificamos dois argumentos, stat e color. Para o primeiro, o *default* é "bin", que se refere à contagem de ocorrências da variável no dataframe. Neste caso, já temos a medida estatística que queremos plotar, que são as proporções (definidas em y na linha anterior), de modo que a opção "identity" informa que não é necessário fazer qualquer transformação nos valores do dataframe.

A cor especificada, "black", é a cor das bordas das barras. Depois, faça outros testes, colocando outras cores. Você pode estar se perguntando por que a cor não foi especificada dentro de `aes()`, por ser um parâmetro estético. Entretanto, note que esta é uma propriedade das barras, e por isso foi colocada dentro da função `geom_bar()`. Alternativamente, a função `aes()`, com todos os argumentos que especificamos, poderia ter sido colocada como argumento de `geom_bar()` também (ver comando no *script*).

A função `theme_bw()` define o fundo como preto e branco. Ao começar a digitar "them...", o RStudio mostra outras opções para você. O *default* é um fundo cinza (`theme_gray()`). Para visualizar outros temas, visite a página de referência do ggplot2: https://ggplot2.tidyverse.org/reference/ggtheme.html.

O R produziu um gráfico de barras empilhadas, mas digamos que você quisesse as barras lado a lado. Para isso, é necessário adicionar um argumento em `geom_bar()`: position = "dodge". Faça isso para ver o resultado (Figura 5.2).

```
ggplot(df.store, aes(x = store, y = prop, fill = r)) +
  geom_bar(stat = "identity", color = "black", position = "dodge") +
#  ggtitle("") +
#  labs(x = "", y = "", fill = "") +
#  scale_x_discrete() +
#  scale_fill_brewer(palette = "", labels = ) +
  theme_bw()
```

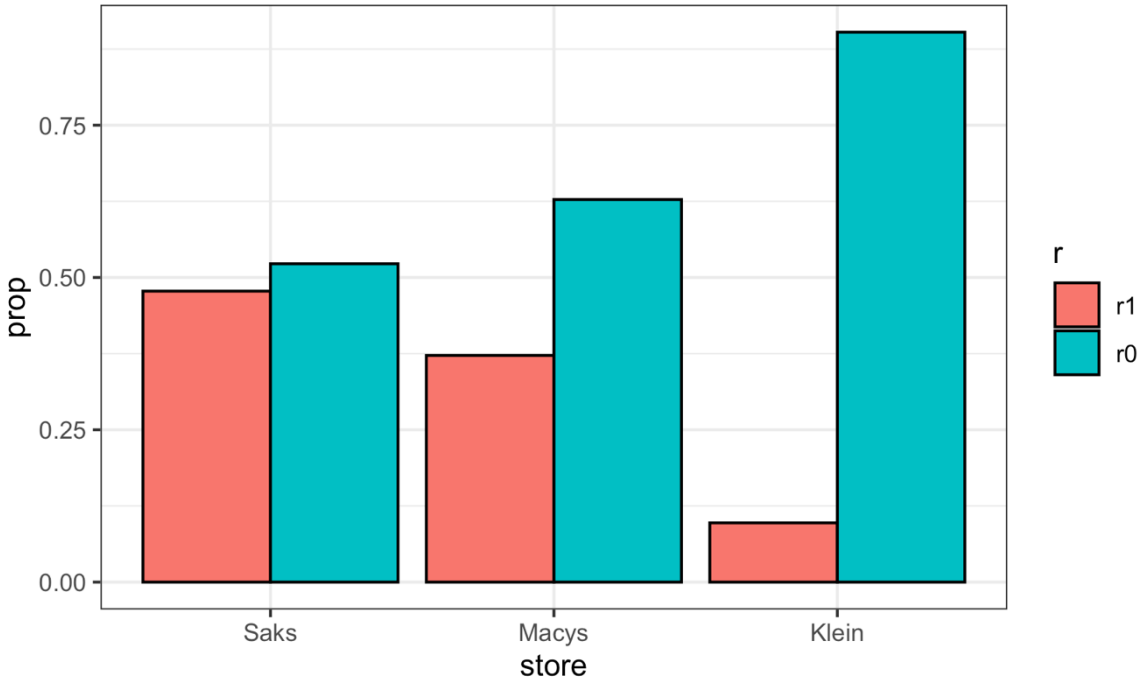

Figura 5.2: Gráfico de barras com position = "dodge". Fonte: própria.

Neste caso, apresentar as barras lado a lado é redundante. É evidente que, onde houve muito apagamento de r, também houve pouca realização. O gráfico que fizemos anteriormente é mais elegante, pois fornece a mesma informação sem que o leitor tenha que parar para entender quais barras somam 100%. Procure incorporar essa máxima na feitura de gráficos: menos é mais! Apague o argumento position que acabamos de incluir no comando (mas, caso você precise dele, já sabe como fazer!).

Você também pode querer que as proporções sejam apresentadas numa escala de 0 a 100, em vez de 0 a 1. Tem um jeito fácil de fazer isso sem ter que refazer o dataframe. Você consegue pensar numa maneira?

Se multiplicarmos os valores de prop por 100, as proporções serão apresentadas nessa escala! Volte então à linha de comando e, onde se lê y = prop, coloque y = prop * 100 (Figura 5.3).

```
ggplot(df.store, aes(x = store, y = prop * 100, fill = r)) +
  geom_bar(stat = "identity", color = "black") +
#  ggtitle("") +
#  labs(x = "", y = "", fill = "") +
#  scale_x_discrete() +
```

```
#  scale_fill_brewer(palette = "", labels = ) +
  theme_bw()
```

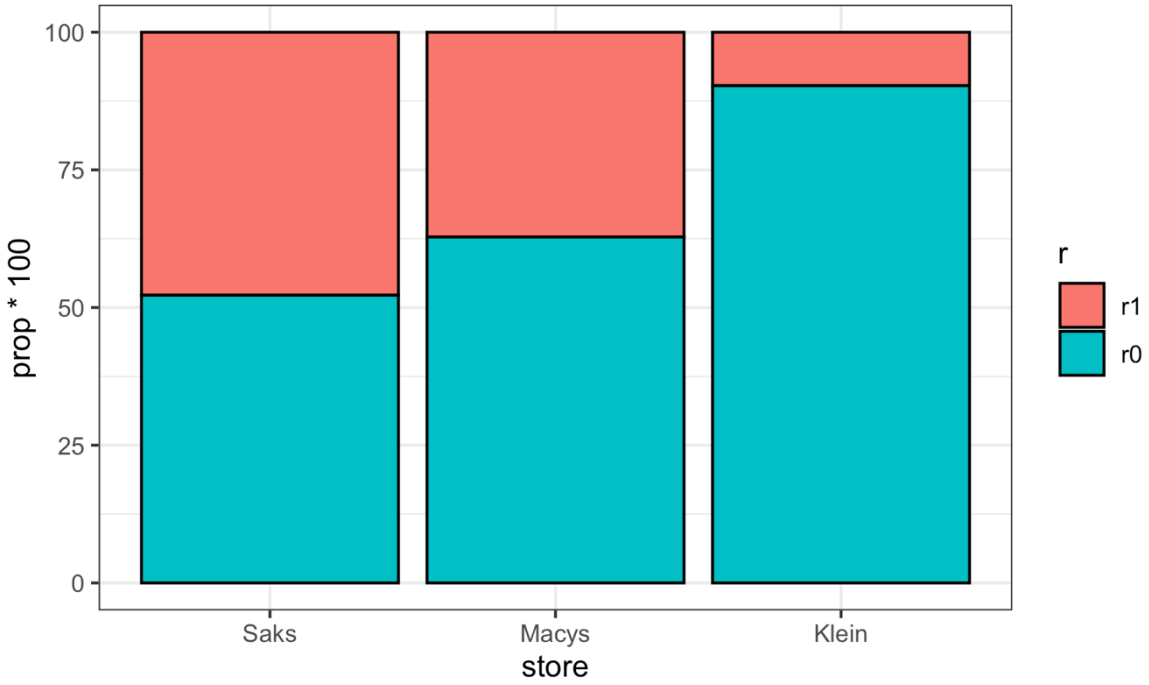

Figura 5.3: Gráfico de barras com proporção até 100. Fonte: própria.

Vamos ver agora os demais itens comentados com # na linha de comando, plotando-os um a um para ver o que fazem. A função `ggtitle()` permite inserir um título (e, se quiser, um subtítulo) na figura (ver Figura 5.4). Primeiro, apague o símbolo # no início dessa linha. Digite então, dentro das aspas, o seguinte título para este gráfico: "Proporção das variantes de /r/ pós-vocálico em três lojas de departamento em Nova Iorque (N = 730)". Atente-se à digitação!

```
ggplot(df.store, aes(x = store, y = prop * 100, fill = r)) +
  geom_bar(stat = "identity", color = "black") +
  ggtitle("Proporção das variantes de /r/ pós-vocálico em três lojas d
e departamento em Nova Iorque (N = 730)") +
#  labs(x = "", y = "", fill = "") +
#  scale_x_discrete() +
#  scale_fill_brewer(palette = "", labels = ) +
  theme_bw()
```

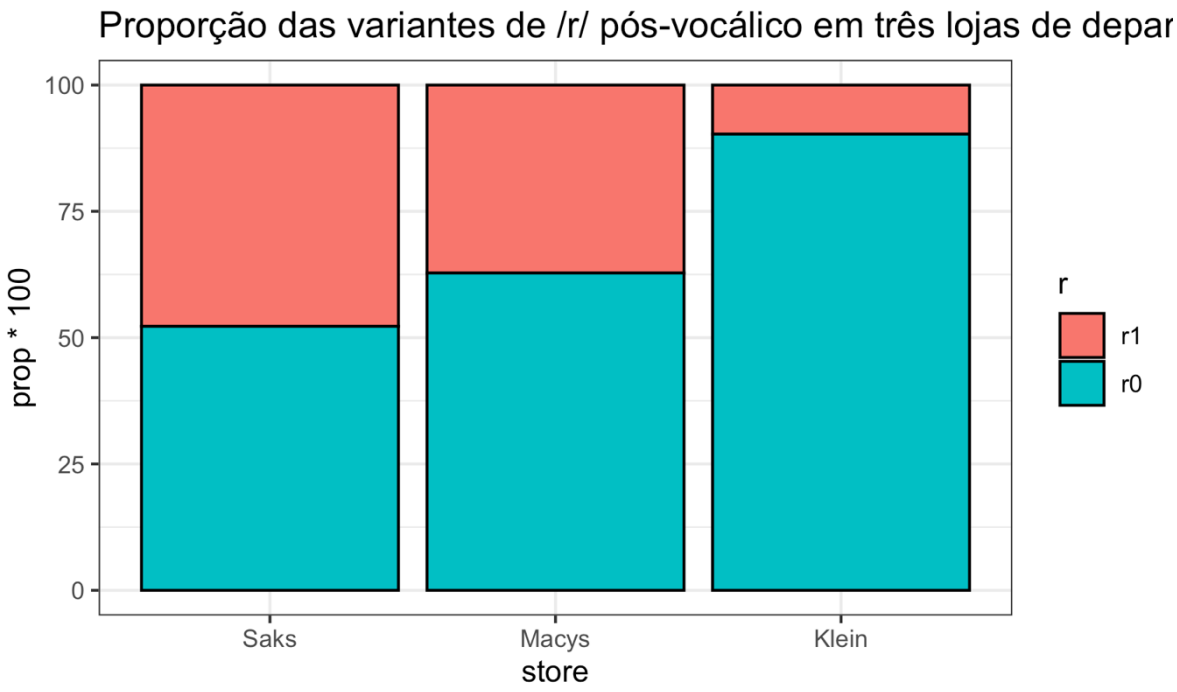

Figura 5.4: Gráfico de barras com título (1). Fonte: própria.

Hmm... o título não ficou bom! Inclua no título o símbolo de quebra de linha \n logo antes de "em três lojas" e veja se melhora! (Figura 5.5)

```
ggplot(df.store, aes(x = store, y = prop * 100, fill = r)) +
  geom_bar(stat = "identity", color = "black") +
  ggtitle("Proporção das variantes de /r/ pós-vocálico \nem três lojas
de departamento em Nova Iorque (N = 730)") +
#  labs(x = "", y = "", fill = "") +
#  scale_x_discrete() +
#  scale_fill_brewer(palette = "", labels = ) +
  theme_bw()
```

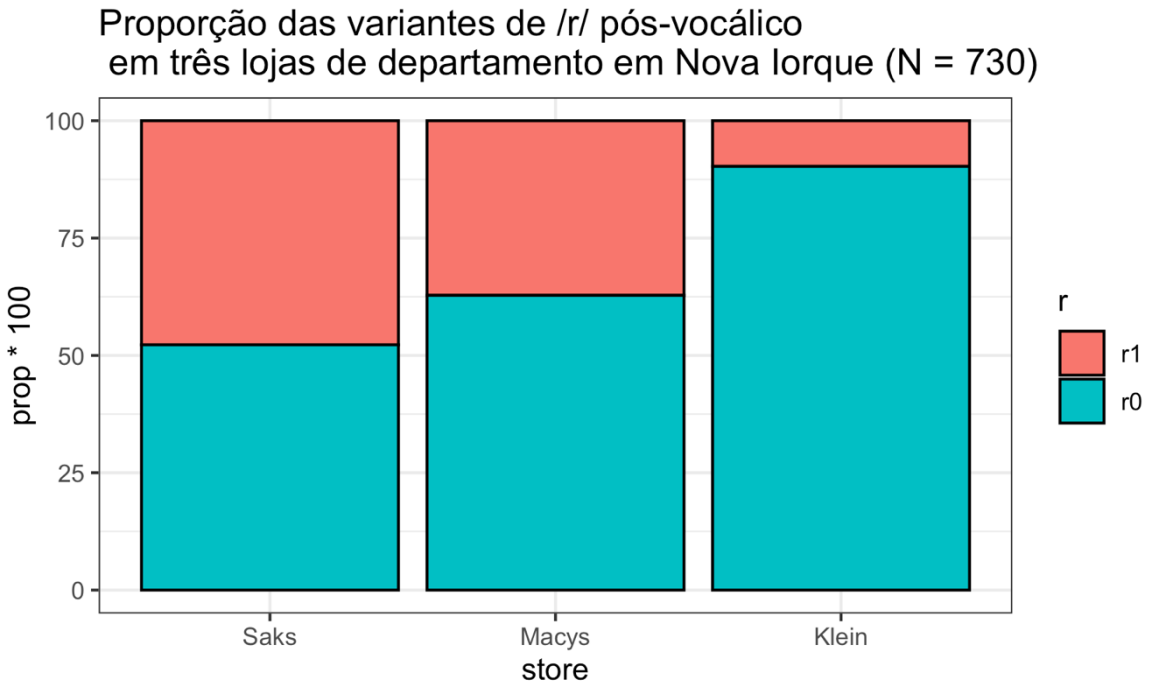

Figura 5.5: Gráfico de barras com título (2). Fonte: própria.

Vale aqui mencionar que títulos de gráficos – bem como os demais elementos – devem ser maximamente informativos, de modo que a figura faça sentido mesmo que o leitor não leia o texto. Se o título de um gráfico ficar demasiadamente longo, você pode optar por deixá-lo de lado e colocá-lo no próprio texto do artigo ou dos slides.

Que tal agora mudar o nome dos eixos x e y, que, no momento, tomam os nomes das variáveis do dataframe, store e prop * 100? Além disso, podemos mudar o nome da variável na legenda, r. Isso pode ser feito com a função `labs()` (lab representa "label" = etiqueta). Apague o # da próxima linha de comando e nomeie o eixo x como "Lojas", o eixo y como "Proporção", e o argumento fill como "Variantes de /r/" (Figura 5.6).

```
ggplot(df.store, aes(x = store, y = prop * 100, fill = r)) +
  geom_bar(stat = "identity", color = "black") +
  ggtitle("Proporção das variantes de /r/ pós-vocálico \nem três lojas
de departamento em Nova Iorque (N = 730)") +
  labs(x = "Lojas", y = "Proporção", fill = "Variantes de /r/") +
#  scale_x_discrete() +
#  scale_fill_brewer(palette = "", labels = ) +
  theme_bw()
```

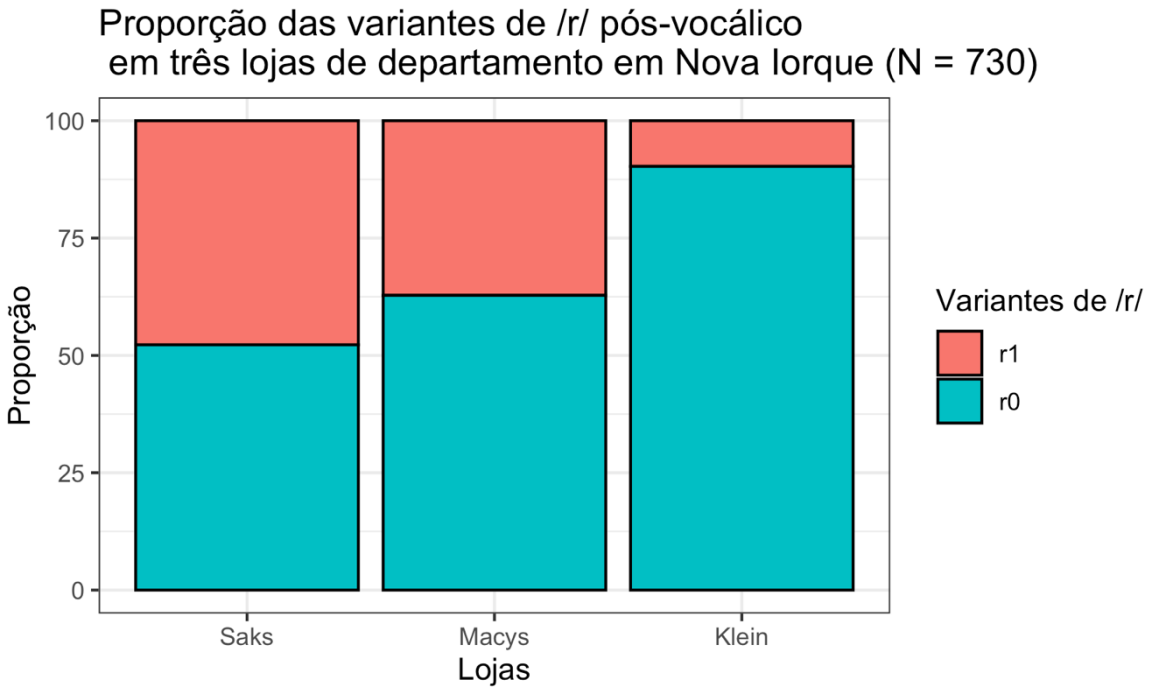

Figura 5.6: Gráfico de barras com nomes dos eixos customizados. Fonte: própria.

Está ficando bem legal! Vamos mexer mais: embora o nome das lojas na planilha seja "Saks", "Macys" e "Klein", seus nomes são, na verdade, "Saks", "Macy's" (com apóstrofe) e , "S. Klein" (com S.). Lógico que isso é detalhe, mas vamos ver como customizar o nome das variantes? Primeiro, queremos juntar os nomes corretos num vetor. Qual função usamos para isso?

- `c()`
- `head()`
- `length()`
- `setwd()`
- `str()`

Vamos usar a função `c()` para juntar esses nomes, que vão entrar numa nova função chamada `scale_x_discrete()`. Apague o # da linha com essa função e inclua o argumento labels, que será a junção de "Saks", "Macy's" e "S. Klein" (Figura 5.7). Atenção às aspas e às vírgulas! (**N.B.**: Se der erro, inclua \ antes do apóstrofe de Macy's.)

```
ggplot(df.store, aes(x = store, y = prop * 100, fill = r)) +
  geom_bar(stat = "identity", color = "black") +
  ggtitle("Proporção das variantes de /r/ pós-vocálico \nem três lojas
```

```
de departamento em Nova Iorque (N = 730)") +
  labs(x = "Lojas", y = "Proporção", fill = "Variantes de /r/") +
  scale_x_discrete(labels = c("Saks", "Macy's", "S. Klein")) +
#  scale_fill_brewer(palette = "", labels = ) +
  theme_bw()
```

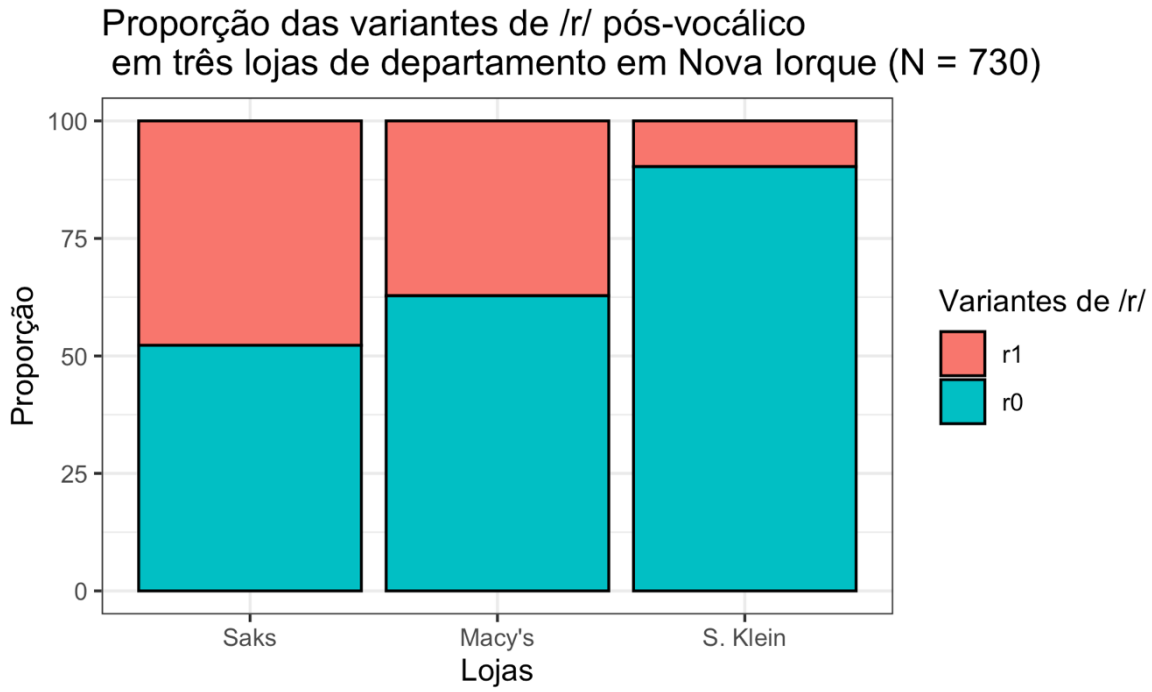

Figura 5.7: Gráfico de barras com rótulos das barras customizados. Fonte: própria.

E vamos finalmente mudar as cores dessa figura! A escolha de cores depende, entre outras coisas, de se um veículo aceita figuras coloridas (para um artigo, por exemplo) e, claro, de seu gosto pessoal. Entretanto, dada a chance de usá-las, é preferível usar figuras coloridas a figuras com tons de cinza – seus leitores interpretarão a informação mais rapidamente. Se precisar de mais de 6 cores numa figura, contudo, daí as informações começam a ficar confusas novamente.

Para mais dicas de como usar cores em suas figuras, uma boa referência é a página Introduction to Data Visualization: http://guides.library.duke.edu/topten. E o R dispõe de centenas de opções de cores, que você pode consultar nesse catálogo: http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf.

O ggplot2 oferece algumas paletas de cores pré-definidas (Figura 5.8). É recomendável utilizá-las, sobretudo em seus primeiros gráficos. Para ver um conjunto de paletas do RColorBrewer, digite `display.brewer.all()`.

```
display.brewer.all()
```

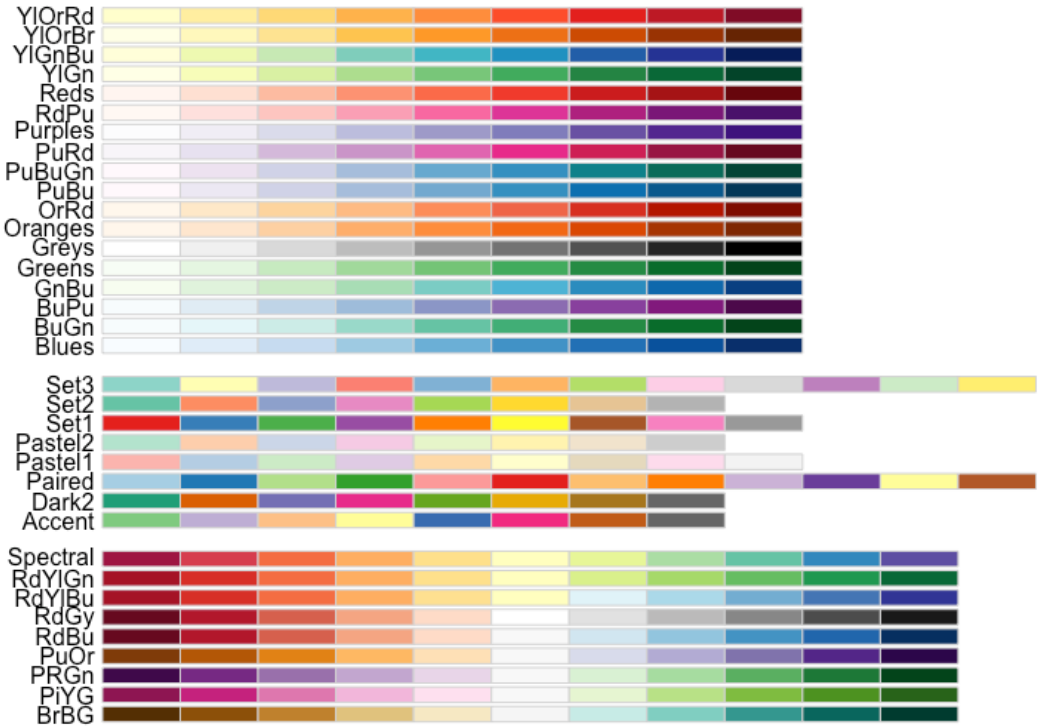

Figura 5.8: Resultado de `display.brewer.all()`. Fonte: própria.

Aqui, vamos usar a opção "Purples". Além disso, na função `scale_fill_brewer()`, também é possível definir outros rótulos para as variantes da variável de fill. Inclua também o argumento labels = c("realização", "apagamento"), para mudar as variantes de r para nomes mais inteligíveis (Figura 5.9).

```
ggplot(df.store, aes(x = store, y = prop * 100, fill = r)) +
  geom_bar(stat = "identity", color = "black") +
  ggtitle("Proporção das variantes de /r/ pós-vocálico \nem três lojas
de departamento em Nova Iorque (N = 730)") +
  labs(x = "Lojas", y = "Proporção", fill = "Variantes de /r/") +
  scale_x_discrete(labels = c("Saks", "Macy's", "S. Klein")) +
  scale_fill_brewer(palette = "Purples", labels = c("realização", "apa
gamento")) +
  theme_bw()
```

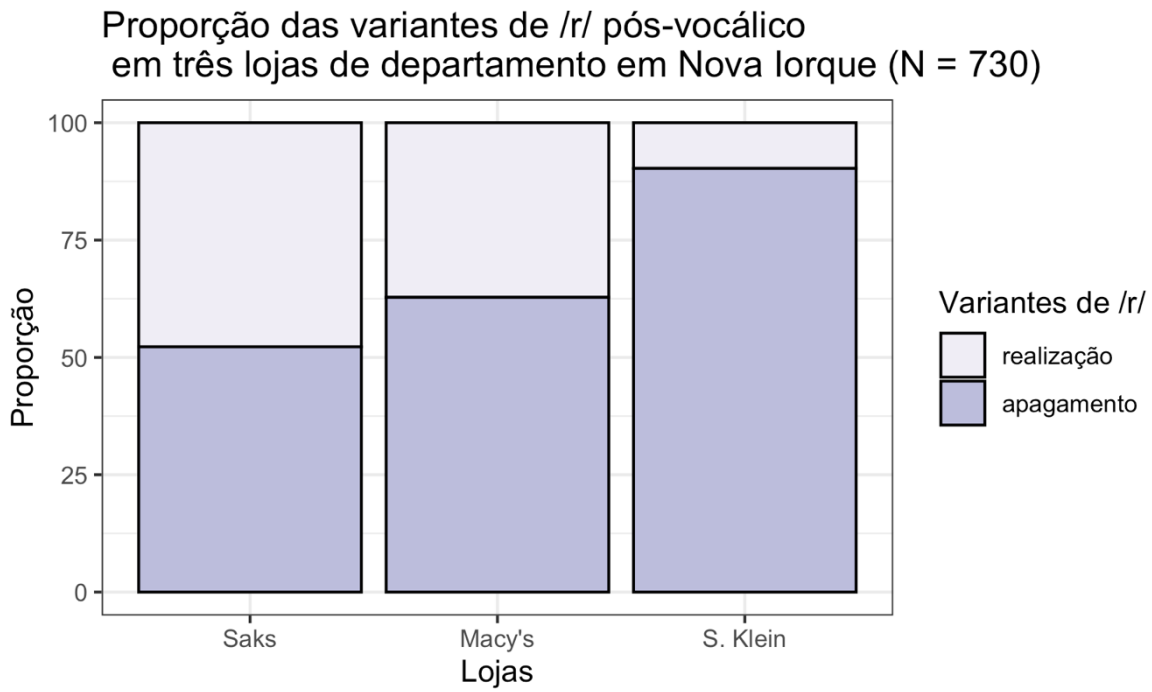

Figura 5.9: Gráfico de barras com cores das barras customizadas. Fonte: própria.

Maravilha! Você pode comparar as figuras que produziu clicando sobre a flecha para a esquerda, no topo esquerdo da aba Plots. Resta agora saber como exportar essa figura para inseri-la em pôsteres, artigos, na tese. No *script*, há duas linhas de comando comentadas: `png()` e `dev.off()`. A primeira define o nome da figura a ser exportada – algo que, evidentemente, merece um nome mais claro do que "figura.png", junto à sua extensão .png. Essa função abre o dispositivo gráfico do R. Depois você deve plotar a figura e, em seguida, rodar o comando `dev.off()`, que fecha o dispositivo gráfico.

Troque o nome da figura e rode a linha de comando; em seguida, rode novamente o comando que gera o gráfico e rode a linha de comando `dev.off()`.

```
png("figPropLojas.png")

ggplot(df.store, aes(x = store, y = prop * 100, fill = r)) +
  geom_bar(stat = "identity", color = "black") +
  ggtitle("Proporção das variantes de /r/ pós-vocálico \nem três lojas
de departamento em Nova Iorque (N = 730)") +
  labs(x = "Lojas", y = "Proporção", fill = "Variantes de /r/") +
  scale_x_discrete(labels = c("Saks", "Macy's", "S. Klein")) +
  scale_fill_brewer(palette = "Purples", labels = c("realização", "apa
gamento")) +
  theme_bw()
```

```
dev.off()
```

A figura em formato .png foi exportada para o seu computador. Onde ela está?

- na pasta Meus Documentos
- numa pasta aleatória
- no diretório de trabalho atual

Outra figura que podemos fazer a partir da mesma tabela é um gráfico de linhas. Antes de ver como fazê-lo, é preciso dar um alerta importante: gráficos de linha só são adequados a variáveis nominais se elas também forem ordinais, ou seja, se puderem ser colocadas numa ordem de acordo com algum critério. Imagine tentar colocar numa ordem os fatores "feminino" e "masculino". Não faria sentido, pois nenhum deles é "mais" ou "menos" do que outro, seja qual for o critério!

No caso das lojas de departamento, Labov as escolheu por serem três lojas com diferentes públicos de consumidores. Saks é a loja de maior prestígio e com preços mais elevados; Macy's é uma loja mais voltada à classe média; e S. Klein era uma loja "popular", com preços mais baratos. (Tão baratos, talvez, que fechou as portas na década de 1970). Desse modo, é possível ordená-las num ranking de maior para menor prestígio (ou vice-versa).

A estrutura da linha de comando para plotar o gráfico de linhas está no *script*. Note que os parâmetros para plotar esse gráfico são muito semelhantes ao do gráfico de barras. Em `ggplot()`, especificamos `df.store` como o dataframe e, dentro dos parâmetros estéticos, especificamos x = store e y = prop * 100. Neste caso, não estamos usando o argumento fill, pois não há o que preencher. No entanto, estamos usando o argumento group com a variável r, para que sejam plotadas duas linhas distintas, uma para r1 e outra para r0. (Depois, teste retirar esse argumento para ver o resultado!)

A geometria para gráficos de linhas é `geom_line()`. Seus argumentos especificam o tipo de linha, o tamanho da linha e sua cor. Aqui, estamos usando a linha "dotted". Para ver outros tipos de linhas, visite a página de referência do ggplot2: https://ggplot2.tidyverse.org/articles/ggplot2-specs.html.

Você já conhece as funções `ggtitle()`, `labs()` e `scale_x_discrete()`, que usamos no gráfico de barras. Rode então a linha de comando. O resultado se encontra na Figura 5.10.

```
ggplot(df.store, aes(x = store, y = prop * 100, group = r)) +
  geom_line(linetype = "dotted", size = 1, color = "blue") +
#  geom_point(shape = , size = , fill = "") +
  ggtitle("Proporção das variantes de /r/ pós-vocálico \nem três lojas
de departamento em Nova Iorque (N = 730)") +
  labs(x = "Lojas", y = "Proporção", fill = "Variantes de /r/") +
  scale_x_discrete(labels = c("Saks", "Macy's", "S. Klein")) +
#  ylim() +
  theme_bw()
```

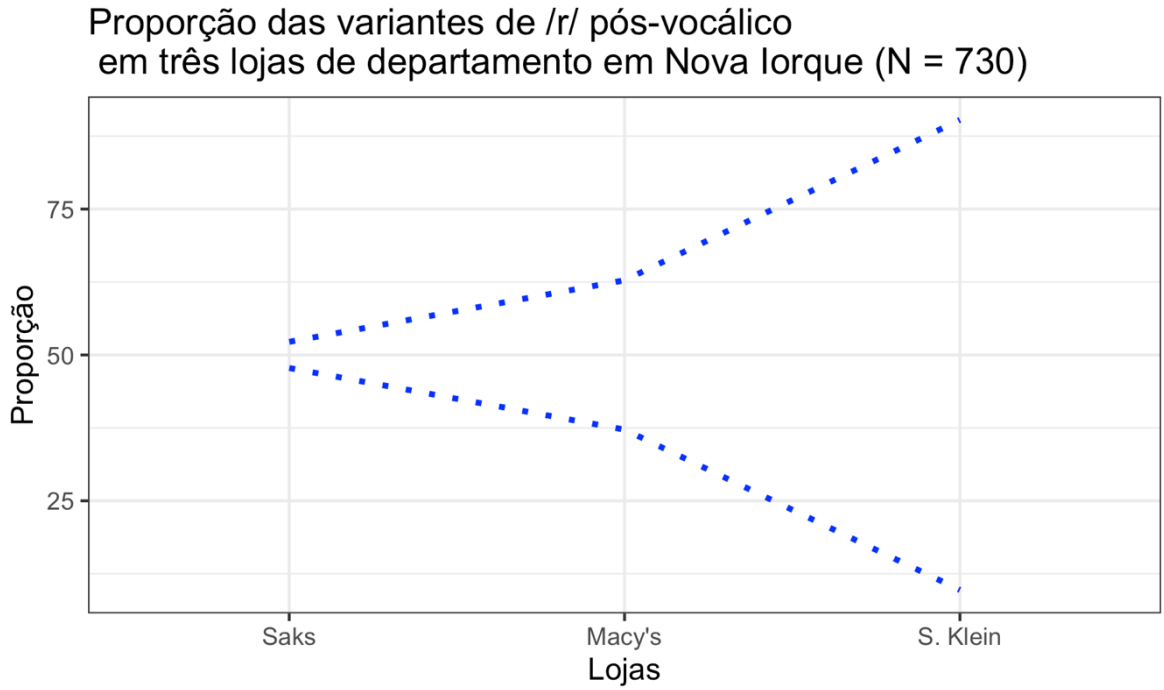

Figura 5.10: Gráfico de linhas simples com ggplot. Fonte: própria.

No ggplot2, é possível adicionar mais de uma geometria ao mesmo gráfico, somando-se novas camadas com outras funções do tipo geom_X(). No *script*, isso é exemplificado pela adição de `geom_point()` ao mesmo gráfico de linhas. Volte à página da Internet em que visualizou os tipos de linhas; mais abaixo da página, há uma lista de opções para o argumento shape e um guia para os tamanhos (em Colour e Fill). Aqui, vamos usar o formato 18, o tamanho 3 e a cor "black". Não se esqueça de apagar o # dessa linha (Figura 5.11).

```
ggplot(df.store, aes(x = store, y = prop * 100, group = r)) +
  geom_line(linetype = "dotted", size = 1, color = "blue") +
  geom_point(shape = 18, size = 3, fill = "black") +
  ggtitle("Proporção das variantes de /r/ pós-vocálico \nem três lojas
de departamento em Nova Iorque (N = 730)") +
  labs(x = "Lojas", y = "Proporção", fill = "Variantes de /r/") +
  scale_x_discrete(labels = c("Saks", "Macy's", "S. Klein")) +
#   ylim() +
  theme_bw()
```

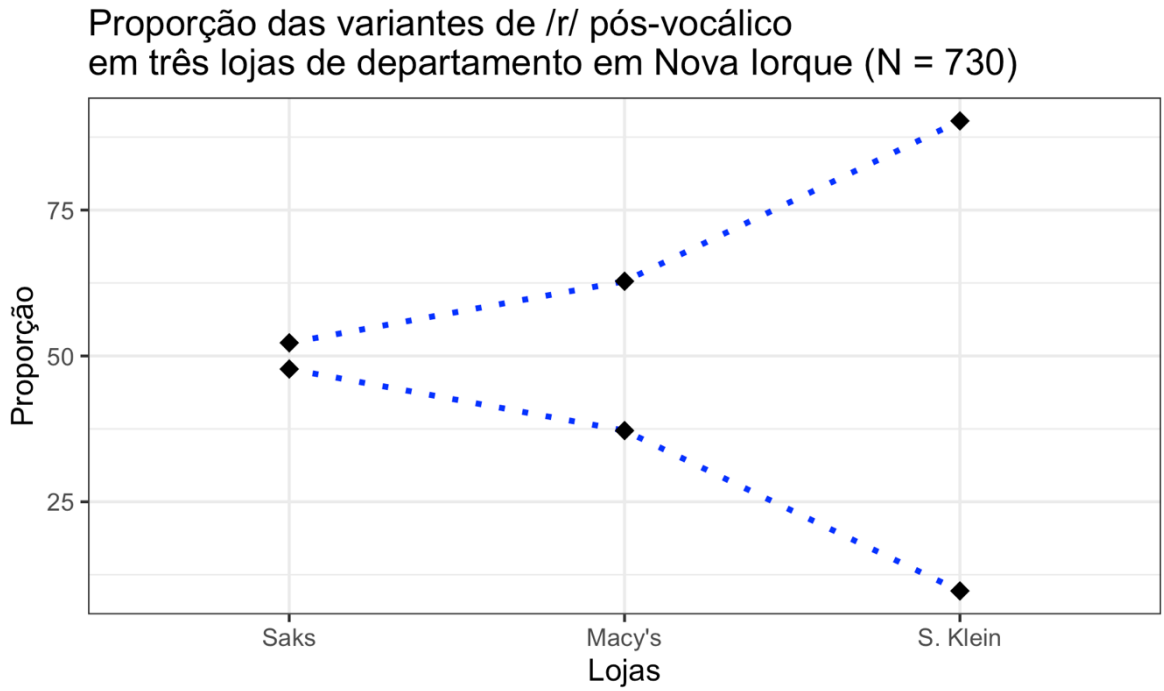

Figura 5.11: Gráfico de linhas com adição de `geom_point()`. Fonte: própria.

Vejamos a função `ylim()`, que permite definir os limites do eixo y (em outros gráficos, pode ser pertinente mudar o eixo x: `xlim()`). No momento, o ggplot2 está plotando o gráfico de acordo com os valores mínimo e máximo do gráfico, mas você pode querer especificar outros limites. Aqui, vamos colocar de 0 a 100. Para tanto, inclua na função ylim os argumentos 0 e 100 (Figura 5.12).

```
ggplot(df.store, aes(x = store, y = prop * 100, group = r)) +
  geom_line(linetype = "dotted", size = 1, color = "blue") +
  geom_point(shape = 18, size = 3, fill = "black") +
  ggtitle("Proporção das variantes de /r/ pós-vocálico \nem três lojas
de departamento em Nova Iorque (N = 730)") +
  labs(x = "Lojas", y = "Proporção", fill = "Variantes de /r/") +
  scale_x_discrete(labels = c("Saks", "Macy's", "S. Klein")) +
  ylim(0, 100) +
  theme_bw()
```

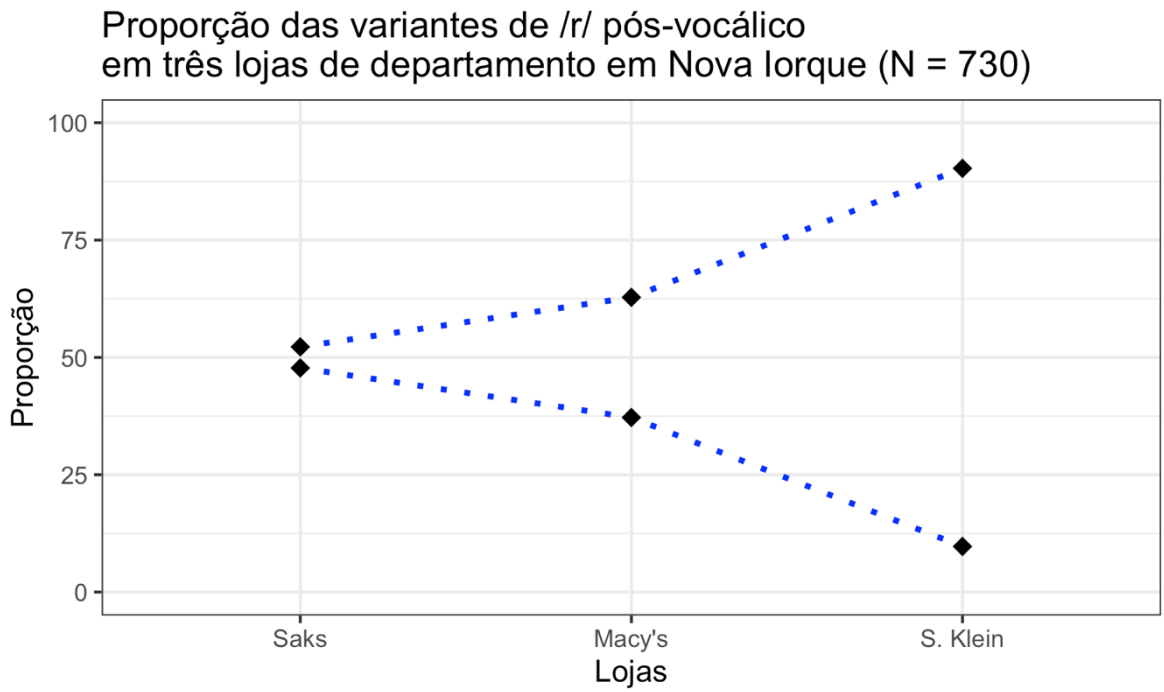

Figura 5.12: Gráfico de linhas com definição dos limites do eixo y. Fonte: própria.

Por fim, vamos fazer mais um ajuste. No gráfico, as duas linhas são redundantes pois, como já observado, todo /r/ que não foi realizado foi apagado. Só uma das linhas basta para comunicar a informação sobre a diferença entre as lojas quanto à pronúncia do /r/. Há mais de uma maneira para fazer esse ajuste. Aqui, vamos fazer uso do pipe, que você aprendeu na aula passada. A partir de `df.store`, use o pipe e, em seguida, a função `filter()` para determinar que quer apenas os dados de r0. Use novamente o pipe e copie e cole o código para o gráfico que acabamos de plotar. No entanto, vai ser necessário mudar o argumento que especifica o dataframe, pois agora ele não é mais `df.store`, mas a modificação que fizemos nele. Quando `ggplot()` aparece na sequência de um pipe, usamos o ponto final para indicar que o dataframe é o resultado das operações anteriores.

Revise a linha de comando. Por fim, mude o título do gráfico para "Proporção de apagamento de /r/ pós-vocálico \nem três lojas de departamento em Nova Iorque (N = 730)" e rode-a para plotar o gráfico apenas com dados de r0 (Figura 5.13).

```
df.store %>%
  filter(r == "r0") %>%
  ggplot(., aes(x = store, y = prop * 100, group = r)) +
  geom_line(linetype = "dotted", size = 1, color = "blue") +
  geom_point(shape = 18, size = 3, fill = "black") +
  ggtitle("Proporção de apagamento de /r/ pós-vocálico \nem três lojas
de departamento em Nova Iorque (N = 730)") +
  labs(x = "Lojas", y = "Proporção", fill = "Variantes de /r/") +
  scale_x_discrete(labels = c("Saks", "Macy's", "S. Klein")) +
  ylim(0, 100) +
  theme_bw()
```

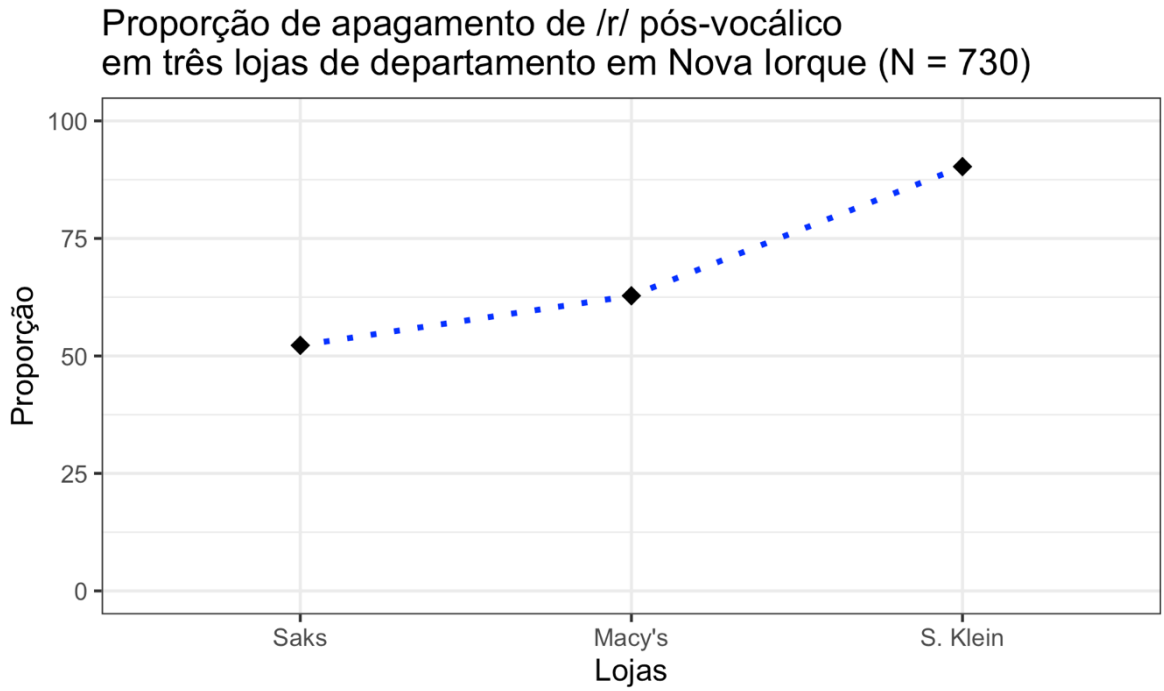

Figura 5.13: Gráfico de linhas com dados de apenas uma variante. Fonte: própria.

Você pode estar pensando: legal, sei que posso fazer um gráfico customizado, do jeitinho que eu quero. Mas eu nunca vou lembrar de todas essas funções na hora em que eu for fazer meu próprio gráfico! Calma! Primeiro, lembre-se que a Ajuda do R está sempre lá, na ponta dos dedos, para você consultar o momento que quiser!

Segundo, dificilmente se tem que digitar códigos diversas vezes. Você pode reutilizar o código quantas vezes quiser, fazendo apenas as adaptações nos trechos necessários. O mais importante é você saber lê-lo, entendê-lo, e saber o que precisa ser modificado.

Além disso, tenha em mente que o grau de detalhes de sua figura depende do propósito. Há gráficos exploratórios – para sua própria compreensão de seus dados – e gráficos explanatórios – cujo objetivo é o de comunicar a outros as suas descobertas. Se se trata de uma exploração sua dos dados, nos primeiros passos da análise, não é necessário usar todos os recursos que usamos aqui. Você pode guardar esse preciosismo para quando estiver preparando uma apresentação.

Ao concluir esta aula, dedique um tempo para revisar as linhas de comando desta lição. E faça os exercícios! Ao final do *script*, deixei mais algumas informações sobre como definir exatamente as dimensões da figura que você quer exportar.

Para saber mais

Leia a seção 3.1 (p.99–109) de Gries (2019) para mais informações sobre gráficos para representar frequências.

Exercícios

Para esta lição, vamos usar novamente o arquivo de dados LabovDS.csv, com a exclusão de dados "d".

1. Carregue o pacote tidyverse.
2. Carregue o pacote RColorBrewer.
3. Defina como diretório de trabalho aquele que, em seu computador, contém a planilha LabovDS.csv.
4. Carregue os dados da planilha em um dataframe chamado `ds`, especificando que todas as colunas são do tipo factor.
5. Cheque o dataframe `ds` por meio da função `str()`.
6. Da coluna r, exclua os dados "d" por meio da função `filter()`. Guarde o novo dataframe em um objeto chamado `ds2` e exclua o nível "d" do dataframe.

7. A partir de `ds2`, faça a tabela de distribuição de dados da variável emphasis pela variável dependente r usando funções da instalação base do R. Guarde a tabela em um objeto chamado `tab.emphasis`.
8. A partir de `ds2`, faça a tabela de distribuição de dados da variável word pela variável dependente r usando funções da instalação base do R. Guarde a tabela em um objeto chamado `tab.word`.
9. A partir de `ds2`, faça uma tabela de proporções da variável emphasis pela variável dependente r usando funções da instalação base do R. Multiplique a tabela de proporções por 100 para melhor visualização dos resultados. Guarde a tabela de proporções em um objeto chamado `prop.emphasis`.
10. A partir de `ds2`, faça uma tabela de proporções da variável word pela variável dependente r usando funções da instalação base do R. Multiplique a tabela de proporções por 100 para melhor visualização dos resultados. Guarde a tabela de proporções em um objeto chamado `prop.word`.
11. Usando funções do pacote tidyverse, faça um dataframe chamado `df.emphasis` das frequências e das proporções dos dados da variável emphasis pela VD r. Nomeie a coluna de proporções prop. Multiplique a tabela de proporções por 100 para melhor visualização dos resultados.
12. Usando funções do pacote tidyverse, faça um dataframe chamado `df.word` das frequências e das proporções dos dados da variável word pela VD r. Nomeie a coluna de proporções prop. Multiplique a tabela de proporções por 100 para melhor visualização dos resultados.
13. Visualize o dataframe `df.emphasis`.
14. Para plotar um gráfico de barras das proporções de r1 e r0 por contexto, a partir de `df.emphasis`, qual variável deve ocupar o eixo x?
    a. emphasis
    b. n
    c. prop
    d. r

15. Para plotar um gráfico de barras das proporções de r1 e r0 por contexto, a partir de `df.emphasis`, qual variável deve ocupar o eixo y?
    a. emphasis
    b. n
    c. prop
    d. r
16. Em um gráfico de barras das proporções de r1 e r0 por contexto, a partir de `df.emphasis`, se quisermos colorir as barras das variantes com cores diferentes, qual argumento deve ser usado dentro da função de parâmetros estéticos?
    a. color
    b. fill
    c. group
    d. linetype
17. A partir do dataframe `df.emphasis`, faça um gráfico de barras simples das proporções de apagamentos e realizações de r por contexto de ênfase. Use apenas as funções `ggplot()` e `geom_bar()`. Na primeira, utilize os parâmetros estéticos x, y e fill, de modo que cada barra represente um contexto de ênfase e cada variante seja representada por uma cor diferente. Na segunda função, utilize os argumentos stat = "identity" e color = "black".
18. Faça um gráfico de barras mais elaborado das proporções de r1 e r0 por contexto de ênfase. Para isso, a partir da linha de comando acima, especifique, nessa ordem: (i) os parâmetros estéticos x, y e fill dentro da função `ggplot()`; (ii) a geometria de barras, com stat = "identity" e color = "black"; (iii) o título do gráfico como "Proporções das variantes de /r/ pós-vocálico por contexto de ênfase"; (iv) o rótulo das barras como "casual" e "enfático"; (v) o rótulo do eixo x como "Contexto" e o rótulo do eixo y como "Proporção"; e (vi) o tema do gráfico como `theme_minimal()`. Revise a linha de comando antes de rodá-la!
19. No ggplot2, é possível plotar barras horizontais com a simples adição da função `coord_flip()`. Faça isso a partir da última linha de comando.

20. Faça um gráfico de barras simples das proporções de apagamentos e realizações de r por palavra, fourth ou floor. Use apenas as funções `ggplot()` e `geom_bar()`. Utilize os parâmetros estéticos x, y e fill, de modo que cada barra represente um item lexical e cada variante seja representada por uma cor diferente. Defina a cor da borda das barras como cinza.
21. Faça um gráfico de barras mais elaborado das proporções de r1 e r0 por item lexical. Para isso, a partir da linha de comando acima, especifique, nessa ordem: (i) os parâmetros estéticos x, y e fill dentro da função `ggplot()`; (ii) a geometria de barras, com barras empilhadas e a borda das barras na cor cinza; (iii) o rótulo do eixo x como "Item Lexical" e do eixo y como "Proporção"; (iv) os rótulos das barras como "fourth" e "floor"; (v) o título do gráfico como "Proporções das variantes de /r/ pós-vocálico por item lexical"; (vi) a paleta de cores "Greens" do RColorBrewer; e (vii) o tema do gráfico como `theme_classic()`. Revise a digitação antes de rodar o comando!
22. No ggplot2, é possível fazer gráficos de barras lado a lado, em vez de empilhadas, por meio do argumento position = "dodge" dentro da função `geom_bar()`. Inclua-o na última linha de comando e rode-a.
23. O estudo de Labov nas lojas de departamento em Nova Iorque na década de 1960 já foi replicado pelo menos duas vezes, em Fowler (1986) e Guy et al (2008). Tais estudos atestam a importância de métodos replicáveis em trabalhos científicos e nos informam sobre a continuidade de uma mudança em progresso no inglês de Nova Iorque. Rode a linha de comando a seguir para carregar um dataframe com os dados de Labov, Fowler e Guy et al.

```
source("https://raw.githubusercontent.com/oushiro/IELv2.0/main/criarDSrealtime.R")

## 'data.frame':    6 obs. of  3 variables:
##  $ study: Factor w/ 3 levels "Labov63","Fowler86",..: 1 2 3 1 2 3
##  $ r1   : num  29 39 57 20 28 60
##  $ store: Factor w/ 2 levels "Macys","Saks": 2 2 2 1 1 1
```

24. O dataframe `DS.real.time` contém os dados de realização de /r/ (r1) nas lojas Saks e Macy's para cada um dos três estudos. Inspecione esse dataframe agora.

25. Com esses dados, você fará um gráfico de linhas que compara a progressão da pronúncia de /r/ pós-vocálico nos três estudos. O eixo x será formado pelos estudos de Labov 1963, Fowler 1986 e Guy et al 2008, da esquerda para a direita. O eixo y indicará as taxas de realização de /r/ (r1). Serão plotadas duas linhas, uma para cada loja. Qual variável do dataframe deve ocupar o parâmetro estético x?
    a. store
    b. study
    c. r1
26. Qual variável do dataframe deve ocupar o parâmetro estético y?
    a. store
    b. study
    c. r1
27. Qual parâmetro estético informa ao R que queremos plotar duas linhas distintas, uma para cada loja?
    a. color
    b. group
    c. linetype
    d. shape
28. A partir do dataframe `DS.real.time`, plote um gráfico de linhas das proporções de realização de /r/ pós-vocálico nos três estudos em lojas de departamento em Nova Iorque, em que cada loja é representada por uma linha; cada linha será de um tipo e de uma cor diferentes. Para isso, especifique, nessa ordem: (i) os parâmetros estéticos x, y, group e color, dentro da função `ggplot()`; (ii) a geometria de linha, com o argumento aes(linetype = store); (iii) o título "Proporção de realização de /r/ pós-vocálico em três estudos \nem lojas de departamento em Nova Iorque"; (iv) o rótulo do eixo x como "" (vazio, para não haver rótulo), do eixo y como "Proporção de r1", do tipo de linha (linetype) como "Loja", e da cor (color) também como "Loja"; (v) os rótulos de cada ponto do

gráfico, da esquerda para a direita, como “Labov 1963”, “Fowler 1986” e “Guy et al 2008”; (vi) os limites do eixo y de 0 até 70; (vii) a paleta de cores “Dark2”, por meio da função `scale_color_brewer()`; e (viii) o tema do gráfico como `theme_bw()`. São vários passos, então faça com calma e revise a digitação antes de rodar!

Lição 6: Variáveis Numéricas: Medidas de Tendências Centrais

N.B.: Rode as linhas de comando a seguir antes de iniciar esta lição.

```
notas_turmaA <- c(7.0, 10.0, 10.0, 0.5, 10.0, 8.2, 9.5, 8.1, 5.0, 8.9,
                  8.2, 7.0, 1.5, 5.5, 9.3, 9.3, 9.3, 1.5, 7.0, 9.5, 6.
0,
                  7.5, 9.9, 8.0, 8.1, 8.8, 2.1, 7.0, 9.0, 0.0, 7.2)
notas_turmaB <- c(6.5, 8.5, 9.4, 7.5, 9.3, 9.9, 9.5, 9.8, 0.0, 0.0)

Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
  }
```

Nas duas últimas lições, lidamos com variáveis nominais/categóricas, e as medidas estatísticas apropriadas para esse tipo de dado: frequências e proporções. Também vimos dois tipos de gráfico bastante usados para representar visualmente os dados de variáveis nominais – o gráfico de barras e o gráfico de linhas (este último, para variáveis ordinais).

Nesta e na próxima lição, vamos tratar de variáveis *numéricas*. Primeiro, vamos revisar alguns conceitos básicos do Ensino Médio: *medidas de tendência central* e *medidas de dispersão*. Você certamente se lembra de uma das medidas de tendência central – a média – e se lembra como calculá-la.

Para refrescá-la na memória, tomemos dois conjuntos de dados, que deixei disponíveis para você nesta sessão: `notas_turmaA` e `notas_turmaB`. Cheque a aba Environment. Imagine que essas são as notas dadas por um professor a duas turmas diferentes ao final do semestre. Primeiro inspecione as notas da turma A. Digite `notas_turmaA` e veja o resultado.

```
notas_turmaA
```

```
##  [1]  7.0 10.0 10.0  0.5 10.0  8.2  9.5  8.1  5.0  8.9  8.2  7.0  1
.5
## [14]  5.5  9.3  9.3  9.3  1.5  7.0  9.5  6.0  7.5  9.9  8.0  8.1  8
.8
## [27]  2.1  7.0  9.0  0.0  7.2
```

Agora inspecione as notas da turma B.

```
notas_turmaB
```

```
##  [1] 6.5 8.5 9.4 7.5 9.3 9.9 9.5 9.8 0.0 0.0
```

Você deve ter notado que a turma A tem mais alunos do que a turma B. Vamos agora calcular a média de cada turma, para verificar qual teve uma nota média mais alta. Você se lembra, do Ensino Médio, que a média é o resultado da soma de todos os elementos dividida pelo número de elementos do conjunto. No R, uma função para realizar operações de adição é `sum()` e para mostrar o número de elementos de um vetor é `length()`. O operador de divisão é /. De posse dessas informações, calcule a média das notas dos alunos da turma A.

```
sum(notas_turmaA) / length(notas_turmaA)
```

```
## [1] 7.06129
```

Agora calcule a média para as notas da turma B.

```
sum(notas_turmaB) / length(notas_turmaB)
```

```
## [1] 7.04
```

Você pode ter imaginado que, para uma medida tão comum como a média, o R tem um jeito mais fácil de chegar a esse resultado – e você tem razão! A função `mean()` faz justamente o que foi feito acima. Aplique-a às notas da turma A para verificar o mesmo resultado.

```
mean(notas_turmaA)
```

```
## [1] 7.06129
```

E aplique a função `mean()` às notas da turma B.

```
mean(notas_turmaB)
```

```
## [1] 7.04
```

Vemos que as médias das duas turmas são bem parecidas. Vejamos agora outra medida de tendência central: a mediana. Dessa talvez você não se lembre, já que não é de uso tão corrente no cotidiano. Se um conjunto de medições for colocado na ordem crescente, a mediana é a observação bem no ponto médio desse conjunto ordenado. Quando um conjunto tem um número ímpar de observações, como é o caso de

`notas_turmaA`, a mediana é o valor de n/2 (ou seja, 31/2 = 15,5), arredondado para cima (=16). A mediana será, então, o 16º valor do vetor colocado em ordem crescente. Colocado de outro modo, a mediana é o valor que separa a metade inferior da metade superior da amostra (15 observações para cada lado).

Coloque os elementos do vetor `notas_turmaA` com uso da função `sort()`. Verifique qual é o valor do 16º elemento para achar a mediana.

```
sort(notas_turmaA)
```

```
##  [1]  0.0  0.5  1.5  1.5  2.1  5.0  5.5  6.0  7.0  7.0  7.0  7.0  7
.2
## [14]  7.5  8.0  8.1  8.1  8.2  8.2  8.8  8.9  9.0  9.3  9.3  9.3  9
.5
## [27]  9.5  9.9 10.0 10.0 10.0
```

Você deve ter encontrado o valor 8,1, certo? Vamos fazer o mesmo agora para `notas_turmaB`. Do mesmo modo que acima, primeiro se colocam os termos em ordem crescente. Neste caso, em que o vetor tem um número par de observações, tomamos dois números para calcular a mediana: n/2 arredondado para baixo e n/2 arredondado para cima. Como `notas_turmaB` tem 10 elementos, a mediana é a média da 5ª e da 6ª observação dos elementos organizados em ordem crescente. Aplique então a função `sort()` para descobrir quais são a 5ª e a 6ª medição de `notas_turmaB`.

```
sort(notas_turmaB)
```

```
##  [1] 0.0 0.0 6.5 7.5 8.5 9.3 9.4 9.5 9.8 9.9
```

Você deve ter encontrado os valores 8,5 e 9,3. A média entre esses dois valores é 8,9, que é a mediana de `notas_turmaB`. Mas você deve estar pensando: "Por que essa complicação toda? Eu vou ter que ficar colocando os números na ordem crescente e procurando no meio da distribuição qual é a observação n/2 arredondada pra cima ou pra baixo? Eu sei que o R tem um jeito mais fácil de calcular a mediana!" E é claro que tem! É a função `median()`. Aplique-a agora às notas da turma A.

```
median(notas_turmaA)
```

```
## [1] 8.1
```

Aplique a função `median()` às notas da turma B.

```
median(notas_turmaB)
```

```
## [1] 8.9
```

Mesmos valores que calculamos previamente, certo? O motivo de eu não ter ido direto ao ponto é porque é fácil aplicar ferramentas computacionais sem saber direito o que se está fazendo. Média e mediana são duas medidas de cálculo fácil, que muitas vezes podem ser feitas à mão ou com uma calculadora, desde que o número de observações não seja muito extenso. Mas por mais banais que pareçam ser essas medidas, *todos* os testes estatísticos que existem derivam em menor ou maior grau delas. É importante ter clareza sobre como essas medidas são calculadas.

Há ainda uma terceira medida de tendência central, chamada moda. A moda é o valor que ocorre mais frequentemente em um conjunto de dados. Interessantemente, o R não tem uma função própria para calcular a moda de um conjunto de dados, mas eu deixei uma tal função disponível para você nesta lição. Ela se chama Mode (com 'M' maiúsculo). Aplique-a às notas dos alunos da turma A.

```
Mode(notas_turmaA)
## [1] 7
```

E agora aplique-a às notas dos alunos da turma B.

```
Mode(notas_turmaB)
## [1] 0
```

Ok! Temos então três medidas para cada conjunto de dados, resumidas na Tabela 6.1.

Tabela 6.1: Medidas de tendência central das turmas A e B.

	Turma A	Turma B
Média	7,06	7,04
Mediana	8,1	8,9
Moda	7,0	0,0

Fonte: própria.

Turma A: 7,06, 8,1 e 7,0 e Turma B: 7,04, 8,9 e 0,0 para média, mediana e moda respectivamente. Embora a média de ambas as turmas tenha sido praticamente a mesma,

há diferenças nas medidas de mediana e moda. Você pode estar se perguntando agora: "Qual é a melhor e qual eu devo usar?"

Pois bem, retire essas perguntas imediatamente da cabeça! A questão aqui não é a melhor ou a pior, mas sim o que cada uma informa. Vale a pena fazer as três medições em todo conjunto de dados numéricos que você tiver. Quando essas três medidas são parecidas entre si, isso significa que você está diante de um conjunto equilibrado de dados, em que todos os valores se distribuem de modo mais ou menos simétrico em torno do ponto médio.

Graficamente, tal conjunto de dados se distribuiria mais ou menos como a imagem do meio da Figura 6.1.

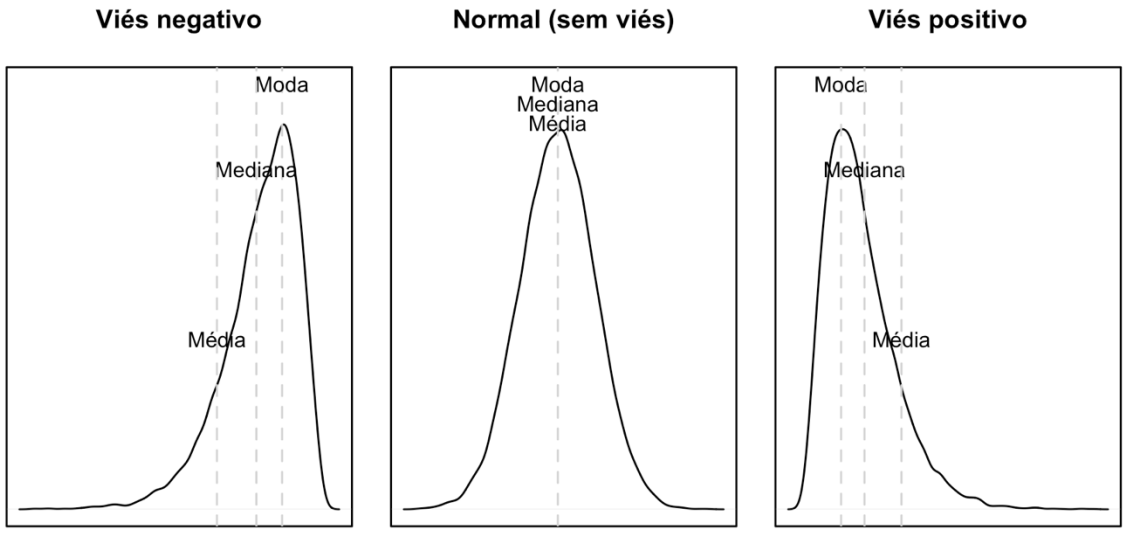

Figura 6.1: Distribuições. Fonte: própria.

Quando as três medidas diferem mais entre si, isso é indicativo de que a distribuição de seus dados é assimétrica, ou seja, que ela provavelmente tem um viés negativo ou positivo. A mediana tanto da turma A quanto da turma B está acima de 8,0, o que significa que pelo menos metade dos alunos de cada turma tirou uma nota acima de 8,0, mas a média está "mascarando" isso – o viés, aqui, é negativo. Na situação contrária – a mediana abaixo da média –, o viés seria positivo. Esses vieses estão ilustrados na Figura 6.1.

Algo semelhante tende a ocorrer com a moda. Quando uma distribuição é simétrica, o valor mais frequente (o ponto mais alto da distribuição) tende a estar próximo da média. Isso parece ser mais verdadeiro para a turma A do que para a turma B, em que a moda difere radicalmente dos demais valores de medidas centrais.

Também há que se levar em conta o número de elementos de um conjunto para avaliar o quanto essas três medidas são representativas dele. Na turma B, há apenas 10 alunos, 3 vezes menos do que na turma A. Isso significa que cada aluno da turma B contribui mais para a média do que cada aluno da turma A.

Façamos um teste: acrescente a cada um dos vetores notas_turmaA e notas_turmaB uma nova nota 0. Digite `notas_turmaA2 <- c(notas_turmaA, 0)`. (Certifique-se que você entende o que está sendo feito nesta linha de comando; se não, reveja a Lição 1.)

```
notas_turmaA2 <- c(notas_turmaA, 0)
```

Vamos fazer o mesmo agora para a turma B. Digite `notas_turmaB2 <- c(notas_turmaB, 0)`.

```
notas_turmaB2 <- c(notas_turmaB, 0)
```

Aplique agora a função `mean()` a notas_turmaA2.

```
mean(notas_turmaA2)
## [1] 6.840625
```

E aplique a função `mean()` ao vetor notas_turmaB2.

```
mean(notas_turmaB2)
## [1] 6.4
```

Veja que a média da turma A baixou de 7,06 para 6,84 (=0,22 de diferença), enquanto a média da turma B baixou de 7,04 para 6,4 (=0,64). A média do segundo grupo baixou quase três vezes mais do que a do primeiro! Isso é porque uma única observação "conta mais" num grupo menor de dados do que num grupo maior e, portanto, tem mais potencial para mudar os resultados. A lição a se tirar daqui é que quanto menor o número de dados de que se dispõe, mais cautela se deve ter na leitura de resultados dos testes estatísticos!

Média, mediana e moda resumem, cada qual, um conjunto de dados em um único número. Mas interessa também saber como os dados se dispersam. Afinal, um conjunto A (5, 5, 5) e outro B (0, 5, 10) têm ambos a mesma média e a mesma mediana (=5), mas são bem diferentes. Para capturar a diferença entre um tal conjunto A e um tal conjunto B, calculamos *medidas de dispersão*.

A *variância* é uma das medidas de dispersão. Seu cálculo é também bastante simples e está ilustrado na Tabela 6.2 (10 primeiros valores de `notas.turmaA`):

Tabela 6.2: Cálculo da variância.

1: Obs	**2**: Obs - μ	**3**: (Obs - μ)^2
7,0	-0,06	0,0036
10,0	2,94	8,6436
10,0	2,94	8,6436
0,5	-6,56	43,0346
10,0	2,94	8,6436
8,2	1,14	1,2996
9,5	2,44	5,9536
8,1	1,04	1,0816
5,0	-2,06	4,2436
8,9	1,84	3,3856
...	...	...
		4: Σ = 264,1136
		5: Σ / 30 = 8,8038

Fonte: própria.

Toma-se cada uma das observações (1), e dela se subtrai o valor da média (2). No vetor notas_turmaA, isso seria: 7 – 7,06; 10 – 7,06; 10 – 7,06 etc. Em seguida, eleva-se cada resultado das subtrações ao quadrado (3). Os quadrados servem o propósito de eliminar o sinal negativo – como você se lembra do Ensino Médio, qualquer número elevado ao quadrado é positivo. Somam-se então os valores quadrados (4). O resultado da soma é dividido pelo número de observações N (se o cálculo for da *população*) ou por n - 1 (se o cálculo for da *amostra*) (5). Não vou entrar em detalhes do porquê disso aqui.

Os curiosos podem consultar este site: http://duramecho.com/Misc/WhyMinusOneInSd.html.

No R, a função para calcular a variância é `var()`. Aplique-a às notas da turma A.

```
var(notas_turmaA)
## [1] 8.803785
```

Aplique agora a função `var()` às notas da turma B.

```
var(notas_turmaB)
## [1] 14.92044
```

Vemos que a variância é maior na turma B. Isso significa que a perfórmance dos estudantes da turma B foi menos homogênea, mais dispersa, do que os da turma A. Mas esses números são um pouco difíceis de interpretar. O que significam 8,8 ou 14,9 de variância? Uma medida mais "intuitiva" é o desvio padrão.

O desvio padrão é a raiz quadrada da variância. Lembra que elevamos todos os valores de diferença entre uma observação e a média ao quadrado? A raiz quadrada reverte essa operação. No R, a função para calcular o desvio padrão é `sd()` (=standard deviation). Aplique-a agora ao vetor `notas_turmaA`.

```
sd(notas_turmaA)
## [1] 2.967117
```

E aplique `sd()` ao vetor `notas_turmaB`.

```
sd(notas_turmaB)
## [1] 3.862699
```

O desvio padrão da turma A é 2,96 e da turma B é 3,86. Esses valores já são mais facilmente interpretáveis: em média, os alunos da turma A desviaram 2,96 pontos da média, e os alunos da turma B desviaram mais. Novamente, a interpretação é que a perfórmance dos alunos da turma A foi relativamente mais homogênea do que a dos alunos da turma B.

Outra medida estatística útil, e que aparecerá nos modelos de regressão linear (Lições 12 e 13), é o erro padrão. Não há uma função específica no R para computá-lo, mas sua definição matemática é bastante simples: o erro padrão é igual ao desvio padrão,

dividido pela raiz quadrada do número de observações. Aplique então essa fórmula a notas_turmaA. (A função para calcular a raiz quadrada é `sqrt()` e para computar o número de observações é `length()`).

```
sd(notas_turmaA) / sqrt(length(notas_turmaA))
## [1] 0.53291
```

E faça o cálculo do erro padrão para notas_turmaB.

```
sd(notas_turmaB) / sqrt(length(notas_turmaB))
## [1] 1.221493
```

Mas para que servem todas essas medidas? Um dos objetivos da análise estatística é criar modelos ou fazer previsões (vamos falar mais sobre isso adiante no curso). As medidas de tendência central resumem em poucos números uma distribuição que pode ter dezenas, centenas, milhares, milhões de dados. As medidas de dispersão dão um indicativo do quanto as medidas de tendência central conseguem prever uma determinada medição. Quanto maior a variância, desvio padrão e erro padrão, menos informativas são as medidas de tendências centrais. A previsão de uma medida dificilmente se refere a um valor exato. Na maior parte das vezes, tais medidas vão ser base para estimar *probabilidades*.

Que tal aplicar esse conhecimento em algo *muito mais importante*, como a análise da altura das vogais médias pretônicas na fala de migrantes nordestinos? Primeiro, defina como diretório de trabalho aquele que, em seu computador, contém o arquivo Pretonicas.csv.

```
setwd("~/Dropbox/_R/swirl/Introducao_a_Estatistica_para_Linguistas/data")
```

N.B.: O diretório em seu computador provavelmente vai ser diferente!

Vamos carregar a planilha em um dataframe chamado `pretonicas`. Para tanto, carregue o pacote tidyverse.

```
library(tidyverse)
```

Use a função `read_csv()` para carregar a planilha. Nela, defina as variáveis AMOSTRA e VOGAL como `col_factor()`; a primeira tem os níveis "PBSP" e "SP2010", e a segunda tem os níveis "i", "e", "a", "o" e "u".

```
pretonicas <- read_csv("Pretonicas.csv",
                       col_types = cols(AMOSTRA = col_factor(levels =
c("PBSP", "SP2010")),
                                        VOGAL = col_factor(levels = c(
"i", "e", "a", "o", "u"))
                                        )
                       )
```

E você também já sabe que a primeira coisa a se fazer após carregar os dados é inspecioná-los. Aplique `str()` ao dataframe `pretonicas` para se certificar de que os dados foram carregados corretamente e ter uma visão global deles.

```
str(pretonicas)
```

```
## spec_tbl_df [2,415 × 27] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ PALAVRA      : chr [1:2415] "fazer" "quatorze" "casou" "casado"
...
##  $ Transc.Fon   : chr [1:2415] "f<a>-'zer" "k<a>-'tor-ze" "k<a>-'zo
w" "k<a>-'za-do" ...
##  $ VOGAL        : Factor w/ 5 levels "i","e","a","o",..: 3 3 3 3 3
3 4 3 3 3 ...
##  $ F1           : num [1:2415] 487 686 731 621 845 ...
##  $ F2           : num [1:2415] 1666 1414 1168 1275 1574 ...
##  $ F1.NORM      : num [1:2415] 397 476 494 450 540 ...
##  $ F2.NORM      : num [1:2415] 1517 1386 1258 1314 1469 ...
##  $ CONT.PREC    : chr [1:2415] "f" "k" "k" "k" ...
##  $ CONT.SEG     : chr [1:2415] "z" "t" "z" "z" ...
##  $ VOGAL.SIL.SEG: chr [1:2415] "e" "o" "ow" "a" ...
##  $ F1.SIL.SEG   : num [1:2415] 498 462 529 842 509 ...
##  $ F2.SIL.SEG   : num [1:2415] 2001 1126 1009 1239 2351 ...
##  $ F1.SEG.NORM  : num [1:2415] 328 317 338 433 331 ...
##  $ F2.SEG.NORM  : num [1:2415] 1518 1095 1038 1149 1687 ...
##  $ VOGAL.TONICA : chr [1:2415] "e" "o" "ow" "a" ...
##  $ DIST.TONICA  : num [1:2415] 1 1 1 1 1 1 1 1 1 1 ...
##  $ ESTR.SIL.PRET: chr [1:2415] "CV" "CV" "CV" "CV" ...
##  $ Begin.Time.s : num [1:2415] 20.4 20.6 33.6 36.5 40.3 ...
##  $ End.Time.s   : num [1:2415] 20.4 20.6 33.6 36.5 40.4 ...
##  $ Duration.ms  : num [1:2415] 19.1 20.2 40.7 25.2 34.7 ...
##  $ AMOSTRA      : Factor w/ 2 levels "PBSP","SP2010": 1 1 1 1 1 1 1
1 1 1 ...
##  $ PARTICIPANTE : chr [1:2415] "MartaS" "MartaS" "MartaS" "MartaS"
...
##  $ SEXO         : chr [1:2415] "feminino" "feminino" "feminino" "fe
minino" ...
##  $ IDADE        : num [1:2415] 32 32 32 32 32 32 32 32 32 32 ...
##  $ IDADE.CHEGADA: num [1:2415] 18 18 18 18 18 18 18 18 18 18 ...
##  $ ANOS.SP      : num [1:2415] 14 14 14 14 14 14 14 14 14 14 ...
```

```
##  $ CONTEXTO     : chr [1:2415] “ai aqui j\u0087 tem treze ano vai f
azer quatorze” “ai aqui j\u0087 tem treze ano vai fazer quatorze” “a\u
0092 depois ele voltou a gente casou e viemos” “que l\u0087 voc\u0090
s\u0097 podia sair se fosse casado n\u008e se fosse pra” ...
##  - attr(*, “spec”)=
##   .. cols(
##   ..   PALAVRA = col_character(),
##   ..   Transc.Fon = col_character(),
##   ..   VOGAL = col_factor(levels = c(“i”, “e”, “a”, “o”, “u”), orde
red = FALSE, include_na = FALSE),
##   ..   F1 = col_double(),
##   ..   F2 = col_double(),
##   ..   F1.NORM = col_double(),
##   ..   F2.NORM = col_double(),
##   ..   CONT.PREC = col_character(),
##   ..   CONT.SEG = col_character(),
##   ..   VOGAL.SIL.SEG = col_character(),
##   ..   F1.SIL.SEG = col_double(),
##   ..   F2.SIL.SEG = col_double(),
##   ..   F1.SEG.NORM = col_double(),
##   ..   F2.SEG.NORM = col_double(),
##   ..   VOGAL.TONICA = col_character(),
##   ..   DIST.TONICA = col_double(),
##   ..   ESTR.SIL.PRET = col_character(),
##   ..   Begin.Time.s = col_double(),
##   ..   End.Time.s = col_double(),
##   ..   Duration.ms = col_double(),
##   ..   AMOSTRA = col_factor(levels = c(“PBSP”, “SP2010”), ordered =
FALSE, include_na = FALSE),
##   ..   PARTICIPANTE = col_character(),
##   ..   SEXO = col_character(),
##   ..   IDADE = col_double(),
##   ..   IDADE.CHEGADA = col_double(),
##   ..   ANOS.SP = col_double(),
##   ..   CONTEXTO = col_character()
##   .. )
##  - attr(*, “problems”)=<externalptr>
```

Na Lição 2, também vimos a função `summary()`, que fornece uma visão global dos dados com medidas estatísticas relevantes. Aplique-a agora ao objeto `pretonicas`.

```
summary(pretonicas)
```

```
##    PALAVRA           Transc.Fon        VOGAL         F1
##  Length:2415        Length:2415        i:409   Min.   : 122.5
##  Class :character   Class :character   e:686   1st Qu.: 426.4
##  Mode  :character   Mode  :character   a:415   Median : 510.4
##                                        o:639   Mean   : 524.6
##                                        u:266   3rd Qu.: 618.7
##                                                Max.   :1095.4
##
##        F2            F1.NORM         F2.NORM        CONT.PREC
##  Min.   : 672.9   Min.   :302.1   Min.   : 946.9   Length:2415
##  1st Qu.:1144.6   1st Qu.:395.3   1st Qu.:1247.4   Class :character
```

```
##  Median :1432.9   Median :425.7   Median :1412.4   Mode  :character
##  Mean   :1477.2   Mean   :427.1   Mean   :1437.8
##  3rd Qu.:1786.3   3rd Qu.:455.0   3rd Qu.:1613.8
##  Max.   :2593.5   Max.   :578.1   Max.   :1994.9
##
##    CONT.SEG         VOGAL.SIL.SEG        F1.SIL.SEG
##  Length:2415        Length:2415        Min.   : 169.6
##  Class :character   Class :character   1st Qu.: 488.9
##  Mode  :character   Mode  :character   Median : 602.9
##                                        Mean   : 611.3
##                                        3rd Qu.: 725.1
##                                        Max.   :2163.0
##
##    F2.SIL.SEG      F1.SEG.NORM     F2.SEG.NORM     VOGAL.TONICA
##  Min.   : 620.8   Min.   :252.1   Min.   : 883.7   Length:2415
##  1st Qu.:1277.5   1st Qu.:341.2   1st Qu.:1213.6   Class :character
##  Median :1478.6   Median :375.6   Median :1293.7   Mode  :character
##  Mean   :1547.0   Mean   :372.3   Mean   :1333.4
##  3rd Qu.:1747.0   3rd Qu.:403.6   3rd Qu.:1442.0
##  Max.   :3182.9   Max.   :648.5   Max.   :1974.5
##
##   DIST.TONICA    ESTR.SIL.PRET       Begin.Time.s
##  Min.   :1.000   Length:2415        Min.   :   7.95
##  1st Qu.:1.000   Class :character   1st Qu.: 378.90
##  Median :1.000   Mode  :character   Median : 938.50
##  Mean   :1.192                      Mean   :1133.67
##  3rd Qu.:1.000                      3rd Qu.:1709.12
##  Max.   :5.000                      Max.   :4735.69
##
##    End.Time.s        Duration.ms       AMOSTRA     PARTICIPANTE
##  Min.   :   7.966   Min.   : 4.211   PBSP  :1171   Length:2415
##  1st Qu.: 378.928   1st Qu.:16.271   SP2010:1244   Class :character
##  Median : 938.521   Median :20.775                 Mode  :character
##  Mean   :1133.695   Mean   :21.896
##  3rd Qu.:1709.143   3rd Qu.:26.661
##  Max.   :4735.718   Max.   :97.504
##
##      SEXO               IDADE       IDADE.CHEGADA      ANOS.SP
##  Length:2415        Min.   :30.00   Min.   :13.00   Min.   :14.00
##  Class :character   1st Qu.:31.00   1st Qu.:14.00   1st Qu.:15.00
##  Mode  :character   Median :35.00   Median :17.00   Median :17.00
##                     Mean   :34.63   Mean   :16.51   Mean   :18.82
##                     3rd Qu.:37.00   3rd Qu.:18.00   3rd Qu.:23.00
##                     Max.   :42.00   Max.   :21.00   Max.   :25.00
##                                     NA's   :1244    NA's   :1244
##    CONTEXTO
##  Length:2415
##  Class :character
##  Mode  :character
##
##
##
##
```

Veja que, para variáveis numéricas, como F1 e F2, a função `summary()` fornece os valores de média e de mediana, além de outras medidas como mínimo, máximo e quartis. Veremos essas outras medidas com mais detalhes na próxima lição. Por ora, voltemos às medidas de tendências centrais.

O arquivo Pretonicas.csv contém medições de F1 e F2 de vogais pretônicas de 7 falantes paraibanos que migraram para a cidade de São Paulo (amostra PBSP). Além disso, o arquivo também contém as mesmas medições para 7 paulistanos nativos (amostra SP2010), que servem como um parâmetro de comparação para os padrões de fala dos migrantes.

A questão mais geral por trás desses dados é descobrir se alguns migrantes paraibanos se acomodaram aos padrões da nova comunidade, e quais padrões são esses. Para isso, foram extraídas medições de F1 e F2 de cerca de 130 a 180 vogais pretônicas da fala de cada um desses participantes, com especial interesse nas vogais médias /e/ e /o/ (como em 'relógio' e 'romã'), em contextos linguísticos que favorecem o abaixamento dessas vogais (as realizações média-baixas [ɛ] e [ɔ]). Em princípio se espera que os paulistanos tenham vogais médias relativamente mais altas do que os paraibanos, mas que alguns dos migrantes tenham se aproximado do novo padrão.

Linguística e acusticamente, a altura das vogais é medida pelo F1, em Hertz. Quanto mais alto é o valor de F1, mais *baixa* é a vogal. Coloquei o quadro de vogais do IPA na Figura 6.2 para facilitar essa visualização.

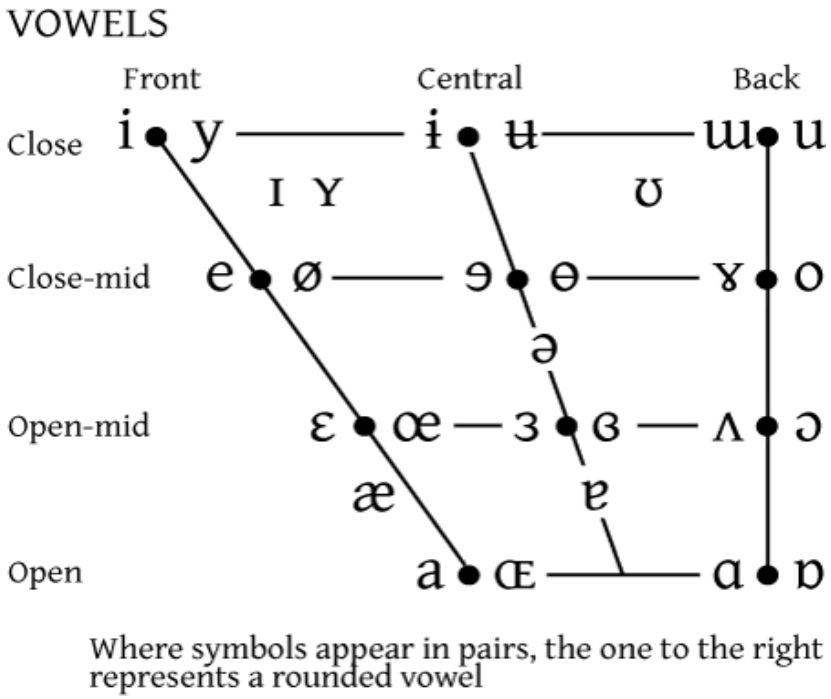

Figura 6.2: Vogais no IPA. Fonte: International Phonetic Association.[9]

No português brasileiro, as vogais [i] e [u] costumam ter valores mais baixos de F1, as vogais [e] e [o] um pouco mais altos, as vogais [ɛ] e [ɔ] mais altos ainda, e a vogal [a] os valores mais altos de F1. (Consulte o manual de Barbosa & Madureira, 2015 para mais informações!). Na página https://en.wikipedia.org/wiki/IPA_vowel_chart_with_audio, você pode visualizar o diagrama de vogais do International Phonetic Alphabet e escutar os sons das vogais.

Da lista de variáveis desse dataframe, qual é a variável dependente?

- CONT.PREC
- F1
- F2
- Palavra
- Transc.Fon

[9] Disponível em https://www.internationalphoneticassociation.org/sites/default/files/IPA_Kiel_2015.pdf.

- Vogal

Uma das questões que queremos responder é: qual é a altura média de cada vogal (principalmente /e/ e /o/) para cada comunidade (PB vs SP)? Considerando-se que há 5 vogais pretônicas e 2 grupos, precisaríamos calcular a média 10 vezes. Além disso, temos interesse, minimamente, na mediana e no desvio padrão, para ter uma ideia de se os dados se distribuem simetricamente ou não, e o grau de sua dispersão. Você ficou cansado só de pensar em fazer tudo isso?

A boa notícia é que, como sempre no R, alguém também já pensou que isso dá muito trabalho e criou uma função para fazer isso mais rapidamente. No tidyverse, isso pode ser feito com as funções `group_by()` – que você já conhece – e `summarize()`, ambas do dplyr.

Veja a linha de comando neste ponto do *script*, que precisará ser completada por você.

```
# Não rodar! Estrutura do código:

df %>%
  group_by(VAR, VAR) %>%
  summarize(nomeVar = mean(VAR))
```

Nela, pedimos para que o R pegue um dataframe e, primeiro, agrupe o dados de acordo com duas variáveis; em seguida, pedimos para que o R compute as médias de outra variável (com a função `mean()`), e guarde o resultado em uma variável chamada nomeVar. Substitua então os termos adequados nos pontos correspondentes: nosso dataframe é `pretonicas`, e queremos agrupar os dados por AMOSTRA e VOGAL; a variável sobre a qual vamos computar as médias é F1; e o nome que vamos atribuir a essa nova variável com as médias é `media_F1`.

```
pretonicas %>%
  group_by(AMOSTRA, VOGAL) %>%
  summarize(media_F1 = mean(F1))

## # A tibble: 10 × 3
## # Groups:   AMOSTRA [2]
##    AMOSTRA VOGAL media_F1
##    <fct>   <fct>    <dbl>
##  1 PBSP    i         410.
##  2 PBSP    e         579.
```

```
##  3 PBSP    a         695.
##  4 PBSP    o         617.
##  5 PBSP    u         446.
##  6 SP2010  i         350.
##  7 SP2010  e         477.
##  8 SP2010  a         660.
##  9 SP2010  o         508.
## 10 SP2010  u         396.
```

O resultado é um tibble, que tem as colunas AMOSTRA, VOGAL e media_F1, e que informa, para cada vogal de cada amostra, qual é a média.

Note que a ordem em que as variáveis são colocadas em `group_by()` determina a ordem de apresentação dos resultados. Para comparar, crie uma nova linha de comando, semelhante à de cima, mas que inverte a ordem das variáveis de agrupamento.

```
pretonicas %>%
  group_by(VOGAL, AMOSTRA) %>%
  summarize(media_F1 = mean(F1))
```

```
## # A tibble: 10 × 3
## # Groups:   VOGAL [5]
##    VOGAL AMOSTRA media_F1
##    <fct> <fct>      <dbl>
##  1 i     PBSP        410.
##  2 i     SP2010      350.
##  3 e     PBSP        579.
##  4 e     SP2010      477.
##  5 a     PBSP        695.
##  6 a     SP2010      660.
##  7 o     PBSP        617.
##  8 o     SP2010      508.
##  9 u     PBSP        446.
## 10 u     SP2010      396.
```

A ordem VOGAL, AMOSTRA torna mais diretamente comparáveis os valores de F1. Vemos que, para todas as vogais, os valores de F1 são maiores (= vogais mais baixas) para os paraibanos do que para os paulistanos.

Dentro da função `summarize()`, podemos já incluir outras medidas estatísticas como novos argumentos. A partir da última linha de comando, mantenha a ordem VOGAL, AMOSTRA para o agrupamento em `summarize()`, e inclua, além do cálculo da média, o cálculo da mediana e do desvio padrão de F1 (nessa ordem), atribuindo a essas novas variáveis os nomes `mediana_F1` e `sd_F1` respectivamente.

```
pretonicas %>%
  group_by(VOGAL, AMOSTRA) %>%
```

```
  summarize(media_F1 = mean(F1),
            mediana_F1 = median(F1),
            sd_F1 = sd(F1)
            )
```

```
## # A tibble: 10 × 5
## # Groups:   VOGAL [5]
##    VOGAL AMOSTRA media_F1 mediana_F1 sd_F1
##    <fct> <fct>      <dbl>      <dbl> <dbl>
##  1 i     PBSP        410.       408.  77.3
##  2 i     SP2010      350.       353.  88.9
##  3 e     PBSP        579.       564. 113.
##  4 e     SP2010      477.       476. 109.
##  5 a     PBSP        695.       663. 127.
##  6 a     SP2010      660.       661. 118.
##  7 o     PBSP        617.       598. 116.
##  8 o     SP2010      508.       509. 114.
##  9 u     PBSP        446.       439.  68.4
## 10 u     SP2010      396.       402. 105.
```

Nesta lição, vimos as medidas de tendência central – média, mediana, moda – e de dispersão – variância, desvio padrão e erro padrão. Em seguida, em um conjunto real de dados, aplicamos a função `summarize()` para rapidamente visualizar algumas dessas medidas por subconjuntos de dados.

O corpus do Projeto SP2010 (http://projetosp2010.fflch.usp.br/) – gravações, transcrições e fichas dos informantes – está todo disponível gratuitamente *on-line*. Convido você a visitar essa página!

Para saber mais

Recomendo a leitura do capítulo 4 de Dalgaard (2008) sobre Estatística Descritiva.

Exercícios

Para estes exercícios, usaremos novamente o conjunto de dados de vogais pretônicas.

1. Carregue o pacote tidyverse.
2. Defina como diretório de trabalho aquele que, em seu computador, contém a planilha Pretonicas.csv.
3. Importe a planilha Pretonicas.csv em um dataframe chamado `pretonicas`. Defina a variável AMOSTRA como factor, com os níveis “PBSP” e “SP2010”; a

variável VOGAL como factor, com os níveis “i”, “e”, “a”, “o” e “u”; e a variável PARTICIPANTE também como factor (sem necessidade de definir a ordem dos níveis).

4. Inspecione o dataframe `pretonicas` com a função `str()`.
5. As variáveis F1.NORM e F2.NORM contêm valores dos formantes normalizados pelo método de Lobanov (1971). A normalização é um procedimento padrão em análises acústicas para minimizar a variação em medições de formantes por conta de diferenças anatômicas entre falantes (p.ex., mulheres tendem a medidas mais altas de formantes do que os homens). Visualize os 6 primeiros elementos da coluna F1.NORM.
6. Calcule a mediana de F1.NORM por VOGAL e por AMOSTRA (nessa ordem). Use o pipe e as funções `group_by()` e `summarize()`, nomeando a coluna para as medianas como mediana_F1.NORM.
7. Calcule a média de F1.NORM por VOGAL e por AMOSTRA (nessa ordem). Nomeie a coluna para as médias como media_F1.NORM.
8. No cálculo da média de F1 com vogais não normalizadas, nesta lição, havíamos visto que as medidas de F1 para paraibanos eram maiores para todas as vogais. Esse resultado se mantém com as vogais normalizadas? Explique sua resposta.
9. No cálculo da média de F1 com vogais normalizadas (F1.NORM), quais vogais têm medidas de F1 maiores (= vogais mais baixas) para paraibanos do que para paulistanos?
    a. /a/ e /o/
    b. /e/ e /o/
    c. /i/ e /e/
    d. /i/ e /o/
    e. /i/ e /u/
10. Se quisermos olhar apenas para as vogais /e/ e /o/, podemos criar um subconjunto de dados com a função `filter()`. Aplique esta função para criar o

subconjunto de dados de vogais médias. Guarde esse subconjunto num dataframe chamado `medias_pretonicas`.

11. Normalmente se verifica bastante variação entre diferentes indivíduos. Um dos interesses em comparar o padrão de paulistanos e migrantes paraibanos é tentar descobrir quais dos migrantes mais se acomodaram ao padrão paulistano (ao menos quanto às vogais médias pretônicas). Para observar o padrão individual, calcule a média de F1.NORM por AMOSTRA e por PARTICIPANTE, inicialmente apenas para a vogal "e". Nomeie a coluna com as médias como media_F1.NORM_e. Use como conjunto de dados o dataframe recém-criado `medias_pretonicas`.
12. Inspecionando o resultado da linha de comando anterior, responda: qual dos paraibanos mais se distancia da média de F1.NORM da vogal /e/ dos paulistanos (= 423 Hz, como calculado mais acima)?
    a. HenriqueA
    b. JoaoS
    c. JosaneV
    d. JosueO
    e. MarinalvaS
    f. MartaS
    g. PedroC
13. Inspecionando o mesmo dataframe, responda: qual dos paraibanos tem uma média de F1.NORM mais próxima da média paulistana para a vogal /e/ (= 423 Hz)?
    a. HenriqueA
    b. JoaoS
    c. JosaneV
    d. JosueO
    e. MarinalvaS
    f. MartaS
    g. PedroC

14. Calcule agora a média de F1.NORM por AMOSTRA e por PARTICIPANTE apenas para a vogal "o". Nomeie a coluna com as médias como media_F1.NORM_o. Use como conjunto de dados o dataframe `medias_pretonicas`.
15. Inspecionando o dataframe gerado pela última linha de comando, responda: qual dos paraibanos tem uma média de F1.NORM mais próxima da média paulistana para a vogal /o/ (= 435 Hz)?
    a. HenriqueA
    b. JoaoS
    c. JosaneV
    d. JosueO
    e. MarinalvaS
    f. PedroC
16. Qual dos paraibanos tem uma média de F1.NORM mais distante da média paulistana para a vogal /o/ (= 435 Hz)?
    a. HenriqueA
    b. JoaoS
    c. JosaneV
    d. JosueO
    e. MarinalvaS
    f. MartaS
    g. PedroC
17. A partir do dataframe `medias_pretonicas`, crie um dataframe chamado `medidas_medias_pretonicas`, com o cálculo da média, da mediana, do desvio padrão e do erro padrão de F1.NORM por VOGAL, AMOSTRA e PARTICIPANTE (nessa ordem). Nomeie as colunas das medições, respectivamente, como media_F1.NORM, mediana_F1.NORM, sd_F1.NORM e ep_F1.NORM.
18. Aplique a função `View()` ao dataframe `medidas_medias_pretonicas`.
19. Acima, aplicamos a função `View()`, pois não é possível visualizar todas as linhas do dataframe por meio do formato tibble. No entanto, o dplyr tem outras

funções que permitem rearranjar os dados de modo a facilitar certas visualizações. A função `arrange()` reorganiza os dados a partir de certas colunas. Para visualizar as medidas de F1.NORM dos paraibanos em ordem crescente, siga os seguintes passos, usando pipe: a partir de `medidas_medias_pretonicas`, filtre os dados da vogal "e" e aplique a função `arrange()` a AMOSTRA e media_F1.NORM.

20. Quem é o paulistano ou a paulistana com vogais /e/ pretônicas relativamente mais baixas?
    a. AliceC
    b. AnaS
    c. LucianoT
    d. MauricioB
    e. NelsonF
    f. RenataC
    g. RobsonF
21. Visualize agora o dataframe na ordem crescente das medidas de desvio padrão da vogal "o", com os dados dos paraibanos antes dos dados dos paulistanos.
22. Quem é o paraibano ou a paraibana com maior dispersão de medidas da altura de vogais /o/ pretônicas?
    a. HenriqueA
    b. JoaoS
    c. JosaneV
    d. JosueO
    e. MartaS
    f. MarinalvaS
    g. PedroC
23. Visualize o dataframe na ordem crescente das medidas de desvio padrão da vogal "e", sem ordenar as amostras.

24. Comparando as medidas de desvio padrão de F1.NORM entre paulistanos e paraibanos, em qual dos grupos há mais dispersão?
    a. PBSP
    b. SP2010

Lição 7: Variáveis Numéricas: Gráficos

Na lição anterior, fizemos algumas tabelas com medidas de médias, medianas e desvio padrão a partir da planilha de dados Pretonicas.csv. Rode as linhas de comando a seguir para carregá-la novamente, se necessário.

```
# Definir diretório de trabalho

#setwd()

# Importar planilha de dados

pretonicas <- read_csv("Pretonicas.csv",
                       col_types = cols(AMOSTRA = col_factor(levels =
c("PBSP", "SP2010")),
                                        VOGAL = col_factor(levels = c(
"i", "e", "a", "o", "u"))
                                        )
                       )
```

N.B.: Defina como diretório de trabalho aquele que contém o arquivo Pretonicas.csv.

Você pode ter achado difícil fazer sentido de tantos números – médias, medianas, desvios padrão... E de fato é! A compreensão de muitos dados estatísticos é sempre mais fácil por meio de gráficos – tanto para você, para entender o que está acontecendo em seus dados (gráficos exploratórios), quanto para seu leitor, ao qual futuramente você vai querer comunicar resultados (gráficos explanatórios). Nesta lição, vamos aprender a fazer alguns tipos de gráficos adequados para variáveis numéricas: *gráficos de linhas*, *gráficos de dispersão*, *boxplots* e *histogramas*.

Primeiro, carregue o pacote tidyverse. (Minha ideia é condicionar você a sempre carregar os pacotes necessários no início da sessão! Tá funcionando?)

```
library(tidyverse)
```

Agora, refamiliarize-se com o conjunto de dados de vogais pretônicas. Inspecione a estrutura do dataframe `pretonicas` com `str()`. Os níveis das variáveis VOGAL e AMOSTRA já foram definidos na importação dos dados.

```
str(pretonicas)
```

```
## spec_tbl_df [2,415 × 27] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ PALAVRA      : chr [1:2415] “fazer” “quatorze” “casou” “casado”
...
##  $ Transc.Fon   : chr [1:2415] “f<a>-'zer” “k<a>-'tor-ze” “k<a>-'zo
w” “k<a>-'za-do” ...
##  $ VOGAL        : Factor w/ 5 levels “i”,”e”,”a”,”o”,..: 3 3 3 3 3
3 4 3 3 3 ...
##  $ F1           : num [1:2415] 487 686 731 621 845 ...
##  $ F2           : num [1:2415] 1666 1414 1168 1275 1574 ...
##  $ F1.NORM      : num [1:2415] 397 476 494 450 540 ...
##  $ F2.NORM      : num [1:2415] 1517 1386 1258 1314 1469 ...
##  $ CONT.PREC    : chr [1:2415] “f” “k” “k” “k” ...
##  $ CONT.SEG     : chr [1:2415] “z” “t” “z” “z” ...
##  $ VOGAL.SIL.SEG: chr [1:2415] “e” “o” “ow” “a” ...
##  $ F1.SIL.SEG   : num [1:2415] 498 462 529 842 509 ...
##  $ F2.SIL.SEG   : num [1:2415] 2001 1126 1009 1239 2351 ...
##  $ F1.SEG.NORM  : num [1:2415] 328 317 338 433 331 ...
##  $ F2.SEG.NORM  : num [1:2415] 1518 1095 1038 1149 1687 ...
##  $ VOGAL.TONICA : chr [1:2415] “e” “o” “ow” “a” ...
##  $ DIST.TONICA  : num [1:2415] 1 1 1 1 1 1 1 1 1 1 ...
##  $ ESTR.SIL.PRET: chr [1:2415] “CV” “CV” “CV” “CV” ...
##  $ Begin.Time.s : num [1:2415] 20.4 20.6 33.6 36.5 40.3 ...
##  $ End.Time.s   : num [1:2415] 20.4 20.6 33.6 36.5 40.4 ...
##  $ Duration.ms  : num [1:2415] 19.1 20.2 40.7 25.2 34.7 ...
##  $ AMOSTRA      : Factor w/ 2 levels “PBSP”,”SP2010”: 1 1 1 1 1 1 1
1 1 1 ...
##  $ PARTICIPANTE : chr [1:2415] “MartaS” “MartaS” “MartaS” “MartaS”
...
##  $ SEXO         : chr [1:2415] “feminino” “feminino” “feminino” “fe
minino” ...
##  $ IDADE        : num [1:2415] 32 32 32 32 32 32 32 32 32 32 ...
##  $ IDADE.CHEGADA: num [1:2415] 18 18 18 18 18 18 18 18 18 18 ...
##  $ ANOS.SP      : num [1:2415] 14 14 14 14 14 14 14 14 14 14 ...
##  $ CONTEXTO     : chr [1:2415] “ai aqui j\u0087 tem treze ano vai f
azer quatorze” “ai aqui j\u0087 tem treze ano vai fazer quatorze” “a\u
0092 depois ele voltou a gente casou e viemos” “que l\u0087 voc\u0090
s\u0097 podia sair se fosse casado n\u008e se fosse pra” ...
##  - attr(*, “spec”)=
##   .. cols(
##   ..   PALAVRA = col_character(),
##   ..   Transc.Fon = col_character(),
##   ..   VOGAL = col_factor(levels = c(“i”, “e”, “a”, “o”, “u”), orde
red = FALSE, include_na = FALSE),
##   ..   F1 = col_double(),
##   ..   F2 = col_double(),
##   ..   F1.NORM = col_double(),
##   ..   F2.NORM = col_double(),
##   ..   CONT.PREC = col_character(),
##   ..   CONT.SEG = col_character(),
##   ..   VOGAL.SIL.SEG = col_character(),
##   ..   F1.SIL.SEG = col_double(),
##   ..   F2.SIL.SEG = col_double(),
##   ..   F1.SEG.NORM = col_double(),
##   ..   F2.SEG.NORM = col_double(),
```

```
##   ..   VOGAL.TONICA = col_character(),
##   ..   DIST.TONICA = col_double(),
##   ..   ESTR.SIL.PRET = col_character(),
##   ..   Begin.Time.s = col_double(),
##   ..   End.Time.s = col_double(),
##   ..   Duration.ms = col_double(),
##   ..   AMOSTRA = col_factor(levels = c("PBSP", "SP2010"), ordered =
FALSE, include_na = FALSE),
##   ..   PARTICIPANTE = col_character(),
##   ..   SEXO = col_character(),
##   ..   IDADE = col_double(),
##   ..   IDADE.CHEGADA = col_double(),
##   ..   ANOS.SP = col_double(),
##   ..   CONTEXTO = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Cheque a ordem dos níveis das vogais por meio da função `levels()` novamente. Digite `levels(pretonicas$VOGAL)`. Essa é a ordem em que as vogais aparecerão nos gráficos adiante.

```
levels(pretonicas$VOGAL)
```

```
## [1] "i" "e" "a" "o" "u"
```

Esse conjunto de dados contém as medições de F1 e F2 em Hertz, de valores brutos e normalizados (pelo método de Lobanov), para cada vogal pretônica. Nesta lição, trabalharemos com os valores normalizados (F1.NORM e F2.NORM).

Na Lição 5, plotamos um gráfico de linhas para as proporções de realização de /r/ em três lojas de departamento em Nova Iorque. O gráfico de linhas foi ali empregado porque a variável store, naquele conjunto de dados, pode ser considerada uma variável ordinal quanto ao grau de prestígio das lojas. Por outro lado, variáveis numéricas, como são as medidas de F1 e F2, sempre são intrinsicamente ordinais também: tem-se valores que vão de menor para maior.

Reveja o quadro de vogais do IPA (Figura 6.2). Se o imaginamos como um plano cartesiano, cada vogal é identificada por uma coordenada no eixo x e no eixo y. No entanto, há variação no espaço vocálico a depender do indivíduo, do item lexical, da comunidade etc.

Vamos plotar o espaço vocálico das pretônicas na fala de paraibanos residentes em São Paulo (PBSP), em comparação com a dos paulistanos nativos. Para isso, vamos

usar as médias de valores de F1 e F2 normalizados para cada comunidade, de modo a obter as coordenadas para o eixo x (medidas de F2) e eixo y (medidas de F1).

No *script* desta lição, a maior parte dos comandos contém a estrutura dos códigos que vamos usar aqui – cabe a você preenchê-lo com os dados relevantes! Como visto anteriormente, na maior parte do tempo, trabalhamos com *scripts*, sem a necessidade de digitar linhas de comando extensas. Se você conhece a sintaxe e os argumentos das funções, saberá como adaptá-los para suas necessidades.

Voltemos então para nosso gráfico de linhas. Para obter as médias de F1.NORM e F2.NORM para cada VOGAL e para cada AMOSTRA, vamos usar a função `summarize()`, como vimos na última lição. Examine o esqueleto da linha de comando no *script*.

```
# Não rodar! Estrutura do código:

novo.df <- df %>%
  group_by(VAR, VAR) %>%
  summarize(novaVAR1 = mean(VAR),
            novaVAR2 = mean(VAR)) %>%
  print()
```

Primeiro, preencha o nome do novo dataframe em que vamos guardar as medidas das médias: `medias`. Em seguida, explicite o dataframe do qual vamos extrair os dados: `pretonicas`. Vamos agrupar os dados por VOGAL e AMOSTRA. Por fim, vamos computar as medidas de F1.NORM e F2.NORM (nessa ordem) e guardá-las, respectivamente, em colunas chamadas `media_F1` e `media_F2`. Aqui, como estamos criando um novo dataframe com `<-`, usamos a função `print()` para que o R já mostre o resultado. Ao terminar, revise a digitação e rode com CTRL + ENTER.

```
medias <- pretonicas %>%
  group_by(VOGAL, AMOSTRA) %>%
  summarize(media_F1 = mean(F1.NORM),
            media_F2 = mean(F2.NORM)) %>%
  print()

## # A tibble: 10 × 4
## # Groups:   VOGAL [5]
##    VOGAL AMOSTRA media_F1 media_F2
##    <fct> <fct>      <dbl>    <dbl>
##  1 i     PBSP        373.    1688.
##  2 i     SP2010      379.    1717.
##  3 e     PBSP        432.    1612.
##  4 e     SP2010      423.    1606.
```

```
##  5 a     PBSP        475.    1377.
##  6 a     SP2010      488.    1369.
##  7 o     PBSP        445.    1217.
##  8 o     SP2010      435.    1237.
##  9 u     PBSP        386.    1184.
## 10 u     SP2010      395.    1204.
```

Temos agora as médias de F1 e F2 por vogal e por amostra, e podemos prosseguir para a plotagem do gráfico! Vamos plotar, primeiramente, um gráfico de linhas que representa o espaço vocálico de paraibanos em São Paulo e de paulistanos nativos, para que possamos compará-los. Cada ponto será uma das vogais i, e, a, o e u, e elas serão ligadas por uma linha. Já vimos, na Lição 5, que o ggplot2 precisa, pelo menos: (i) do dataframe do qual extrair as informações; (ii) dos parâmetros estéticos; e (iii) da geometria do gráfico a ser plotado – as duas primeiras linhas do comando. Mantenha as demais linhas com o #, por enquanto.

```
# Não rodar! Estrutura do código:

ggplot(df, aes(x = VAR, y = VAR, color = VAR)) +
  geom_line() +
#  geom_label() +
#  scale_x_reverse() +
#  scale_y_reverse() +
#  ggtitle("Valores médios de F1 e F2 normalizados nas amostras PBSP e
SP2010") +
#  labs(x = "F2 normalizado", y = "F1 normalizado") +
  theme_bw()
```

De qual dataframe vamos extrair os dados das médias de F1 e F2?

- ds
- medias
- pretonicas

Queremos plotar um gráfico do espaço vocálico, em que os eixos x e y vão representar as medidas de F1 e F2. Qual variável do dataframe em questão representa o eixo x?

- media_F1
- media_F2
- F1.NORM
- F2.NORM

Qual variável do dataframe em questão deve ocupar o eixo y?

- media_F1
- media_F2
- F1.NORM
- F2.NORM

No argumento color, queremos definir uma cor diferente para a linha dos paraibanos e outra para a dos paulistanos. Qual variável categoriza o local de origem dos falantes? (Desculpa por essa pergunta tão óbvia...)

- AMOSTRA
- VOGAL
- PARTICIPANTE

Com essas informações, já conseguimos plotar uma primeira versão do gráfico. Preencha o comando com essas informações, substituindo df e as palavras VAR, e rode com CTRL + ENTER. O resultado se encontra na Figura 7.1.

```
ggplot(medias, aes(x = media_F2, y = media_F1, color = AMOSTRA)) +
  geom_line() +
#  geom_label() +
#  scale_x_reverse() +
#  scale_y_reverse() +
#  ggtitle("Valores médios de F1 e F2 normalizados nas amostras PBSP e
SP2010") +
#  labs(x = "F2 normalizado", y = "F1 normalizado") +
  theme_bw()
```

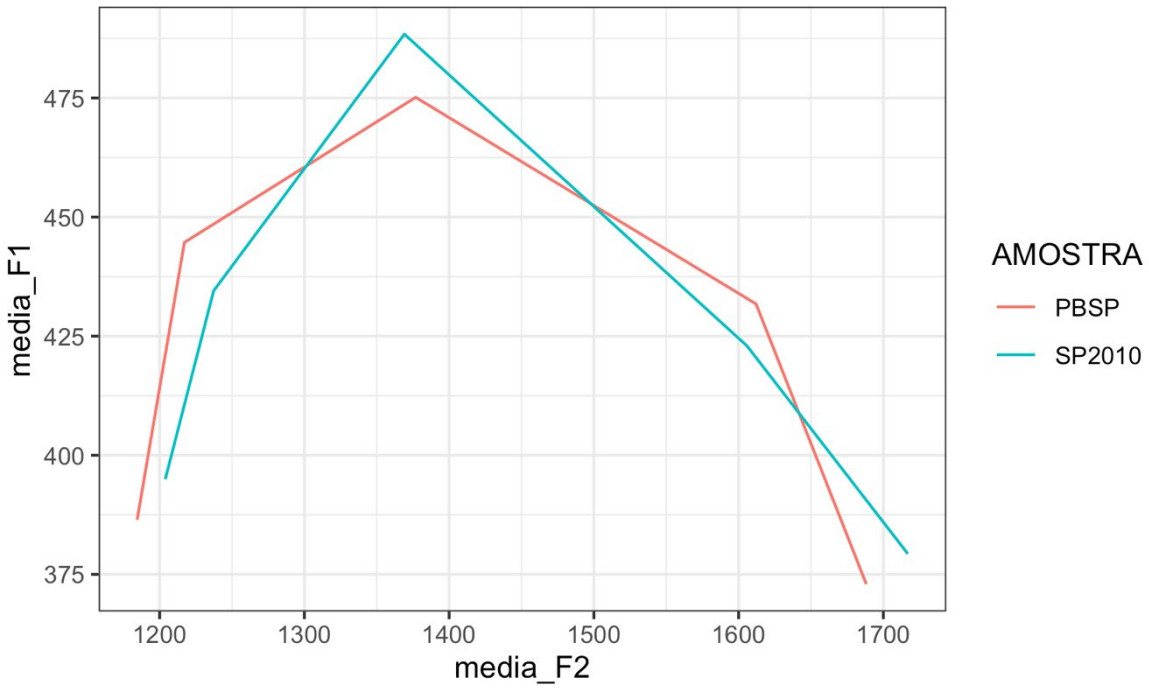

Figura 7.1: Gráfico de linhas com espaço vocálico de paraibanos e paulistanos. Fonte: própria.

Eita! Se você está familiarizado com a representação das vogais no quadro do IPA, deve ter percebido que tem um problema nesse gráfico! Os eixos x e y, ambos de variáveis numéricas/contínuas, estão ordenados de modo crescente: os valores aumentam de baixo para cima e da esquerda para a direita. Mas lembre-se de suas aulas de Fonética: os valores de F1 e de F2 são convencionalmente representados de modo invertido, para deixar as vogais anteriores à esquerda e as vogais fechadas na parte de cima.

Conseguimos consertar isso rapidinho com as funções `scale_x_reverse()` e `scale_y_reverse()`. Retire o comentário # dessas duas linhas e rode o comando novamente (Figura 7.2).

```
ggplot(medias, aes(x = media_F2, y = media_F1, color = AMOSTRA)) +
  geom_line() +
#  geom_label() +
  scale_x_reverse() +
  scale_y_reverse() +
#  ggtitle("Valores médios de F1 e F2 normalizados nas amostras PBSP e
SP2010") +
#  labs(x = "F2 normalizado", y = "F1 normalizado") +
  theme_bw()
```

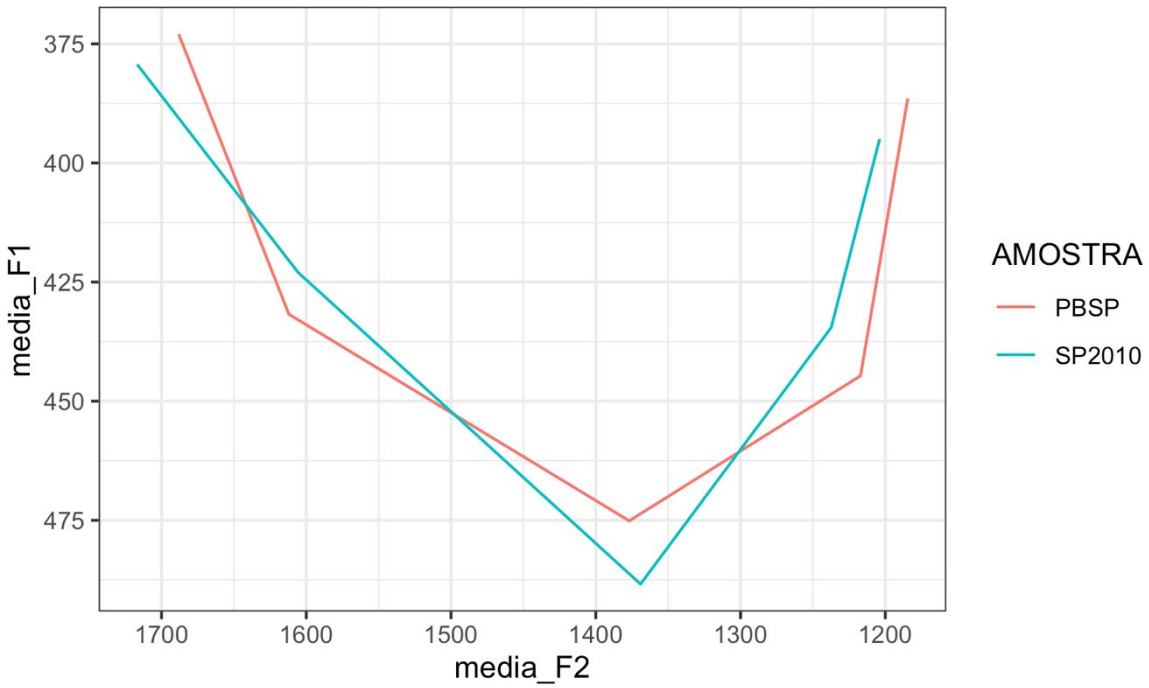

Figura 7.2: Gráfico de linhas com espaço vocálico de paraibanos e paulistanos com eixos x e y revertidos. Fonte: própria.

Bem melhor, não? Um linguista bem treinado já consegue ver esse gráfico e tirar várias informações interessantes. Mas vamos deixá-lo mais informativo, adicionando as vogais para facilitar sua interpretação. Para isso, podemos usar a geometria `geom_label()`, que insere rótulos em gráficos e que neste código está logo abaixo de `geom_line()`. Mas, para que as vogais sejam devidamente mapeadas aos rótulos, precisamos especificar qual variável contém os rótulos, com o argumento label = VOGAL dentro dos parâmetros estéticos `aes()`. Execute então esses dois passos: retire o # da linha com geom_label, e insira o argumento label em `aes()` (Figura 7.3).

```
ggplot(medias, aes(x = media_F2, y = media_F1,
                   color = AMOSTRA, label = VOGAL)) +
  geom_line() +
  geom_label() +
  scale_x_reverse() +
  scale_y_reverse() +
#  ggtitle("Valores médios de F1 e F2 normalizados nas amostras PBSP e
SP2010") +
#  labs(x = "F2 normalizado", y = "F1 normalizado") +
  theme_bw()
```

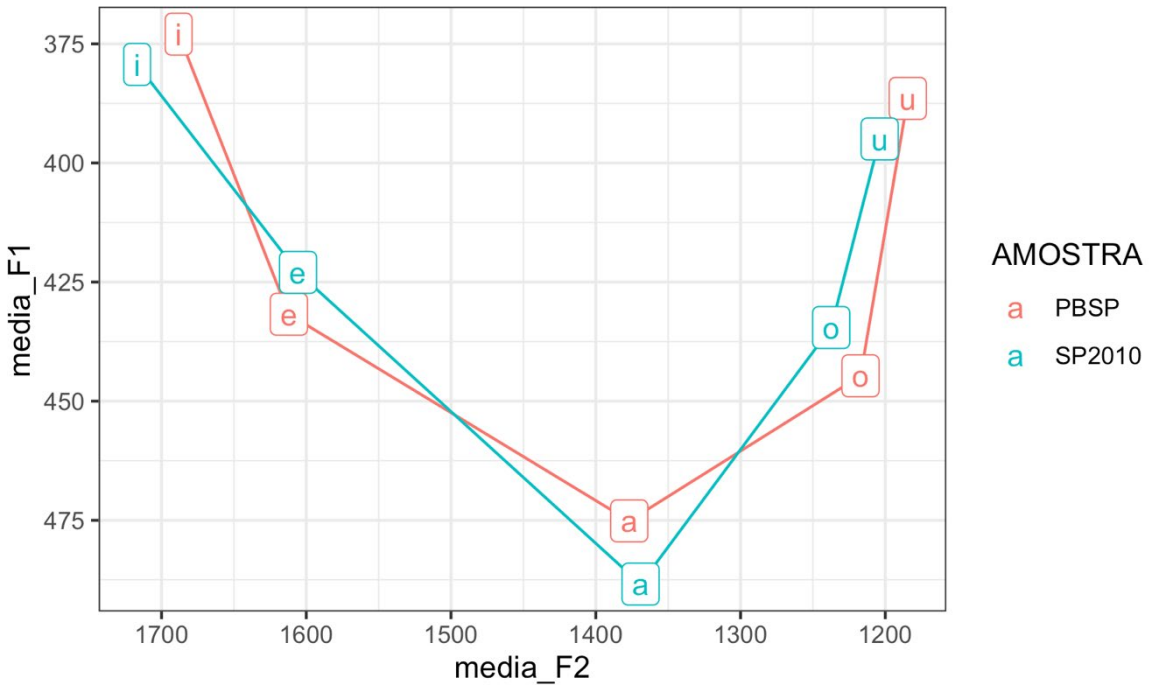

Figura 7.3: Gráfico de linhas com espaço vocálico de paraibanos e paulistanos com eixos x e y revertidos e `geom_label()`. Fonte: própria.

As duas últimas linhas desse comando já são conhecidas por você: a função `ggtitle()` insere um título na figura e a função `labs()` permite definir o nome dos eixos. Já deixei esses textos prontos. Basta descomentar as linhas (apagar o #) e rodá-las (Figura 7.4).

```
ggplot(medias, aes(x = media_F2, y = media_F1, color = AMOSTRA, label
= VOGAL)) +
  geom_line() +
  geom_label() +
  scale_x_reverse() +
  scale_y_reverse() +
  ggtitle("Valores médios de F1 e F2 normalizados nas amostras PBSP e
SP2010") +
  labs(x = "F2 normalizado", y = "F1 normalizado") +
  theme_bw()
```

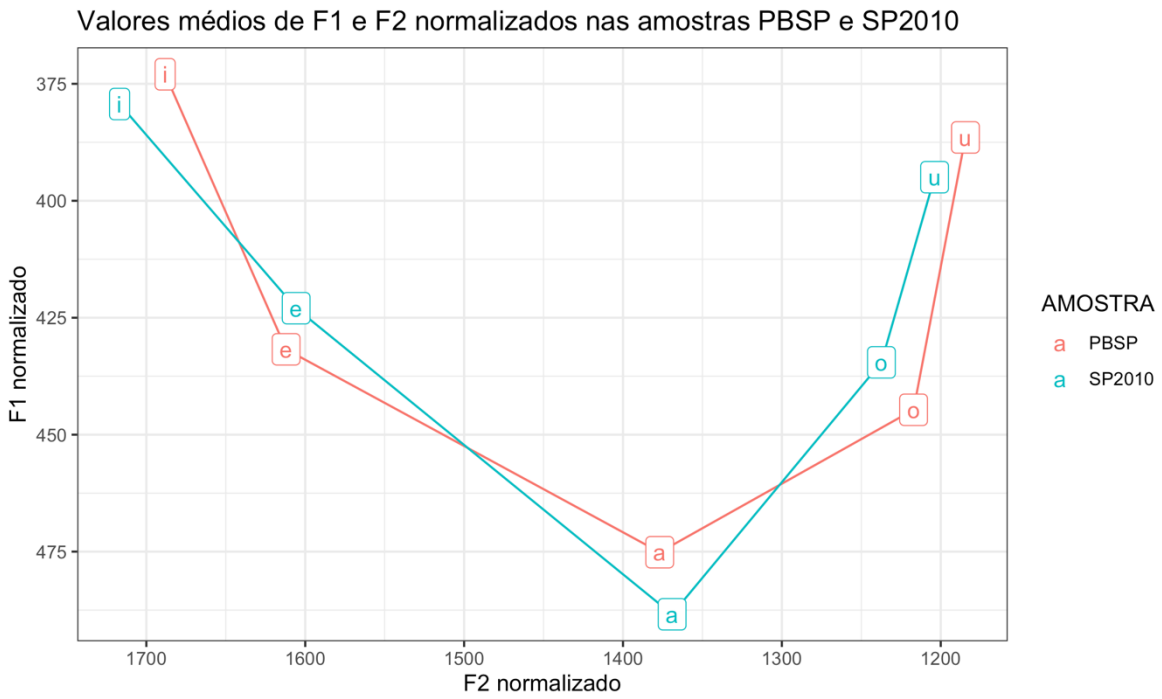

Figura 7.4: Gráfico de linhas com espaço vocálico de paraibanos e paulistanos com eixos x e y revertidos, `geom_label()`, título e rótulos dos eixos. Fonte: própria.

Na última lição, já havíamos visto que as médias de altura (F1) das vogais /e/ e /o/ dos parabainos têm valores mais altos – i.e., tendem a ser realizações mais baixas – do que as vogais dos paulistanos. Essa figura, no entanto, é capaz de informar também a posição relativa quanto ao traço [+- anterior]. Para mais bem visualizá-las, clique sobre Zoom na aba Plots. Caso queira exportar o gráfico, lembre-se das funções `png()` e `dev.off()`, que vimos na Lição 5.

Nesse gráfico que plotamos, as milhares de vogais dos participantes paraibanos e paulistanos foram resumidas em poucos pontos, as médias de F1 e F2. Vimos na última lição que, para além das medidas de tendência central, também é importante conhecer a dispersão dos dados. Um gráfico de dispersão nos permite visualizar não só pontos isolados, mas também a distribuição de todos os dados da amostra.

Neste próximo comando, não há linhas comentadas, pois você já conhece a maior parte das funções: `ggplot()` – que define o dataframe e os parâmetros gráficos, `geom_point()` – que plota pontos, `scale_x_reverse()` e `scale_y_reverse()` – que

invertem os eixos x e y, `ggtitle()` – que inclui um título, `labs()` – que inclui rótulos para os eixos, e `theme_bw()` – que define o tema visual.

```
# Não rodar! Estrutura do código:

ggplot(df, aes(x = VAR, y = VAR, color = VAR)) +
  geom_point() +
  scale_x_reverse() +
  scale_y_reverse() +
  facet_grid(. ~ VAR) +
  ggtitle("Dispersão das medidas de F1 e F2 normalizados nas amostras
PBSP e SP2010") +
  labs(x = "F2 normalizado", y = "F1 normalizado") +
  theme_bw()
```

A única função nova é `facet_grid()`. Essa função permite criar vários subgráficos semelhantes, de acordo com algum critério, que serão dispostos num grid. Aqui, vamos criar dois gráficos de dispersão, um para a amostra PBSP, e outro para SP2010.

Nesse comando, só falta definir os argumentos de `ggplot()`. Vamos plotar a localização (medidas de F1 e F2) de todas as vogais da planilha original, de modo que o dataframe será `pretonicas`. O eixo x será ocupado pelas medidas de F2.NORM, e o eixo y, pelas medidas de F1.NORM. Para mais bem visualizar os níveis de VOGAL, vamos plotá-las cada uma com uma cor. Em `facet_grid()`, defina a variável AMOSTRA. Substitua então os termos df e VAR na linha de comando pelos dados relevantes, revise o código e rode com CTRL + ENTER. O resultado se encontra na Figura 7.5.

```
ggplot(pretonicas, aes(x = F2.NORM, y = F1.NORM, color = VOGAL)) +
  geom_point() +
  scale_x_reverse() +
  scale_y_reverse() +
  facet_grid(. ~ AMOSTRA) +
  ggtitle("Dispersão das medidas de F1 e F2 normalizados nas amostras
PBSP e SP2010") +
  labs(x = "F2 normalizado", y = "F1 normalizado") +
  theme_bw()
```

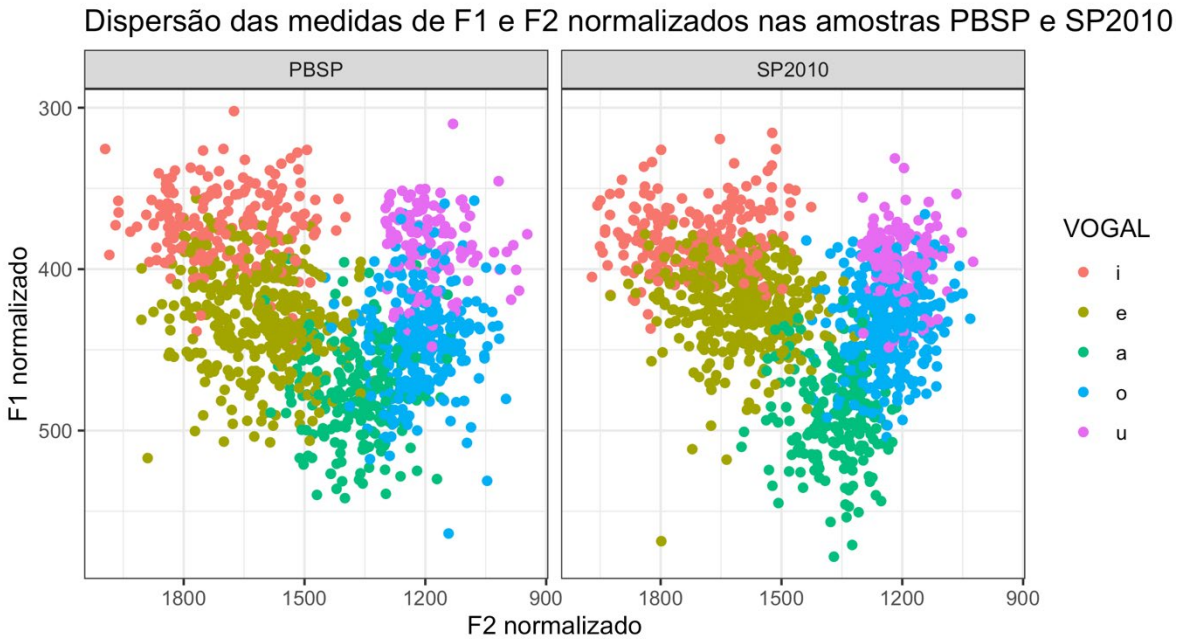

Figura 7.5: Gráfico de dispersão com ggplot. Fonte: própria.

Não ficou bacana? São várias observações que podem ser feitas sobre esse gráfico como, por exemplo, a maior dispersão das vogais /e/ e /o/ dos migrantes paraibanos do que aquelas dos paulistanos (talvez, justamente, pela situação de migração e contato com outro dialeto?...)

Vamos ver agora outro tipo de gráfico para variáveis numéricas, o boxplot. Antes de plotarmos um, vamos ver quais são suas características. Na Figura 7.6 está um esquema do que representa cada elemento de um boxplot.

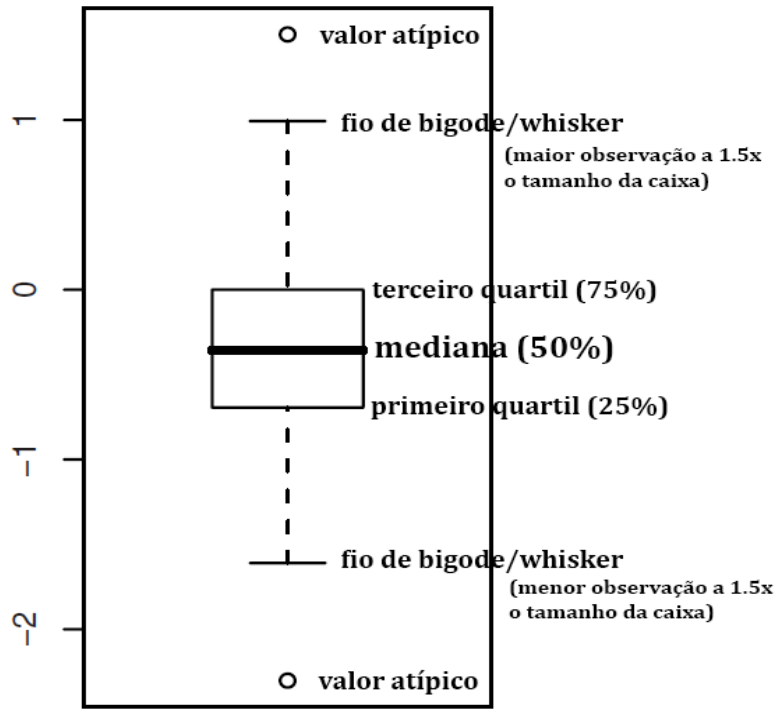

Figura 7.6: Estrutura do boxplot. Fonte: própria.

Comecemos pelo meio. A mediana é normalmente representada por uma linha mais escura e, como você já sabe, ela é a observação central quando se colocam os dados numa ordem crescente. Semelhantes à mediana são as medidas de primeiro quartil e terceiro quartil; mas, diferentemente da mediana (que representa o ponto em 50% da distribuição), eles indicam, respectivamente, os valores em 25% e 75% da distribuição.

Em seguida temos os whiskers, ou fios de bigode. Há um para cima e outro para baixo da "caixa" que forma o meio da distribuição. Eles são calculados tendo o primeiro e o terceiro quartil como referências. Subtrai-se o valor do primeiro quartil do terceiro, o que dá a extensão da caixa. Esse valor é multiplicado por 1,5, o que dará a extensão aproximada do fio do bigode para cima e para baixo.

Por fim, qualquer valor para além dos fios de bigode, para cima e para baixo, é considerado *outlier*, ou valor atípico. Desse modo, o boxplot também permite visualizar a distribuição e a dispersão dos dados.

No boxplot que vamos plotar, vamos comparar a distribuição de dados das 5 vogais para as 2 amostras. Aqui, assim como no gráfico de dispersão, não há linhas

comentadas, porque você já é capaz de interpretá-las. Falta substituir devidamente os valores df e VAR.

```
# Não rodar! Estrutura do código:

ggplot(df, aes(x = VAR, y = VAR, color = VAR)) +
  geom_boxplot(notch = FALSE) +
  scale_y_reverse() +
  labs(x = "Amostra", y = "F1 normalizado") +
  facet_grid(. ~ VAR) +
  theme_bw()
```

Desta vez, vamos fazer um pouco diferente. Veja a Figura 7.7: este deve ser o resultado de seu código! A partir dela, determine o que deve entrar em df, quais variáveis definem os parâmetros estéticos de x, y e color, e qual variável define as facetas.

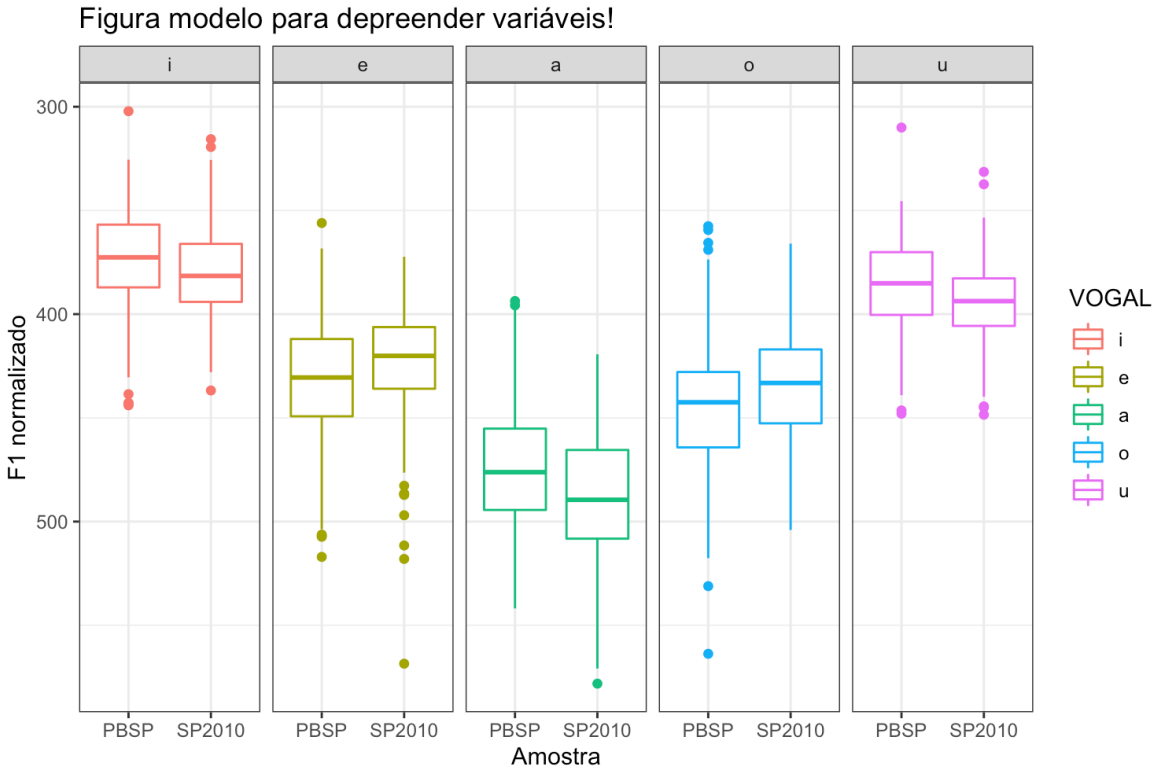

Figura 7.7: Boxplot a ser plotado. Fonte: própria.

Qual dataframe compreende os dados de interesse?

- `df`
- `medias`
- `pretonicas`

Qual variável ocupa o eixo x?

- AMOSTRA
- PBSP
- SP2010

Qual variável ocupa o eixo y?

- F1
- F1 normalizado
- F1.NORM

Qual variável define as cores dos boxplots?

- AMOSTRA
- F1.NORM
- VOGAL

Qual variável define o grid de facetas?

- AMOSTRA
- F1.NORM
- VOGAL

De posse dessas informações, substitua os termos `df` e `VAR` devidamente no *script*, revise o código e rode com CTRL + ENTER. O resultado deve ser o mesmo da Figura 7.7.

```
ggplot(pretonicas, aes(x = AMOSTRA, y = F1.NORM, color = VOGAL)) +
  geom_boxplot(notch = FALSE) +
  scale_y_reverse() +
  labs(x = "Amostra", y = "F1 normalizado") +
  facet_grid(. ~ VOGAL) +
  theme_bw()
```

Esse exercício de visualizar o gráfico que se quer plotar e, a partir dele, determinar o código, é algo que deve acontecer com frequência na prática. Muitas vezes, temos uma figura como modelo, e queremos reproduzi-la em nossos dados; em outros momentos, você pode simplesmente imaginar o gráfico que quer plotar, e o desafio é traduzi-lo em R!

Você pode estar se perguntando o que faz o argumento notch = FALSE em `geom_boxplot()`. Para descobrir, mude-o para TRUE e rode o comando (Figura 7.8).

```
ggplot(pretonicas, aes(x = AMOSTRA, y = F1.NORM, color = VOGAL)) +
  geom_boxplot(notch = TRUE) +
  scale_y_reverse() +
  labs(x = "Amostra", y = "F1 normalizado") +
  facet_grid(. ~ VOGAL) +
  theme_bw()
```

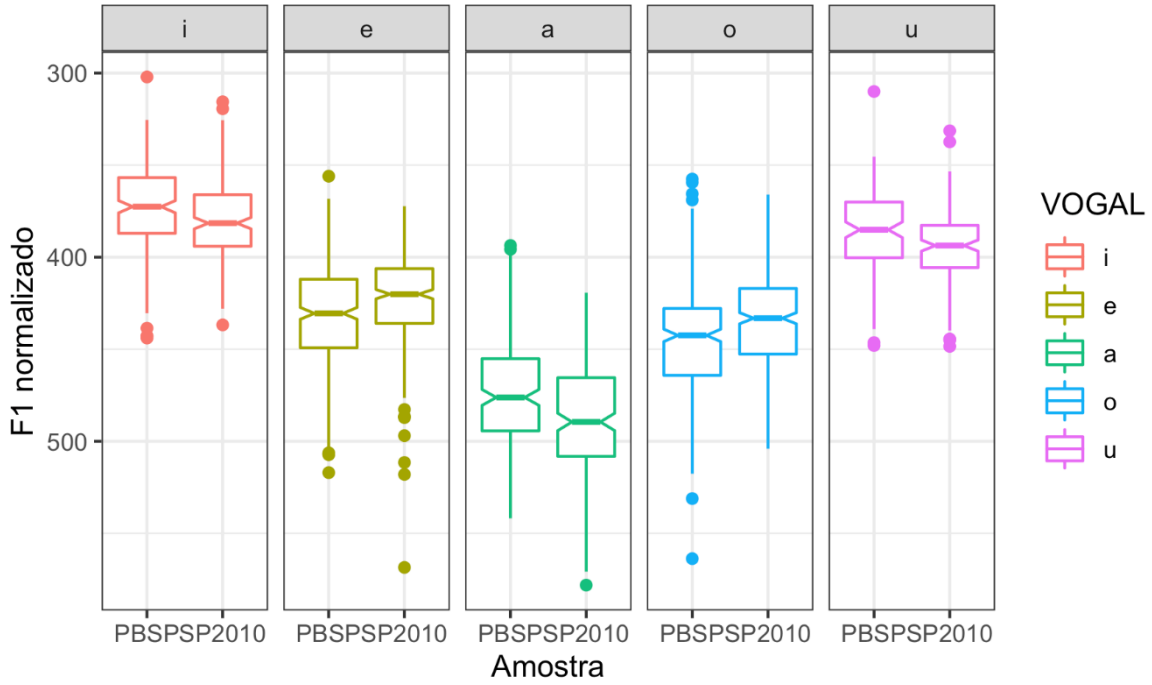

Figura 7.8: Boxplot com notch = TRUE. Fonte: própria.

O argumento notch estabelece se queremos ou não plotar um entalhe na altura da mediana, por meio de um valor lógico (T ou TRUE para verdadeiro ou F ou FALSE para falso). Mas ele não é apenas uma questão estética.

Na figura, vemos claramente que a vogal /a/ é a mais baixa e que as vogais /i/ e /u/ são mais altas. Vemos também que os paulistanos tendem a realizar as vogais /i/, /a/ e /u/ pretônicas como mais baixas que os paraibanos, mas que as vogais médias /e/ e /o/ são mais altas. No entanto, há também uma boa sobreposição entre os intervalos de cada vogal quando comparamos as duas amostras. Não é o caso que os paulistanos sempre realizam /e/ e /o/ pretônicas mais altos e os paraibanos tenham realizações sempre mais baixas. Pelos fios de bigode, vemos também que às vezes um /e/ pode ser tão alto quanto um /i/, ou tão baixo quanto um /a/.

Uma questão que se coloca, ao observar os boxplots, é se essas diferenças na altura das vogais são reais, ou se se devem simplesmente à distribuição dos dados. Mais especificamente, podemos nos perguntar: as vogais /e/ e /o/ dos paraibanos que vivem em São Paulo são realmente mais baixas do que as dos paulistanos? Para responder a essa questão, vamos precisar de ferramentas além da estatística descritiva (que temos implementado por meio de tabelas e gráficos). Vamos precisar de *testes estatísticos* e de ferramentas da *estatística inferencial*.

Os entalhes dos boxplots, no entanto, já são uma primeira pista para saber se podemos afirmar que as vogais /e/ e /o/ de paraibanos são em média mais baixas do que as dos paulistanos. Elas representam um intervalo de confiança – algo que veremos com mais detalhes na próxima lição. Por ora, basta indicar que, quando os entalhes não se sobrepõem, é provável que a diferença entre as amostras seja estatisticamente significativa.

Vamos, por fim, ver outro tipo de gráfico que se aplica a variáveis numéricas: o histograma. Quando tratamos de variáveis nominais, vimos que frequências são a medida de quantas vezes algo aconteceu numa amostra. O histograma "quebra" uma variável numérica em intervalos (p.ex., de 200 a 300, de 301 a 400 etc.) e representa graficamente a frequência dentro de cada intervalo. Isso permite visualizar em qual ou quais intervalos se concentram os dados.

Veja o código para plotar um histograma no *script*, no qual aproveito para revisar o pipe, do tidyverse: `%>%` permite encadear diversas operações – pegar o resultado da operação à esquerda e usá-lo no comando seguinte.

```
# Não rodar! Estrutura do código:

pretonicas %>%
  filter(VOGAL == "e" & AMOSTRA == "PBSP") %>%
  ggplot(., aes(x = VAR)) +
  geom_histogram(binwidth = n, fill = "white", color = "black") +
  labs(x = "F1 normalizado", y = "Frequência") +
  theme_bw()
```

Vamos plotar um histograma das medições de F1.NORM da vogal /e/ na amostra dos paraibanos. No código, primeiro pegamos o dataframe `pretonicas` e, com

`filter()`, criamos um subconjunto de dados da vogal /e/ na amostra PBSP. Este novo subconjunto de dados (sem nome!) é o dataframe utilizado para fazer o gráfico.

Para plotar um histograma, precisamos apenas de uma variável numérica. Vamos usar F1.NORM novamente. Na geometria `geom_histogram()`, podemos definir o tamanho dos intervalos (binwidth) – se de 5 em 5, 10 em 10 unidades etc.; com qual cor preencher (fill) as barras e com qual cor definir o contorno das barras (color). No código, inclua a variável F1.NORM e o número 20 para o tamanho dos intervalos de cada barra. O resultado se encontra na Figura 7.9.

```
pretonicas %>%
  filter(VOGAL == "e" & AMOSTRA == "PBSP") %>%
  ggplot(., aes(x = F1.NORM)) +
  geom_histogram(binwidth = 20, fill = "white", color = "black") +
  labs(x = "F1 normalizado", y = "Frequência") +
  theme_bw()
```

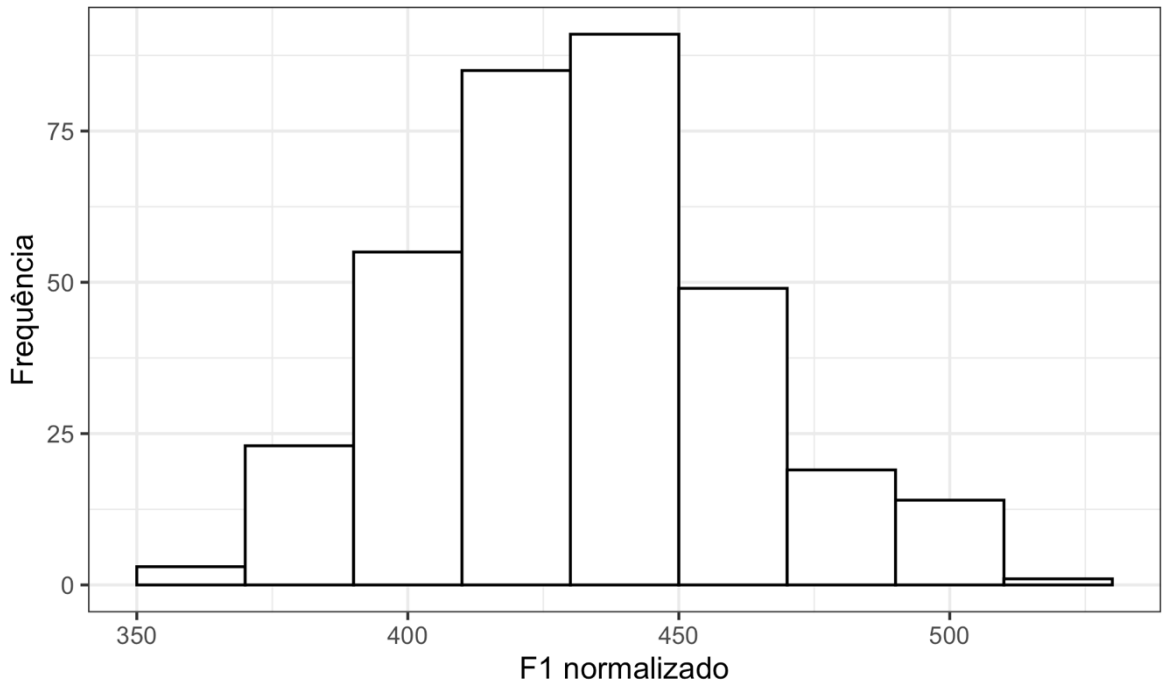

Figura 7.9: Histograma com ggplot. Fonte: própria.

Um histograma se parece com um gráfico de barras, mas não devemos confundi-los. Vimos que gráficos de barras se aplicam a variáveis nominais (como as lojas de departamento em Nova Iorque). Um histograma, diferentemente, se aplica a variáveis

contínuas, de modo que as barras são representadas no contínuo do eixo x. São informações diferentes.

O que o histograma nos informa? Ele mostra como se *distribuem* e se *dispersam* os dados da variável numérica. Nesse sentido, o histograma é uma representação visual das medidas de desvio padrão / variância, bem como das medidas de tendência central (média, mediana, moda). Quanto menos dispersos os dados, maior será sua concentração em torno de um ponto médio.

A figura mostra que a maior parte das medições de F1.NORM da vogal /e/ de paraibanos se concentra entre 410 e 430 Hz e entre 430 e 450 Hz – as duas barras com maior número de ocorrências (frequência). Os intervalos são de 20 Hz porque assim definimos com binwidth acima. Também houve ocorrências de /e/ com medidas acima e abaixo desses valores, mas eles foram menos frequentes.

Vamos checar como os dados se dispersam em torno da média e da mediana. O comando para computar a média e a mediana já está pronto neste ponto do *script*. Basta rodá-lo – mas certifique-se de que entende o que está sendo feito aí!

```
med_centrais_PBSPe <- pretonicas %>%
  filter(., VOGAL == "e" & AMOSTRA == "PBSP") %>%
  summarize(media = mean(F1.NORM),
            mediana = median(F1.NORM)) %>%
  print()
```

```
## # A tibble: 1 × 2
##   media mediana
##   <dbl>   <dbl>
## 1  432.    431.
```

Poderíamos inspecionar os valores de `media` e `mediana` no Console. No entanto, melhor do que isso, é visualizá-los! Vamos plotar, sobre o histograma, duas linhas que indicam esses pontos.

A linha de comando aqui reproduz o mesmo histograma que acabamos de criar, mas acrescenta mais duas linhas com a função `geom_vline()` que, como o nome já indica, plota uma linha vertical. Dentro dessa função, especificamos em qual ponto a linha vertical deve cortar o eixo x (xintercept): respectivamente, nos valores da média e

da mediana do dataframe que acabamos de computar. A média será representada por uma linha azul e a mediana por uma linha vermelha (Figura 7.10).

```
pretonicas %>%
  filter(., VOGAL == "e" & AMOSTRA == "PBSP") %>%
  ggplot(., aes(x = F1.NORM)) +
  geom_histogram(binwidth = 20, fill = "white", color = "black") +
  labs(x = "F1 normalizado", y = "Frequência") +
  theme_bw() +
  geom_vline(xintercept = med_centrais_PBSPe$media, color = "blue") +
  geom_vline(xintercept = med_centrais_PBSPe$mediana, color = "red")
```

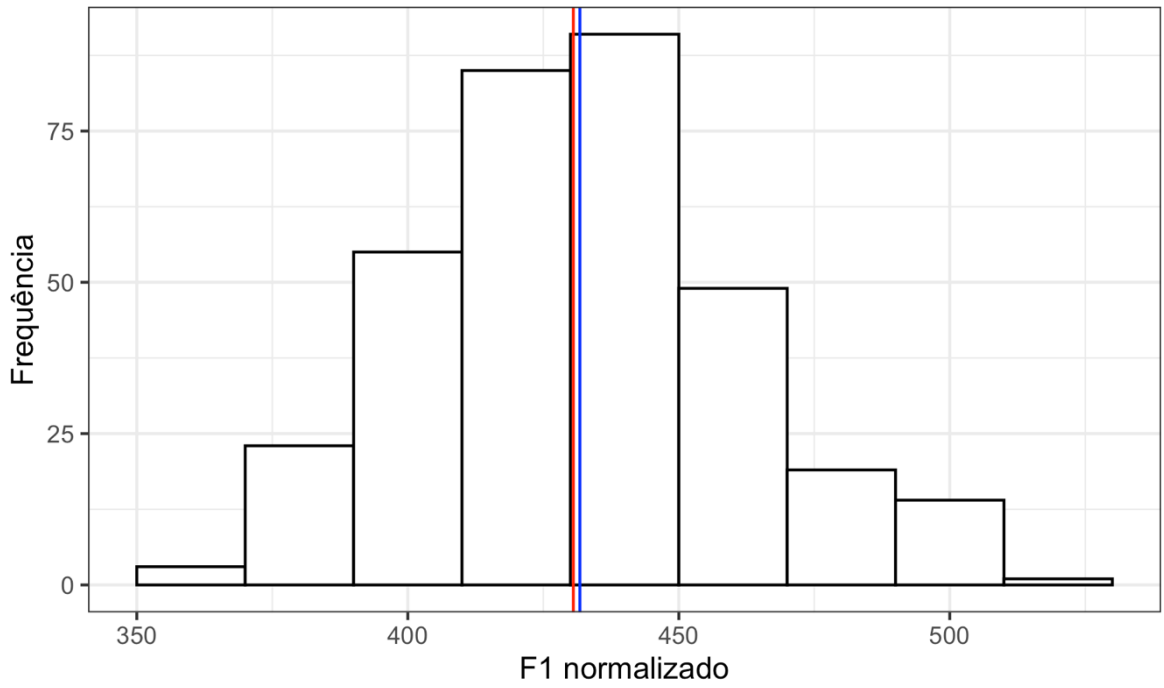

Figura 7.10: Histograma com indicação dos valores de média e de mediana. Fonte: própria.

Vemos que a média e a mediana são praticamente coincidentes, próximas ao topo da distribuição. Você se lembra da Figura 6.1, vista na lição passada, que representa modelos idealizados da distribuição e dispersão dos dados? Nosso histograma representa uma distribuição real.

Com qual das figuras você acha que nosso histograma mais se parece?

- com a figura do meio, uma distribuição normal
- com a figura à esquerda, com viés negativo
- com a figura à direita, com viés positivo

Nosso histograma se parece com uma distribuição normal e, na verdade, isso não é surpreendente. Temos usado os dados de F1 e de F2 *normalizados* de acordo com o método de Lobanov. O que a normalização faz é justamente isso: transformar os dados em uma distribuição que se aproxima da distribuição normal.

Você já deve ter percebido que o termo "distribuição normal", aqui, é o um termo técnico. Ela se refere a dados que se distribuem na forma de uma curva de sino. Veremos as propriedades dessa distribuição com mais detalhes na próxima lição.

Para saber mais

Na Lição 5 e nesta, vimos apenas alguns poucos tipos de gráficos: de barras, de linhas, histogramas, dispersão e boxplots. Mas existem vários outros que você deve explorar. No site The R Graph Gallery (http://www.r-graph-gallery.com/), há uma enorme coleção de gráficos produzidos no R, com seus respectivos códigos. Outros tipos de gráficos podem mais bem expressar as relações que você quer mostrar.

Além disso, recomendo a consulta constante ao livro "R Graphics Cookbook", de Winston Chang. Trata-se realmente de um livro de receitas, que apresenta "problemas" – o que você quer fazer – e o passo a passo para solucioná-los. Uma versão dele se encontra *on-line* em https://r-graphics.org/.

A mensagem aqui é: não deixe as ferramentas conduzirem sua análise! Você, como pesquisador ou pesquisadora, deve estabelecer o que é mais adequado para os seus dados.

Exercícios

Nesta lista de exercícios, você vai precisar do dataframe `pretonicas`, que foi usado na lição. Siga os comandos da lição caso precise recarregá-lo no R.

1. Nesta lista, você vai precisar dos pacotes tidyverse e RColorBrewer. Carregue-os.
2. Nesta lição, fizemos um histograma da distribuição das medições de F1 normalizado da vogal /e/ dos dados da amostra PBSP. Crie um novo histograma

semelhante, mas agora com os dados de F1 (i.e., sem normalização.). Para tanto, com auxílio do pipe `%>%` e a partir do dataframe `pretonicas`, (i) crie um subconjunto apenas dos dados da vogal "e" na amostra PBSP; (ii) determine o parâmetro estético para o eixo x; (iii) use a geometria de histograma com bindwidth = 30, com barras "lightblue" e bordas das barras "gray"; (iv) nomeie o eixo x como "F1" e o eixo y como "Frequência"; (v) intitule o gráfico "Medidas de F1 da vogal /e/ da amostra PBSP"; e (vi) use o tema `theme_minimal()`.

3. Reproduza o gráfico acima para a vogal /o/ dos paraibanos. Modifique apenas o estritamente necessário!
4. Compare os dois gráficos que acabou de plotar. Você pode voltar a gráficos anteriores ou posteriores clicando sobre a flecha para a esquerda e para a direita no topo da aba Plots. Pelo histograma, qual das vogais deve ter maiores valores de média e mediana? Justifique sua resposta.
5. Determine os valores de média e mediana das vogais /e/ e /o/ dos paraibanos. Para tanto, com auxílio do pipe e a partir do dataframe `pretonicas`, (i) crie um subconjunto dos dados das vogais "e" e "o" na amostra "PBSP"; (ii) agrupe os dados por VOGAL; e (iii) calcule as medidas de média e de mediana, nomeando as colunas media_F1 e mediana_F1 respectivamente.
6. Crie dois histogramas da distribuição dos dados da vogal /e/, um para cada amostra (PBSP e SP2010), cada qual com uma cor diferente. Para tanto, com auxílio do pipe e a partir do dataframe `pretonicas`, (i) crie um subconjunto dos dados da vogal /e/; (ii) defina os parâmetros gráficos para x e para fill dentro da função `ggplot()`; (iii) use a geometria de histograma com binwidth = 20 e alpha = 0.4; (iv) nomeie o eixo x como "F1" e o eixo y como "Frequência"; (v) use a função `facet_grid()` para que os histogramas de cada amostra sejam dispostos um sobre o outro; (vi) use a paleta "Pastel2" do RColorBrewer; e (vii) aplique o tema `theme_minimal()`.
7. Faça boxplots dos dados de F1 da vogal /e/ por PARTICIPANTE, diferenciando os falantes da amostra PBSP e SP2010 com cores distintas. Para tanto, com auxílio

do pipe e a partir do dataframe pretonicas, (i) crie um subconjunto de dados da vogal “e”; (ii) defina os parâmetros gráficos x, y e color dentro da função `ggplot()`; (iii) use a geometria para boxplots sem o notch; (iv) inverta a escala dos valores do eixo y (para que os valores de F1 sejam representados do modo fonético convencional); (v) nomeie o eixo x como “Falante”, o eixo y como “F1”, e a variável da legenda como “Amostra”; (vi) intitule o gráfico como “Medidas de F1 da vogal /e/ por falantes das amostras PBSP e SP2010”; e (vii) use o tema `theme_light()`. Nota: se você achar que os nomes dos falantes não estão bem dispostos na figura, veja-a com Zoom, pois os nomes não estão de fato encavalados!

8. Para comparar, reproduza o mesmo gráfico acima para os dados de F1 da vogal /e/ com normalização (F1.NORM). Faça modificações (i) no parâmetro estético relevante; (ii) no rótulo do eixo y, de “F1” para “F1 normalizado”; e (iii) no título do gráfico, trocando igualmente “F1” por “F1 normalizado”.
9. Comparando os gráficos, em qual deles há maior dispersão das medições da altura da vogal /e/: com F1 ou com F1.NORM? Explique sua resposta.
10. Determine a diferença da dispersão das medidas da altura da vogal /e/ numericamente. Para tanto, com auxílio do pipe e a partir do dataframe `pretonicas`, (i) crie um subconjunto de dados da vogal “e”; e (ii) compute as médias de desvio padrão de F1 e de F1.NORM, nomeando as colunas como sd_F1 e sd_F1.NORM respectivamente.
11. Nesta lição, plotamos um gráfico de dispersão que representa todas as vogais “i”, “e”, “a”, “o” e “u” de paraibanos migrantes e paulistanos, e em que cada vogal é identificada por pontos de cores distintas. No ggplot2, também é possível representar cada ponto por um caractere específico – p.ex., as próprias vogais – por meio da geometria `geom_text()`. Façamos então um tal gráfico. Para tanto, com auxílio do pipe e a partir do dataframe `pretonicas`, (i) defina os parâmetros gráficos x, y e color dentro da função `ggplot()`, de modo que o eixo x represente o traço [+- anterior] de vogais normalizadas, o eixo y represente o traço [+- alto]

de vogais normalizadas e cada vogal tenha uma cor diferente; (ii) use a geometria `geom_text()` com o argumento aes(label = VOGAL); (iii/iv) inverta a escala dos eixos x e y, de modo que a representação do espaço vocálico siga a convenção fonética; (v) use a função `facet_grid()` de modo que sejam plotados os espaços vocálicos separados por AMOSTRA, dispostas lado a lado; (vi) intitule o gráfico "Dispersão das medidas de F1 e F2 normalizados nas amostras PBSP e SP2010"; (vii) nomeie o eixo x como "F2 normalizado" e o eixo y como "F1 normalizado"; e (viii) adicione theme(legend.position = "none") ao final, para que não seja plotada a legenda, que agora não é mais necessária.

12. Nesta lição, fizemos um gráfico de linhas que representa os espaços vocálicos de paraibanos migrantes e paulistanos, com as médias de F1 e F2 normalizados. Você vai fazer agora um gráfico semelhante, mas com as medidas das vogais de F1 e F2 não normalizados. Para tanto, faça primeiro, com auxílio do pipe e a partir do dataframe `pretonicas`, um dataframe que computa as médias de F1 e de F2 para cada vogal e para cada amostra: (i) guarde o resultado em dataframe chamado `medias`; (ii) agrupe os dados por VOGAL e AMOSTRA; (iii) calcule as médias de F1 e de F2, nomeando as colunas como media_F1 e media_F2 respectivamente; (iv) visualize o resultado em tela com `print()`.
13. Com auxílio do pipe e a partir do dataframe `medias`, (i) defina os parâmetros x, y, color e label dentro da função `ggplot()`; (ii/iii) empregue as geometrias de linha e de rótulo (label); (iv/v) inverta as escalas dos eixos x e y, de modo que os espaços vocálicos sejam representados da maneira convencional em Fonética; (vi) intitule o gráfico como "Valores médios de F1 e F2 nas amostras PBSP e SP2010"; (vii) nomeie o eixo x como "F2 (Hz)", o eixo y como "F1 (Hz)" e a variável da legenda como "Amostra"; (viii) use a paleta "Paired" do pacote RColorBrewer na função `scale_color_brewer()`; e (ix) use o tema `theme_classic()`.
14. Reproduza o gráfico da Figura 7.11, que foi feito a partir dos dados de pretonicas. Ele consiste nos espaços vocálicos de cada um dos sete falantes paraibanos, separados por seu gênero. Atente-se a todos os detalhes (nomes de eixos, cores,

título, tema etc.). Como mencionado na lição, um bom modo de aprimorar a representação gráfica de dados é tentar reproduzir gráficos vistos em outras fontes, ou mesmo visualizar um gráfico em sua mente e tentar fazê-lo no ggplot2.

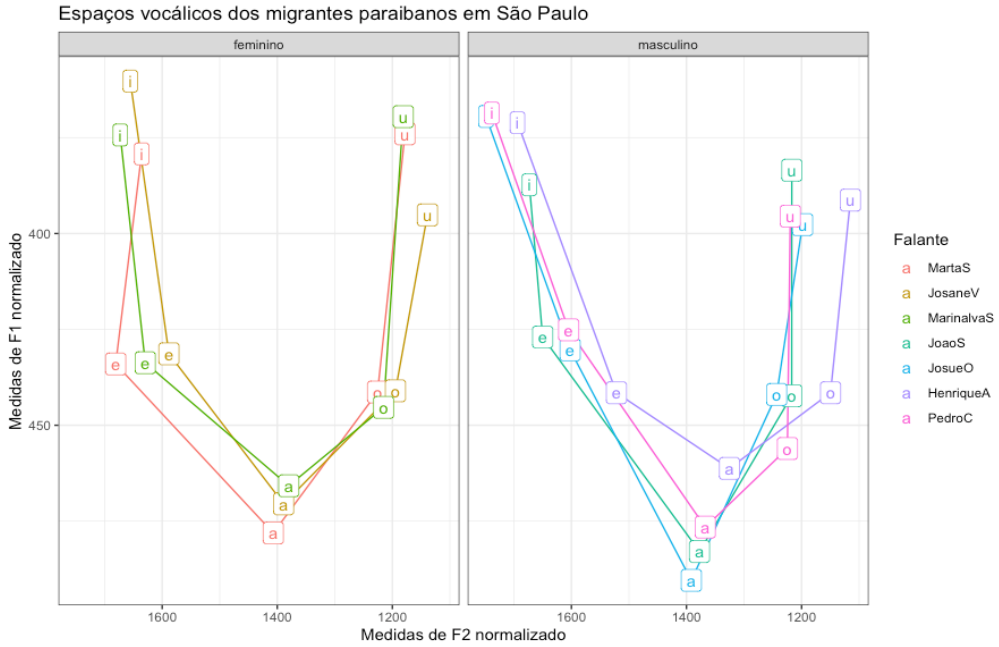

Figura 7.11: Gráfico para reprodução no ggplot2. Fonte: própria.

Lição 8: Conceitos Básicos de Estatística Inferencial

Nesta lição, vamos tratar de alguns conceitos centrais da Estatística Inferencial: *hipótese alternativa* e *hipótese nula*; *distribuição normal*; *significância*; *erros do Tipo I* e do *Tipo II*; e *intervalo de confiança*.

Até agora, fizemos tabelas e gráficos de dados de variáveis nominais, ordinais e numéricas. Reproduzo um desses gráficos na Figura 8.1: a proporção de apagamento de /r/ pós-vocálico em lojas de departamento de Nova Iorque na década de 1960, com base nos dados de Labov (1972). Vimos que houve menos r0 entre os funcionários da Saks, um pouco mais na Macy's, e mais ainda na S. Klein. Em nenhuma loja houve 0% ou 100% da variante "apagamento de /r/".

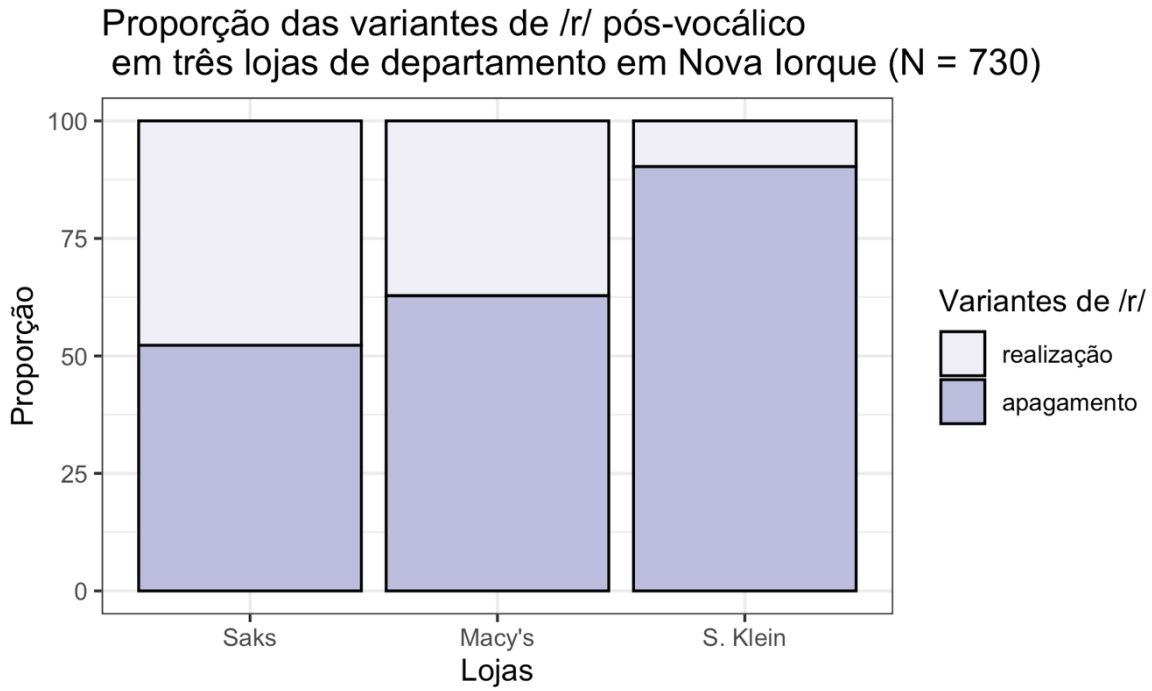

Figura 8.1: Gráfico de barras feito na Lição 5. Fonte: própria.

Na Figura 8.2 está outro gráfico que plotamos anteriormente: a distribuição das vogais pretônicas nas amostras de falantes paraibanos e paulistanos residentes em São

Paulo. Vemos que há diferenças entre as vogais e entre os dois grupos de falantes, mas que também há muita sobreposição no intervalo de F1 para cada vogal. Há dados, por exemplo, de /o/ cujas medidas de F1 são tão baixas quanto as de vogais /a/.

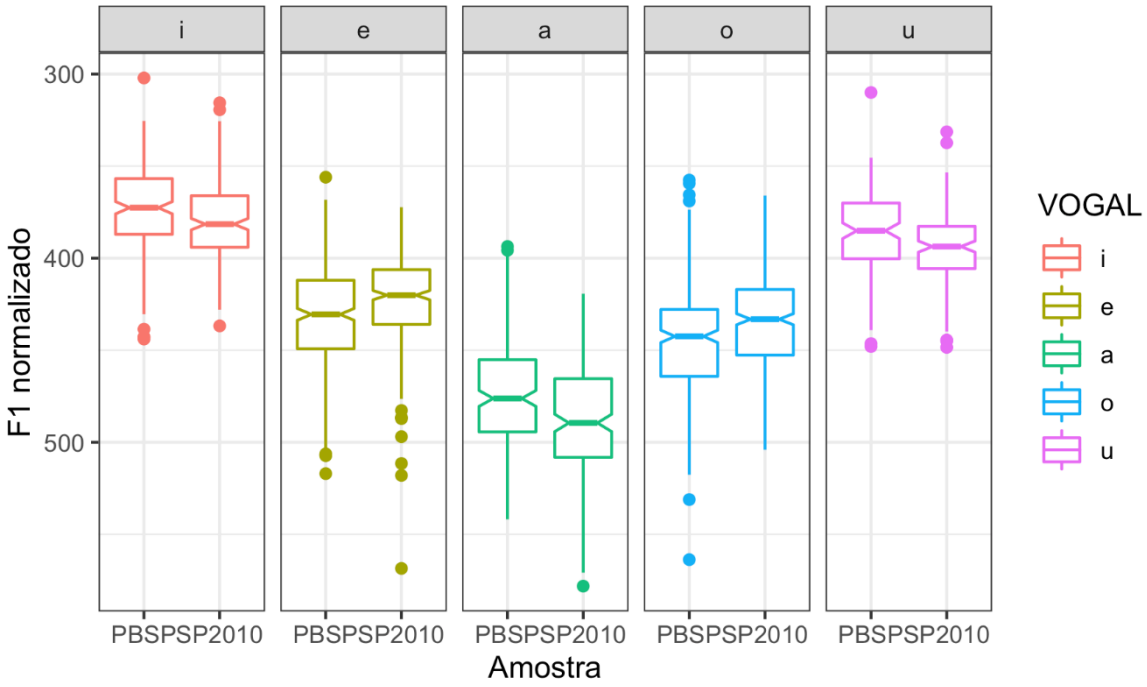

Figura 8.2: Boxplot feito na Lição 7. Fonte: própria.

Quando se lida com dados reais, inevitavelmente se depara com a variação. Há uma série de fenômenos no mundo – e na língua! – que não são categóricos. E, quando se trata de variação, sempre há um grau de incerteza. Quem vai ganhar a próxima eleição? Será que hoje vai chover?

O fato de que há variação no mundo, contudo, não implica que tudo seja caótico, regido pelo puro acaso e aleatoriedade. Mesmo nos casos em que determinado resultado não seja 100% certo, podemos estimar *probabilidades*. As pesquisas de intenção de voto antes de uma eleição e a previsão diária do tempo são exemplos disso.

Nos dados de vogais pretônicas, vemos que as vogais médias /e/ e /o/ são mais baixas na fala de paraibanos do que na de paulistanos. Mas com tanta variação e sobreposição de medições, uma pergunta que se coloca é: será que isso aconteceu por acaso? Será que, em outro conjunto de dados, a distribuição de F1 dessas vogais poderia

ter sido igual, ou quiçá se observassem vogais mais baixas entre paulistanos? Uma primeira resposta para essas perguntas é: sim, isso é *possível*. Mas quão *provável* isso é?

Em pesquisas científicas, levantamos hipóteses a serem testadas. Para que hipóteses possam ser colocadas à prova, elas precisam se configurar como uma afirmação que se refere a mais do que um único evento e precisam ser falseáveis (Gries, 2019, p. 22). A primeira dessas características parece ser autoevidente. Se uma hipótese não é generalizável para além de uma única observação, ela é de pouco uso para ser testada.

A falseabilidade diz respeito à própria possibilidade de colocar a hipótese à prova e refutá-la. Por exemplo, a seguinte afirmação pode ser falseada: "Alunos universitários cotistas têm desempenho escolar tão bom quanto ou melhor do que de alunos universitários não cotistas".

A hipótese acima é chamada de H1 ou hipótese alternativa. A ela, opõe-se a H0, ou hipótese nula. A hipótese nula é normalmente formulada a partir da negação de H1. Neste exemplo, H0 seria "Alunos universitários cotistas não têm desempenho escolar tão bom quanto ou melhor do que de alunos universitários não cotistas".

H1 + H0 devem abarcar todos os resultados possíveis. Se algum cenário ou resultado não é previsto nem por H1 nem por H0, a hipótese está mal formulada. Em um levantamento sobre o desempenho escolar de cotistas e não cotistas, só há três resultados possíveis: (i) cotistas têm melhor desempenho do que não cotistas; (ii) cotistas têm desempenho tão bom quanto de não cotistas; (iii) cotistas têm desempenho pior do que não cotistas. Todos esses cenários estão cobertos em H1 + H0. H1 e H0 são formuladas como afirmações categóricas para que sejam falseáveis. Mas, como vimos, a realidade dos dados nem sempre (na verdade, quase nunca) é categórica.

Vejamos um exemplo cotidiano para ver H1 e H0 em ação. O exemplo a seguir é adaptado de Gries (2019, p.34ss). Imagine que decidimos jogar cara e coroa. Eu sou cara e você é coroa, ok? Para tornar as coisas mais interessantes, apostamos dinheiro: toda vez que cair cara, você me dá 5 reais; toda vez que cair coroa, eu te dou 5 reais. Estabelecemos de antemão que a moeda será jogada 100 vezes.

Digamos que, ao final de 100 jogadas, eu tenha ganhado 52 vezes e você 48. Você diria que eu roubei? E se eu tivesse ganhado 54 vezes e você 46? E se eu tivesse ganhado 60 vezes e você 40?

Não há um limite absolutamente exato para decidir que alguém está roubando num jogo da moeda. Mas o ponto aqui é: a depender do resultado, você ia desconfiar seriamente de que eu roubei. Você nem coloca em questão o cenário oposto – você estar roubando, pois você, que é honesto, sabe que não está roubando; a dúvida é apenas sobre seu oponente. Se eu tivesse ganhado 90 vezes e você apenas 10, acho muito difícil que você pensasse: "Puxa, como ela é sortuda!...". E o motivo disso é simples: embora 90 vs. 10 seja um resultado *possível*, ele é muito pouco *provável*.

Pare um pouco para pensar a partir de qual resultado você acharia que estou roubando (55? 60? 62?...). Digite esse número no *script*, atribuindo-o a uma variável chamada x.

```
x <- 60
```

N.B.: No exemplo acima está o número "60", mas você pode escolher qualquer outro entre 51 e 100!

Você está efetivamente estabelecendo uma hipótese: minha oponente está roubando no jogo de moeda! Esta é a hipótese alternativa, H1. A hipótese nula afirma o contrário: minha oponente *não* está roubando.

A beleza na Estatística está no desenvolvimento de métodos para calcular essas probabilidades. Vamos simplificar nosso jogo da moeda de 100 para 3 jogadas para facilitar a demonstração. Em três jogadas de moeda, há no total 8 resultados possíveis, discriminados na Tabela 8.1.

Tabela 8.1: Resultados possíveis para três jogadas de cara ou coroa e suas respectivas probabilidades (H_0 está correta).

Jogada 1	Jogada 2	Jogada 3	# cara	# coroa	$p_{resultado}$
cara	cara	cara	3	0	0,125
cara	cara	coroa	2	1	0,125
cara	coroa	cara	2	1	0,125

cara	coroa	coroa	1	2	0,125
coroa	cara	cara	2	1	0,125
coroa	cara	coroa	1	2	0,125
coroa	coroa	cara	1	2	0,125
coroa	coroa	coroa	0	3	0,125

Fonte: Gries (2019, p.37).

Considerando que a moeda é honesta, não viciada, a probabilidade de cada um dos 8 resultados ocorrer é 1 / 8 = 0,125. Com isso é fácil computar a probabilidade de cada resultado final: a probabilidade de 3 caras e 0 coroas (ou de 0 caras e 3 coroas) é 0,125; a probabilidade de 2 caras e 1 coroa (ou 1 cara e 2 coroas) é 3 * 0,125 = 0,375.

É fácil calcular à mão quais são todos os resultados possíveis de três jogadas de moeda, mas ficaria mais difícil fazer isso em um número maior de jogadas. Para 100 jogadas, o total de possíveis resultados é 2 ^ 100 (em que ^ representa "elevado a"). Faça esse cálculo no R agora.

```
2 ^ 100
```

```
## [1] 1.267651e+30
```

Esse é um número com tantos dígitos que o R fornece o resultado em *notação e*: 1.267651e+30, que equivale a 1,267651 * (10^{30}). Já é difícil fazer sentido desse número – quem dirá calcular as probabilidades de cada um desses possíveis resultados! Mas é claro que o R tem uma função para fazer esse cálculo.

Os resultados de um jogo de cara ou coroa seguem uma distribuição binomial, pois há dois resultados possíveis. Uma função para calcular probabilidades dentro de uma distribuição binomial é `dbinom()`. Em uma forma simples, ela toma como argumentos (i) o número de sucessos de uma variável binária (p.ex., 3 caras); (ii) o número de tentativas; (iii) a probabilidade de se ter aquele resultado em uma tentativa (para a moeda, 50%, ou 0,5). Digite então `dbinom(3, 3, 0.5)` para ver o resultado.

```
dbinom(3, 3, 0.5)
```

```
## [1] 0.125
```

O R informa que a probabilidade de se ter 3 caras em 3 jogadas de moeda, em que a probabilidade de cara é 0.5, é 0,125 – exatamente como calculamos acima. Podemos também pedir que o R calcule a probabilidade de todos os resultados possíveis – de 0 a 3 caras (o que também cobre os resultados de se ter 3 a 0 coroas). Digite `dbinom(0:3, 3, 0.5)` para ver o resultado.

```
dbinom(0:3, 3, 0.5)
```

```
## [1] 0.125 0.375 0.375 0.125
```

O R mostra, em ordem, qual é a probabilidade de se ter 0 cara, 1 cara, 2 caras e 3 caras. Que tal colocar isso num gráfico? Já vimos que é mais fácil compreender elementos gráficos do que textuais. Vamos usar aqui a função `barplot()`, da instalação base do R, para você também conhecê-la (e ver que o resultado é bem mais feinho que os gráficos do ggplot2, mas ele já vai dar conta do recado). Como primeiro argumento, use a função `dbinom()` do modo como especificamos acima. Como segundo argumento, use beside = T, para que as barras fiquem lado a lado. O resultado se encontra na Figura 8.3.

```
barplot(dbinom(0:3, 3, 0.5), beside = T)
```

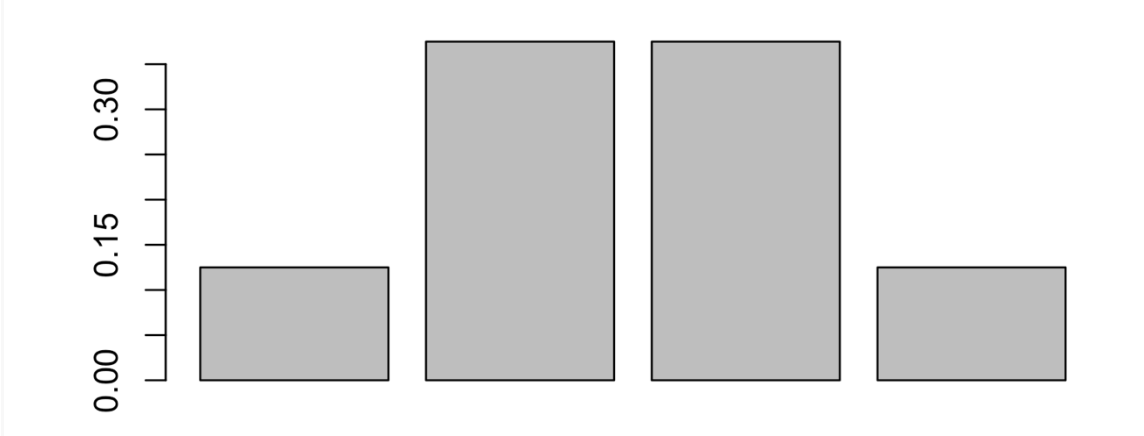

Figura 8.3: Distribuição binomial da probabilidade de se obter 0 a 3 caras em três jogadas de moeda. Fonte: própria.

As barras representam, da esquerda para a direita, os valores 0.125, 0.375, 0.375 e 0.125 computados acima para a probabilidade de 0, 1, 2, e 3 caras respectivamente.

Vamos agora visualizar quais são as probabilidades de se ter de 0 a 6 caras em 6 jogadas de moeda. A partir da linha de comando acima, substitua os dois '3' por '6' para obter `barplot(dbinom(0:6, 6, 0.5), beside = T)`.

```
barplot(dbinom(0:6, 6, 0.5), beside = T)
```

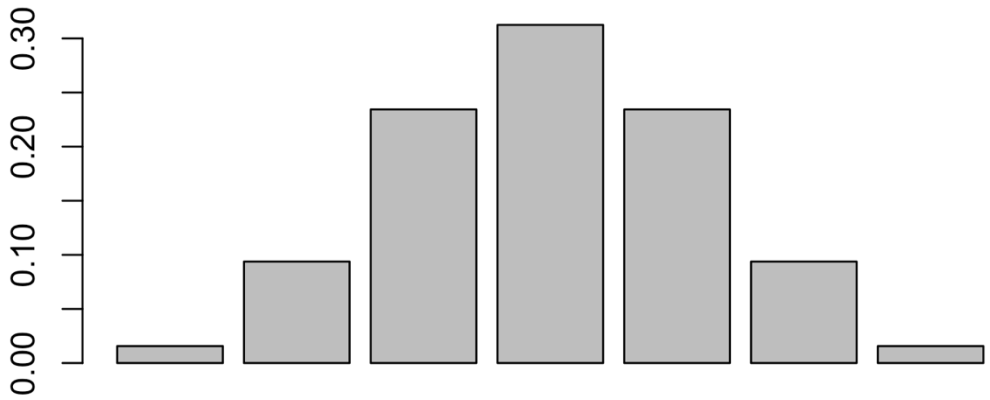

Figura 8.4: Distribuição binomial da probabilidade de se obter 0 a 6 caras em seis jogadas de moeda. Fonte: própria.

Faça o mesmo para 12 jogadas de moeda. A partir da linha de comando acima, substitua '6' por '12'.

```
barplot(dbinom(0:12, 12, 0.5), beside = T)
```

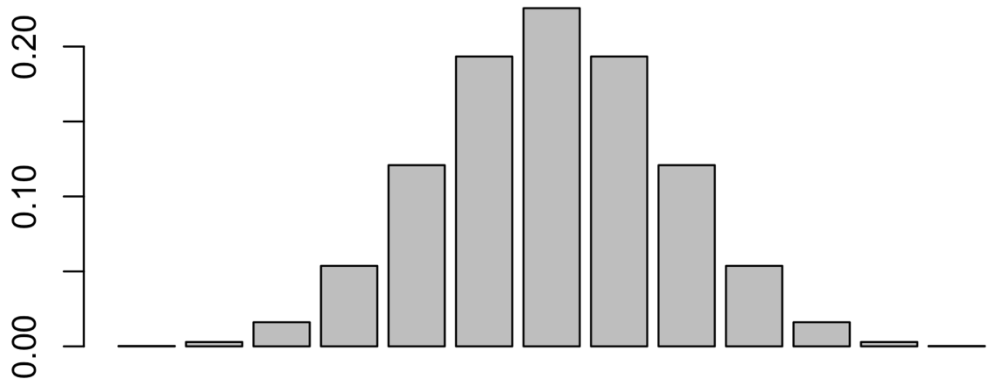

Figura 8.5: Distribuição binomial da probabilidade de se obter 0 a 12 caras em 12 jogadas de moeda. Fonte: própria.

Faça o mesmo para 25 jogadas de moeda para computar todas as probabilidades de 0 a 25 caras.

```
barplot(dbinom(0:25, 25, 0.5), beside = T)
```

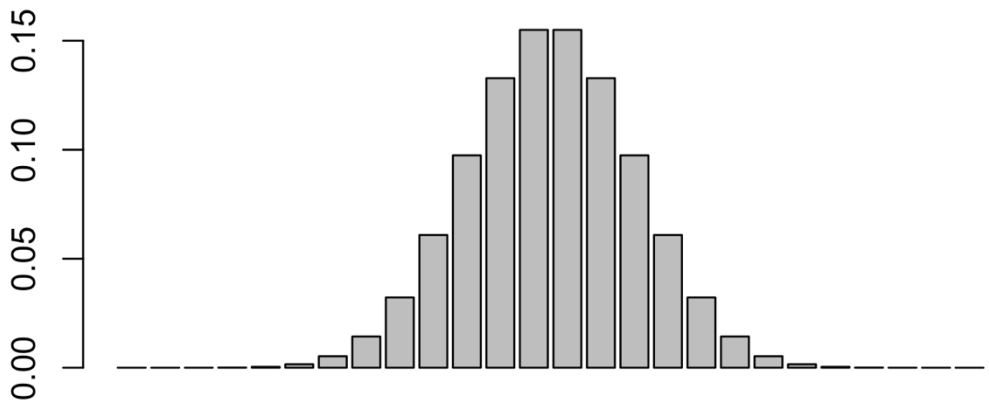

Figura 8.6: Distribuição binomial da probabilidade de se obter 0 a 25 caras em 25 jogadas de moeda. Fonte: própria.

Faça o mesmo para 50 jogadas de moeda para computar todas as probabilidades de 0 a 50 caras.

```
barplot(dbinom(0:50, 50, 0.5), beside = T)
```

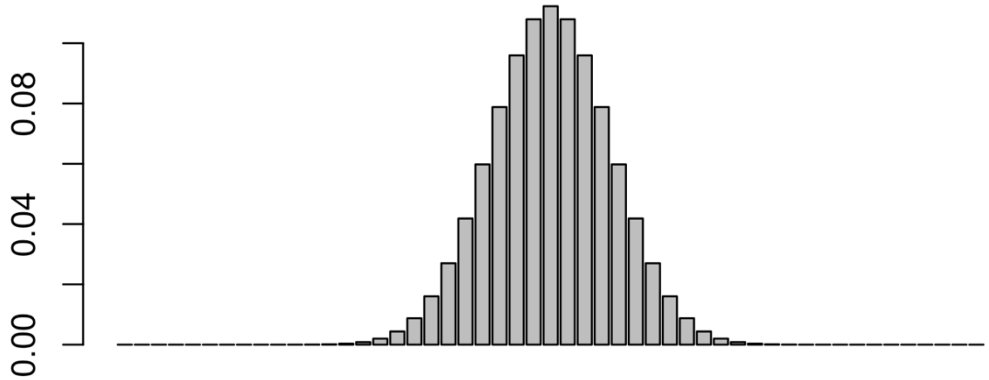

Figura 8.7: Distribuição binomial da probabilidade de se obter 0 a 50 caras em 50 jogadas de moeda. Fonte: própria.

Faça o mesmo para 100 jogadas de moeda para computar todas as probabilidades de 0 a 100 caras.

```
barplot(dbinom(0:100, 100, 0.5), beside = T)
```

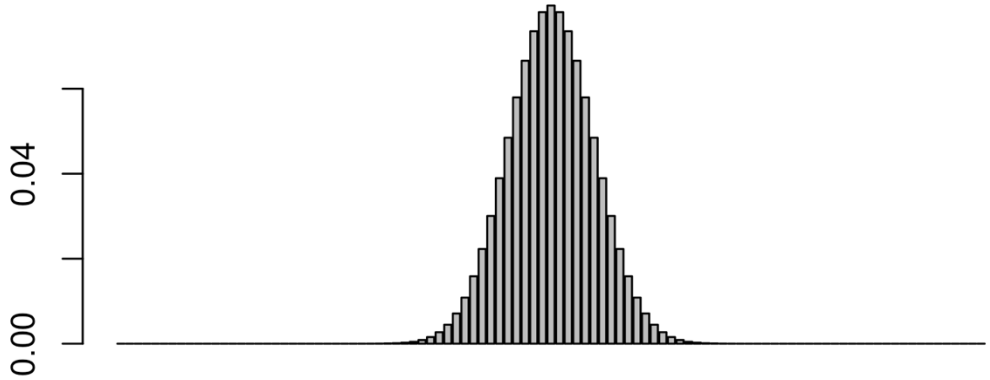

Figura 8.8: Distribuição binomial da probabilidade de se obter 0 a 100 caras em 100 jogadas de moeda. Fonte: própria.

Faça o mesmo para 1000 jogadas de moeda para computar todas as probabilidades de 0 a 1000 caras.

```
barplot(dbinom(0:1000, 1000, 0.5), beside = T)
```

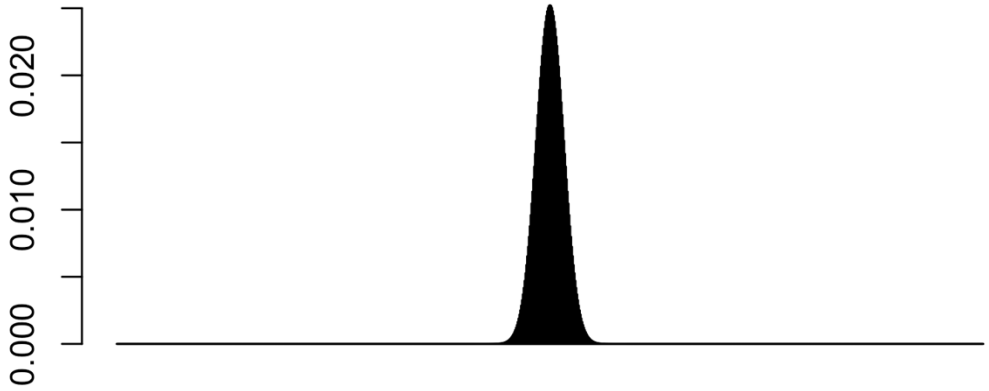

Figura 8.9: Distribuição binomial da probabilidade de se obter 0 a 1000 caras em 1000 jogadas de moeda. Fonte: própria.

Ok! Acho que deu pra pegar a ideia. Esses gráficos te lembram alguma coisa?

Sim! Trata-se da distribuição normal, que vimos em lições anteriores. Reveja os gráficos plotados na aba Plots por meio das flechas pra esquerda e pra direita. À medida que acrescentamos mais tentativas (3, 6, 12, 25 etc.), o gráfico de probabilidades se aproxima cada vez mais da distribuição normal, também conhecida como curva em forma de sino ou curva gaussiana.

Veja o eixo y da figura em que plotamos a probabilidade de 0 a 1000 caras em 1000 jogadas de moeda. A maior probabilidade – que é resultado de 500 caras em 1000 jogadas – está em torno de 0,025 (2,5%). Neste ponto, cada uma das probabilidades é pequena.

Não importa qual é o número N de tentativas, a soma de todas as probabilidades 0:N é 1. Faça essa checagem: aplique a função `sum()` ao cálculo das probabilidades de 0 a 3 caras em 3 tentativas. Digite `sum(dbinom(0:3, 3, 0.5))`.

```
sum(dbinom(0:3, 3, 0.5))
## [1] 1
```

Faça o mesmo agora para o cálculo de probabilidades de 0 a 100 caras em 100 tentativas. A partir da linha de comando acima, teste `sum(dbinom(0:100, 100, 0.5))`.

```
sum(dbinom(0:100, 100, 0.5))
## [1] 1
```

Ambos os resultados são 1. E é claro que tem que ser! 1 equivale a 100%, e é certeza de que um dos resultados vai ser um dos resultados possíveis. Isso te parece uma obviedade? Acompanhe o raciocínio, porque isso vai ter consequências importantes para o cálculo de probabilidades.

Podemos então estipular que a área total da figura que se forma pela distribuição binomial é 1 (ou 100%). Vemos também que as menores probabilidades se encontram nos extremos da distribuição, também chamados de "caudas". Você consegue imaginar que o resultado de ter 90 caras em 100 jogadas de moeda está na cauda superior da distribuição, certo?

Imagine então se fizermos um corte vertical a partir da probabilidade de ter 90 caras. Exemplifico isso na Figura 8.10. Neste ponto, as probabilidades são tão pequenas que parecem ser zero – mas não são! É possível calcular a probabilidade de ter 90 caras ou mais, em 100 jogadas, fazendo o cálculo da área que vai de 90 até 100, concorda? Isso nada mais é do que a soma de todas as probabilidades a partir daquele ponto.

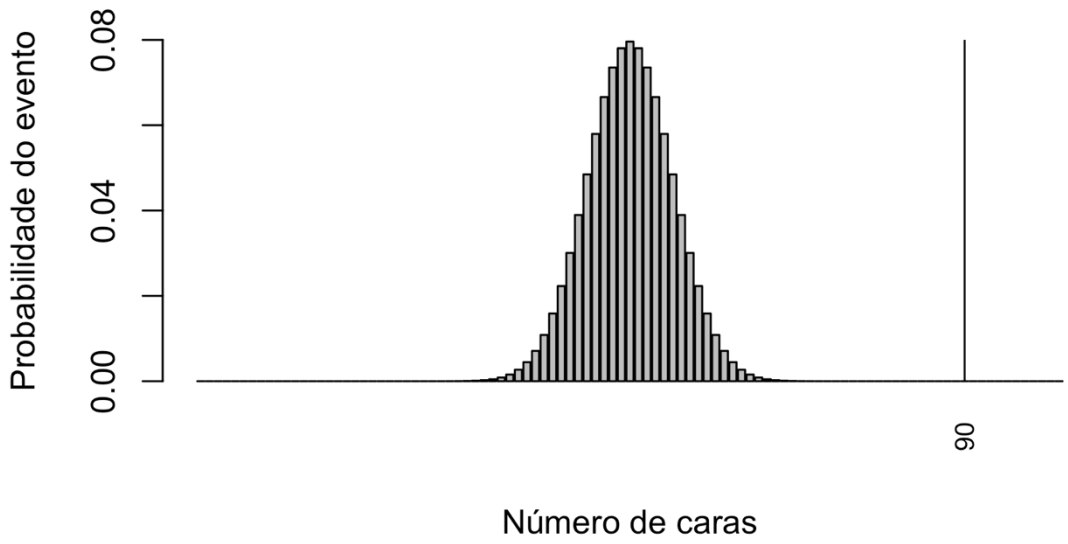

Figura 8.10: Distribuição binomial da probabilidade de se obter 0 a 100 caras em 100 jogadas de moeda, com corte em 90 ou mais caras. Fonte: própria.

Calculemos isso no R. A partir da linha de comando acima, em vez de colocar 0:100 no primeiro argumento, vamos colocar 90:100, para obter `sum(dbinom(90:100, 100, 0.5))`.

```
sum(dbinom(90:100, 100, 0.5))
## [1] 1.531645e-17
```

A probabilidade de ter qualquer resultado entre 90 e 100 caras, em 100 jogadas de moeda, é 1.531645e-17, que o R novamente nos dá em notação e. Veja que, neste caso, o valor após 'e' ('elevado a') é negativo. Esse número equivale a 1,531645 * (10^{-17}). Quando elevamos um número a uma potência negativa, ele se torna um número menor do que 1. Em termos práticos, esse número é 0,0000... com 16 zeros após a vírgula e terminado em 1531645! Você pode pensar assim: o número negativo é o número de zeros, incluindo aquele antes da vírgula. Ou você ainda pode pensar: esse é um número absurdamente pequeno!!! (Veja também: http://www.miniwebtool.com/scientific-notation-to-decimal-converter/)

Ou seja, não só a probabilidade de ter 90 caras em 100 jogadas é pequena, mas também a soma de todas as probabilidades entre 90 e 100 é infinitesimalmente pequena. É esse conhecimento intuitivo que te faz desconfiar que um resultado de 90 caras em 100

jogadas, caso ocorresse, provavelmente não é fruto do acaso, e que te faz concluir, portanto, que sua oponente deve estar roubando! A probabilidade de isso acontecer *por puro acaso* é tão pequena que você é levado a rejeitar a hipótese nula ("minha oponente não está roubando") e aceitar a hipótese alternativa ("minha oponente está roubando"). Essa probabilidade calculada é chamada de *valor-p*, ou *valor de significância*.

Uma probabilidade de 1.531645e-17 é tão pequena que logo a descartamos como extremamente improvável. Façamos agora o raciocínio inverso: a partir de qual probabilidade você consideraria algo muito pouco provável? A partir de 0,01%? 0,1%? 1%? 2%?

Não há um número certo aí. Mas, convencionalmente, a comunidade científica costuma usar o limite de 5% para considerar algo como muito pouco provável para acontecer ao acaso. Isso é chamado de nível α: o limite estabelecido pelo pesquisador para rejeitar a hipótese nula. Note que, se ele é estabelecido pelo pesquisador, ele não necessariamente precisa ser 5%; você pode estabelecer um limite mais baixo, i.e., um teste mais rigoroso, antes de rejeitar a hipótese nula. O importante é respeitar o limite que você impôs, não mudá-lo de acordo com os resultados!

É possível também fazer o cálculo inverso ao que fizemos acima: se eu estabelecer um nível α 5%, a partir de qual resultado eu devo rejeitar H0 de que minha oponente não está roubando e aceitar H1? No R, isso pode ser calculado com a função `qbinom()`. Digite `qbinom(0.05, 100, 0.5, lower.tail = F)`.

```
qbinom(0.05, 100, 0.5, lower.tail = F)
## [1] 58
```

Na linha de comando acima, o que inserimos foi: o nível α estabelecido (0.05), o número de tentativas (100), a probabilidade de o evento ocorrer (0.5) e para ignorar a cauda inferior da distribuição (só estamos atentando à cauda superior). O resultado é 58. Isso significa que a soma de todas as probabilidades de ocorrerem 58 caras ou mais é 5% – ou seja, já é bem baixa *em caso de a hipótese nula ser verdadeira* (= de sua oponente não estar roubando). Tal ponto é marcado na Figura 8.11, por meio da linha azul.

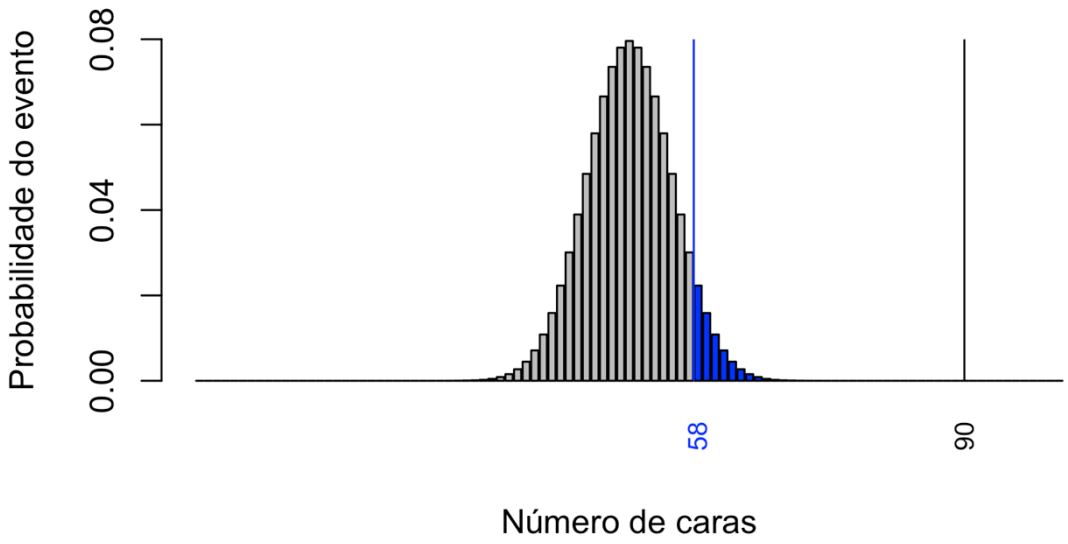

Figura 8.11: Distribuição binomial da probabilidade de se obter 0 a 100 caras em 100 jogadas de moeda, com corte em 58 e em 90 ou mais caras. Fonte: própria.

Como esse número – 58 – se compara com o valor que você estabeleceu quando pedi para que estipulasse um valor a partir do qual você consideraria que estou roubando? Para saber o nível α que você estabeleceu para este teste, faça a soma das probabilidades de se obter de x a 100 caras, em 100 jogadas, quando a probabilidade de se ter cara é 50%, com uso de `sum()` e `dbinom()`.

```
sum(dbinom(x:100, 100, 0.5))
```

```
## [1] 0.02844397
```

N.B.: O resultado acima é para x = 60. Seu resultado será diferente se tiver estipulado outro valor para `x`.

Se você estabeleceu um valor mais alto do que 58, você queria ter muita certeza de que eu não estava roubando antes de sair fazendo acusações! Em outras palavras, você estabeleceu um nível α mais rigoroso, abaixo de 5%. Se você estabeleceu um valor mais baixo do que 58, o nível α foi menos rigoroso, acima de 5%. Este ponto está ilustrado na Figura 8.12 com a linha vermelha (neste exemplo, em 60, que foi o valor estabelecido para `x` acima).

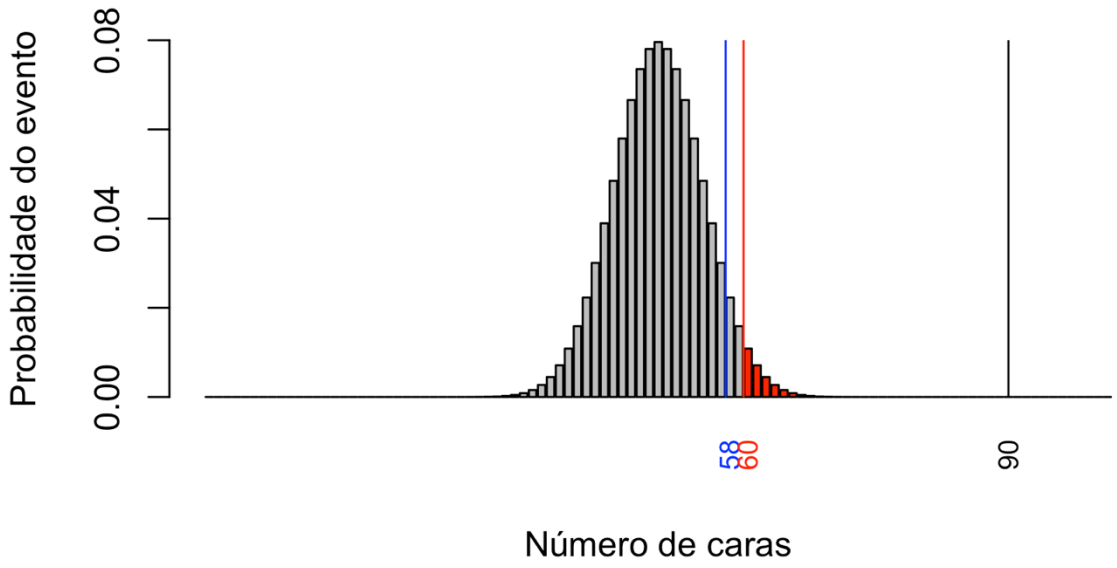

Figura 8.12: Distribuição binomial da probabilidade de se obter 0 a 100 caras em 100 jogadas de moeda, com corte em 58, 90 e x ou mais caras. Fonte: própria.

Neste exemplo, em que você é um dos jogadores e você *sabe* que não está roubando, sua hipótese só investigou a possibilidade de que *eu* pudesse estar roubando. Imagine agora outra situação: e se houver um juiz imparcial que está observando nossas jogadas? A partir de que ponto essa pessoa poderia começar a desconfiar de que um dos jogadores está roubando? Seria o mesmo limite (= 58 caras ou 58 coroas)?

- Sim
- Não
- Não faço ideia!

Pois é, o limite não seria o mesmo. Da perspectiva de uma terceira pessoa, é necessário olhar para ambos os lados da distribuição. Se o nível α estabelecido permanecer 5%, há que se levar em conta 2,5% da cauda inferior e 2,5% da cauda superior (os resultados menos prováveis tanto de um lado quanto de outro).

Vamos fazer esse cálculo. A partir de uma das linhas de comando acima, em que usamos a função `qbinom()`, digite `qbinom(0.025, 100, 0.5, lower.tail = T)`, para verificar a cauda inferior.

```
qbinom(0.025, 100, 0.5, lower.tail = T)
```

```
## [1] 40
```

E digite `qbinom(0.025, 100, 0.5, lower.tail = F)`, para verificar a cauda superior.

```
qbinom(0.025, 100, 0.5, lower.tail = F)
## [1] 60
```

O R deu os resultados 40 e 60. No jogo da moeda, isso quer dizer que somente a partir de um resultado de 40 caras e 60 coroas (ou vice-versa) é que se poderia rejeitar a hipótese nula e assumir que um dos jogadores está roubando.

Este último é chamado de um *teste bidirecional*, e o teste anterior (em que você sabe que não está roubando) foi *unidirecional*. Ambos estão ilustrados na Figura 8.13. Note que, neste último tipo de teste (figura no topo), é mais fácil rejeitar H0.

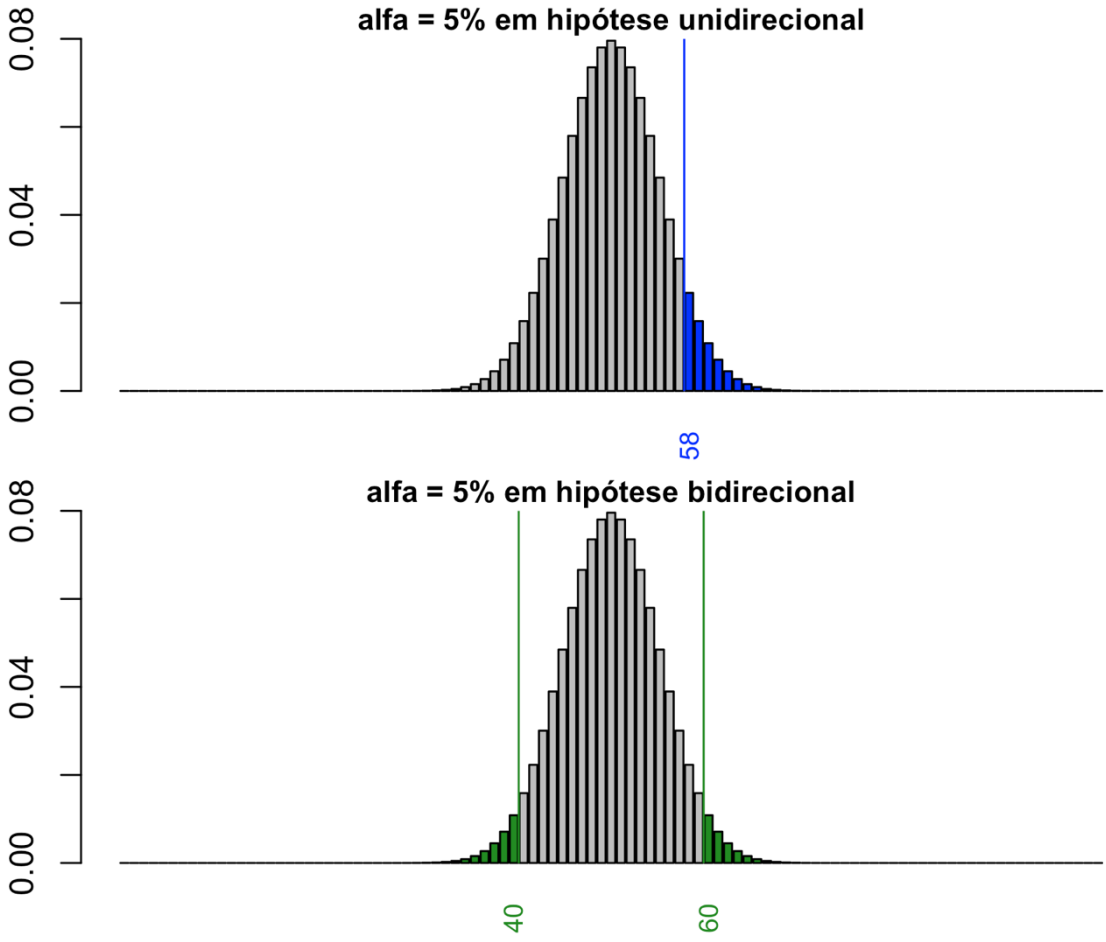

Figura 8.13: Distribuição binomial da probabilidade de se obter 0 a 100 caras em 100 jogadas de moeda, com corte em α = 0,05 em hipóteses uni e bidirecional.

Quando se aplica um teste uni ou bidirecional? No exemplo da moeda, você tinha o conhecimento prévio de que não estava roubando e, portanto, se alguém estava fazendo isso, só poderia ser eu. No caso de um juiz imparcial, ele não pode saber se qualquer dos jogadores está roubando, e precisa ficar atento a ambas as possibilidades.

A lição aqui é o fato de que conhecimento prévio compensa. Se você tem boas evidências – a partir da literatura sobre o assunto, por exemplo – de que determinado fenômeno ocorre de certa maneira, você pode rejeitar de antemão certos cenários.

Vamos pensar num exemplo linguístico. Para as vogais médias pretônicas /e/ e /o/ (em palavras como "relógio" e "romã") – ver lições anteriores! –, temos, pela literatura, que paraibanos têm realização mais baixa (= F1 mais alto) do que a dos paulistanos. Neste caso, há boas evidências para que um pesquisador estabeleça, como H1, que os paraibanos residentes em São Paulo terão medidas de F1 mais altas do que para falantes paulistanos. Essa é uma hipótese unidirecional, que já estabelece de antemão a direção da diferença, se houver.

O que torna possível aplicar os mesmos cálculos no exemplo da moeda e em vogais pretônicas? É a distribuição normal! Você se lembra da forma da distribuição das medidas de F1 da vogal /e/ pretônica para paraibanos (Lição 7)? As mesmas propriedades aqui discutidas são válidas para diversos outros fenômenos variáveis no mundo.

Antes de prosseguir, veja o quadrinho da Figura 8.14.

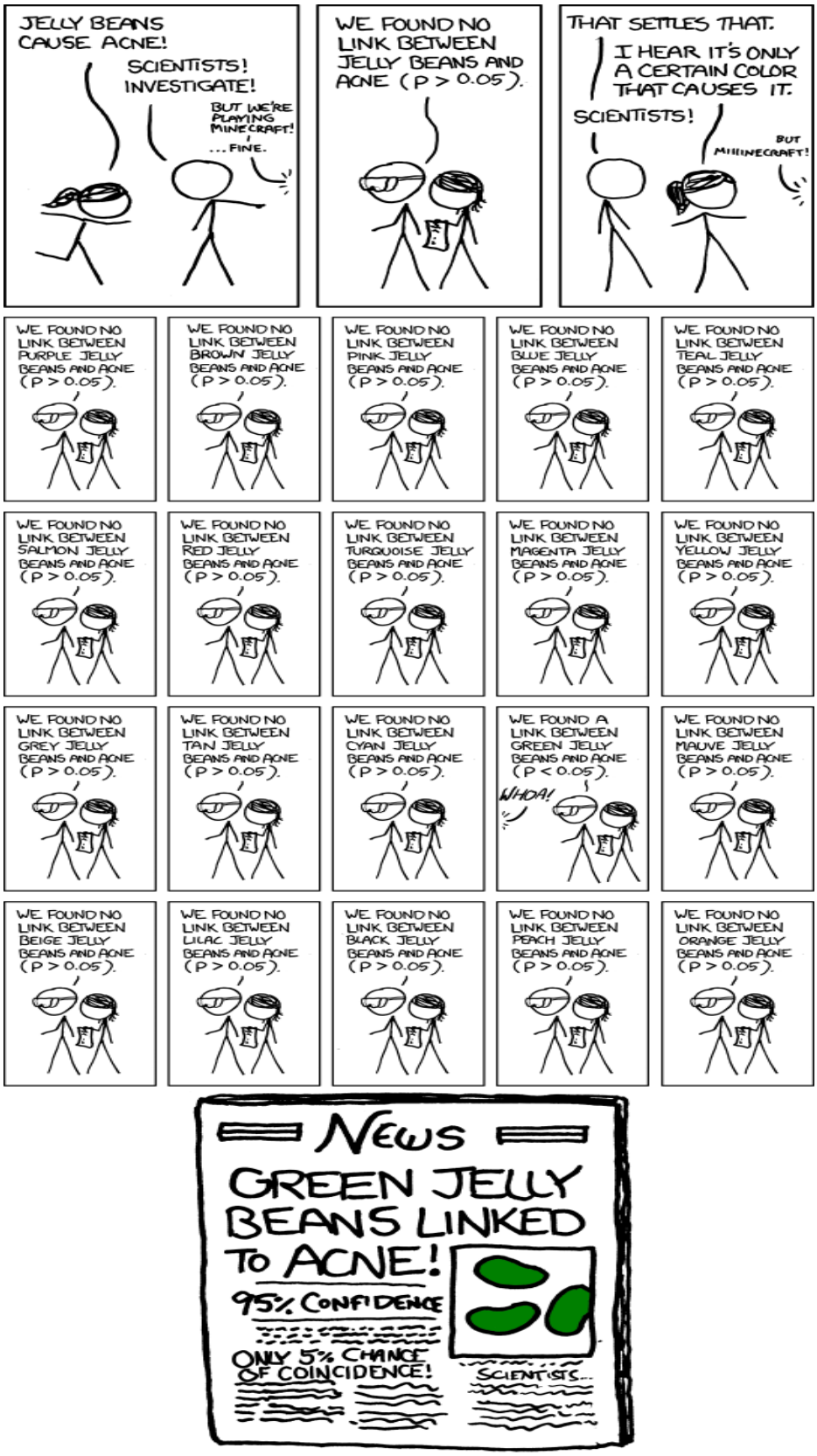

Figura 8.14: Significância. Fonte: https://xkcd.com/882/.

O quadrinho bem exemplifica o conceito de significância, assim como um problema para as análises estatísticas que decorrem do próprio conceito. Vimos que significância é a probabilidade de se observar determinada distribuição em caso de a hipótese nula ser verdadeira. Certas distribuições são tão improváveis que rejeitamos a hipótese nula e acatamos a hipótese alternativa.

Contudo, trata-se de uma medida de *probabilidade*. Isso significa que sempre há uma chance de que nossas conclusões foram errôneas. Se se estabelece um nível α de 5%, existe uma probabilidade de que chegamos a conclusões equivocadas em média 5% das vezes, ou 1 em 20 – exatamente o que ocorre no quadrinho! O número de conclusões equivocadas tende a aumentar quanto mais testes realizamos – se realizamos 100 testes estatísticos, com nível $\alpha = 5\%$, as chances são de que 5 dos nossos resultados estão equivocados.

Um tipo de erro é o chamado *Erro do Tipo I*, que se refere à rejeição incorreta da hipótese nula. Trata-se de um falso positivo: encontramos um efeito que não existe de fato. Um problema é que não há como identificar qual ou quais dos resultados são falsos, uma vez que se trata de probabilidade.

Um problema correlato é fazer o oposto: erroneamente acatar a hipótese nula e não assumir a hipótese alternativa. Isso é chamado de *Erro do Tipo II*, que se refere a um falso negativo: não encontramos um efeito que existe de fato. Novamente, não temos como saber quais dos múltiplos testes aplicados geraram uma conclusão errônea.

O que seria um exemplo de Erro do Tipo I?

- numa sala de aula, não reprovar um aluno que colou na prova
- num tribunal, condenar um réu inocente
- as duas situações descritas

O que seria um exemplo de Erro do Tipo II?

- numa fábrica de automóveis, enviar para venda um carro com falha no airbag
- num laboratório, identificar como HIV positivo um paciente que não tem o vírus HIV

- as duas situações descritas

Se realizarmos 1000 testes com $\alpha = 5\%$, quantos falsos positivos esperamos obter em média?

- 5
- 50
- 500

Os erros tanto do Tipo I quanto do Tipo II podem ser graves. Felizmente, há métodos para contornar esse problema. Um desses métodos é a *Correção de Bonferroni*, cuja aplicação é bastante simples. Se se deseja realizar 20 testes com $\alpha = 5\%$, a correção de Bonferroni sugere que cada hipótese seja testada com α 0,05 / 20 = 0,0025. Em outras palavras, o limite para se considerar um resultado significativo deve ser ajustado para que não se aumente a chance de uma conclusão errônea.

Outro método para diminuir a chance de falsos positivos ou falsos negativos é simplesmente não realizar todo e qualquer possível teste estatístico em determinado conjunto de dados. Uma pesquisa bem estruturada tem claras quais são as perguntas e as hipóteses que serão testadas, e não se perde em um sem número de questões.

Cabe ainda apresentar o conceito de *intervalo de confiança*. Nas últimas duas lições, calculamos as médias de valores de F1 e F2 das vogais pretônicas para plotar os espaços vocálicos de paraibanos e paulistanos. As médias resumem em um único número a estimativa de um parâmetro da população. Mas vimos também, pelos boxplots, que a distribuição real dos valores de formantes não se concentram em um único ponto, e sim são variáveis. A Figura 8.2 ilustra essa observação.

Existe a possibilidade, portanto, de que a média calculada esteja incorreta, e que seu verdadeiro valor esteja acima ou abaixo do valor estimado. O intervalo de confiança estabelece um valor mínimo e um valor máximo em que se calcula estar o verdadeiro parâmetro da população. Podemos não ter um valor exato, mas ter um intervalo em que é mais provável estar tal medida.

Quando se estabelece um nível α, na prática, também se estabelece um *nível de confiança*. Quando α = 5%, o nível de confiança é de 95%; quando α = 1%, o nível de confiança é 99%. Nos boxplots, quando plotamos a figura com notch = T, o R faz o cálculo do intervalo de confiança e o representa graficamente pelos entalhes.

Por fim, é importante apontar para algumas concepções *errôneas* acerca do conceito de significância. Embora, no uso comum, dizer que algo é "significativo" implica dizer que é "importante" ou "relevante", a significância estatística *nada* tem a ver com importância ou relevância. A significância ou valor-*p* é uma medida de probabilidade de se observar determinada distribuição em caso de a hipótese nula ser verdadeira (no exemplo, a probabilidade de ter 90 caras em 100 jogadas se um dos jogadores não estiver roubando). Como tal, a medida não consegue avaliar quão importante é esse fato – só o pesquisador e a comunidade científica podem dizer.

Segundo, significância *não* é boa nem ruim. Ela simplesmente é! Um valor-*p* abaixo de 5% *não* é intrinsicamente bom! Isso depende da hipótese levantada pelo pesquisador, da teoria, daquilo que se quer testar. Veja que, no estudo sobre vogais médias pretônicas na fala de migrantes, levanta-se a hipótese de que as vogais médias sejam mais baixas para os paraibanos do que para os paulistanos, mas há mais interesse em saber quem são os falantes que, em certa medida, não seguem essa regra – ou seja, aqueles que não apresentam diferenças significativas em relação aos paulistanos e para quem a hipótese nula não é rejeitada.

Por fim, um valor-*p* ou de significância abaixo de 5% *não* prova que sua hipótese estava correta. O que se faz é *rejeitar a hipótese nula* e aceitar a hipótese alternativa, mas isso é diferente de dizer que H1 estava certa.

Para saber mais

Recomendo a leitura de Gigerenzer (2004) para entender um pouco mais desses conceitos básicos e evitar a aplicação impensada de testes estatísticos.

Exercícios

1. Qual das afirmações a seguir é uma hipótese (H1) bem formulada? Explique sua resposta.
    a. A medida média de F1 em Hz da vogal pretônica /e/ na fala de paraibanos é mais alta do que na fala de paulistanos.
    b. O apagamento de /r/ pós-vocálico na fala de novaiorquinos possivelmente é mais frequente entre funcionários da Macy's do que os da Saks.
    c. O próximo sintagma nominal plural a ser falado por mim será na forma padrão do português.
2. Qual é a H0 correspondente à H1 bem formulada acima?
    a. A medida média de F1 em Hz da vogal pretônica /e/ na fala de paraibanos não é mais alta do que na fala de paulistanos.
    b. O apagamento de /r/ pós-vocálico na fala de novaiorquinos possivelmente não é mais frequente entre funcionários da Macy's do que os da Saks
    c. O próximo sintagma nominal plural a ser falado por mim não será na forma padrão do português.
3. Nesta lição, vimos o conceito de significância. Qual é o nível α normalmente adotado como limite de significância?
    a. 10%
    b. 5%
    c. 1%
    d. 0,1%
    e. 0,05%
4. Qual é a definição de significância?
    a. a probabilidade de se observar determinada distribuição em caso de a hipótese nula ser verdadeira
    b. a probabilidade de a hipótese nula ser verdadeira

   c. a possibilidade de que se chegou a uma conclusão errônea quando a hipótese nula é verdadeira
   d. a possibilidade de rejeitar H1 equivocadamente em caso de a hipótese nula ser verdadeira

5. Numa rolada de dado, qual é a probabilidade de se ter o número 5, considerando-se que o dado não é viciado?
6. Numa rolada de dado, qual é a probabilidade de se ter um número ímpar, considerando-se que o dado não é viciado?
7. A probabilidade de obter três caras em três jogadas de moeda é 1/8 = 0,125. Qual é a probabilidade de ter três números ímpares em três roladas de um dado, considerando-se que o dado não é viciado?
   a. 0,05
   b. 0,125
   c. 0,375
   d. 0,50
8. Calcule a probabilidade de saírem 75 caras, em 100 jogadas de moeda, considerando-se que a moeda é honesta. Use a função `dbinom()` para seu cálculo.
9. Qual é o resultado da soma das probabilidades de todos os resultados possíveis em 1000 jogadas de moeda?
   a. 0,5
   b. 1
   c. 100
   d. É impossível calcular
10. Calcule a soma das probabilidades de todos os resultados possíveis em 1000 jogadas de moeda. Para isso, use as funções `sum()` e `dbinom()`.
11. Qual é a notação decimal para o número em notação e 2e+05?
12. Qual é a notação decimal para o número em notação e 1.45e-03?
13. Qual não é outro nome para a forma da distribuição normal?

a. curva em forma de Gauss
b. curva em forma de sino
c. distribuição gaussiana

14. Se se estabelece um nível $\alpha = 1\%$ e se observa um valor de significância igual a 2e-02, o pesquisador deve...
    a. rejeitar a hipótese alternativa e acatar a hipótese nula
    b. rejeitar a hipótese nula e acatar a hipótese alternativa
    c. realizar o teste estatístico novamente
15. Quando um teste estatístico gera um $p < 0,05$,...
    a. deve-se acatar a hipótese alternativa
    b. tem-se confirmação de que a hipótese está correta
    c. pode-se ter certeza da veracidade da hipótese alternativa
16. Rejeitar equivocadamente a hipótese nula é um...
    a. Erro do Tipo I
    b. Erro do Tipo II
    c. Erro do Tipo III
17. Erros do Tipo I e do Tipo II ocorrem...
    a. porque a medida de significância se refere a uma probabilidade e sempre há chance de erro
    b. porque certos cuidados não foram tomados no momento da escolha do teste estatístico
    c. quando não se aplica a correção de Bonferroni na realização de múltiplos testes estatísticos
18. O intervalo de confiança...
    a. estabelece um valor mínimo e um valor máximo dentre os quais há maior probabilidade de estar o verdadeiro parâmetro da população
    b. estabelece o valor com maior probabilidade de ser o verdadeiro parâmetro da população

c. estabelece um valor mínimo e um valor máximo do verdadeiro parâmetro da população
d. estabelece o valor que tem 95% de probabilidade de ser o verdadeiro parâmetro da população

Lição 9: Testes de Proporção e Qui-Quadrado

A lição anterior, sobre conceitos fundamentais da Estatística Inferencial, é a base de todas as próximas lições, que apresentam diferentes testes estatísticos que podem ser aplicados aos dados. Esta e as próximas duas lições tratam de análises *univariadas*, que verificam se há correlação entre duas variáveis – em geral, uma variável *dependente* e uma variável *independente*.

Rode as linhas de comando a seguir para deixar disponível o dataframe `ds` nesta sessão. Defina como diretório de trabalho aquele que, em seu computador, contém o arquivo LabovDS.csv.

```
# Definir diretório de trabalho

#setwd()

# Importar planilha de dados

ds <- read_csv("LabovDS.csv",
               col_types = cols(.default = col_factor(),
                                r = col_factor(levels = c("r0", "r1", "d
"))
                                )
               ) %>%
  filter(r != "d") %>%
  droplevels()
```

Antes de tudo, carregue o pacote tidyverse, que vamos usar nesta lição (ó, surpresa!).

```
library(tidyverse)
```

A escolha do teste que pode ser aplicado depende fundamentalmente da natureza das variáveis. Que tipo de variável é o apagamento (r0) vs. a realização (r1) de /r/ pós-vocálico no inglês, como é o caso da variável estudada por Labov em seu famoso estudo nas lojas de departamento?

- nominal
- numérica

- ordinal

Para variáveis nominais, como é o caso da variável /r/ pós-vocálico, podemos aplicar *testes de proporção* e *testes de qui-quadrado* para avaliar se há diferenças significativas entre proporções das variantes.

Nesta lição, vamos trabalhar com o arquivo de dados LabovDS.csv, que havíamos visto na Lição 4. Cheque a estrutura do dataframe `ds`.

```
str(ds)
```

```
## spec_tbl_df [730 × 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ r       : Factor w/ 2 levels “r0”,”r1”: 2 2 2 2 2 2 2 2 2 2 ...
##  $ store   : Factor w/ 3 levels “Saks”,”Macys”,..: 1 1 1 1 1 1 1 1
1 1 ...
##  $ emphasis: Factor w/ 2 levels “casual”,”emphatic”: 1 1 1 1 1 1 1
1 1 1 ...
##  $ word    : Factor w/ 2 levels “fouRth”,”flooR”: 1 1 1 1 1 1 1 1 1
1 ...
##  - attr(*, “spec”)=
##   .. cols(
##   ..   .default = col_factor(),
##   ..   r = col_factor(levels = c(“r0”, “r1”, “d”), ordered = FALSE,
include_na = FALSE),
##   ..   store = col_factor(levels = NULL, ordered = FALSE, include_n
a = FALSE),
##   ..   emphasis = col_factor(levels = NULL, ordered = FALSE, includ
e_na = FALSE),
##   ..   word = col_factor(levels = NULL, ordered = FALSE, include_na
= FALSE)
##   .. )
##  - attr(*, “problems”)=<externalptr>
```

Neste dataframe, a variável r contém apenas r0 e r1, pois os dados d (= duvidosos) já foram excluídos.

Como vimos nas Lições 4 e 5, a análise estatística começa com a estatística descritiva, com tabelas e gráficos. Façamos então cálculo de frequências e proporções da distribuição da variável r. No *script*, vamos criar um dataframe chamado `prop.r`, com os dados de `ds`, a frequência de r, e uma nova coluna chamada prop por meio da aplicação da função `prop.table()` aos dados.

```
prop.r <- ds %>%
  count(r) %>%
  mutate(prop = prop.table(n)) %>%
  print()
```

```
## # A tibble: 2 × 3
##   r         n  prop
##   <fct> <int> <dbl>
## 1 r0      499 0.684
## 2 r1      231 0.316
```

Como já havíamos visto na Lição 4, há 499 dados de r0 e 231 dados de r1, que correspondem a 68% e 32%. Vamos agora plotar um gráfico de barras exploratório para visualizar essas proporções (Figura 9.1).

```
ggplot(prop.r, aes(x = r, y = prop, fill = r)) +
  geom_bar(stat = "identity")
```

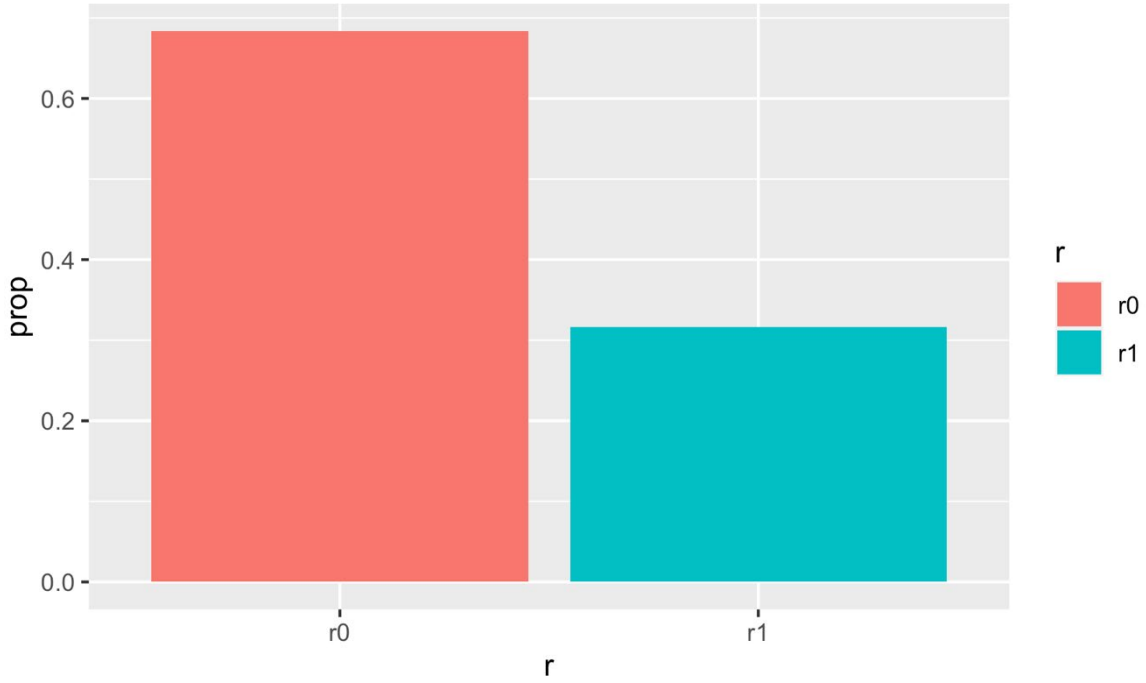

Figura 9.1: Distribuição das variantes de /r/ pós-vocálico nos dados de Labov (1972). Fonte: própria.

Faça a tabela de distribuição de dados da variável r pelas funções da instalação base do R – `with()` e `table()`. Guarde-a em um objeto chamado `tab.r`.

```
tab.r <- with(ds, table(r))
```

Inspecione o objeto `tab.r`.

```
tab.r
```

```
## r
##  r0  r1
## 499 231
```

Até aqui, refizemos alguns dos passos da Lição 4, certo? A partir das tabelas e dos gráficos – o primeiro passo de qualquer boa análise quantitativa! –, o pesquisador deve começar a avaliar quais diferenças podem ter ocorrido por acaso e quais têm menor chance de terem ocorrido aleatoriamente. O teste de proporções contrasta uma distribuição observada com uma distribuição esperada.

Em sua forma mais simples, o teste de proporções indica se a diferença entre as proporções das variantes de uma variável é significativa. No caso em questão, podemos nos perguntar se a diferença entre as proporções das variantes de /r/ (aproximadamente 68% e 32%) é significativa. A função para responder a essa questão é `prop.test()`, que é aplicada sobre uma tabela de *frequências*. Aplique-a então à tabela `tab.r` e guarde o resultado em um objeto chamado `teste.prop`.

```
teste.prop <- prop.test(tab.r)
```

Veja o resultado do teste de proporções.

```
teste.prop

##
##  1-sample proportions test with continuity correction
##
## data:  tab.r, null probability 0.5
## X-squared = 97.656, df = 1, p-value < 2.2e-16
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.6482366 0.7169260
## sample estimates:
##         p
## 0.6835616
```

Vejamos o que o R nos informa. A primeira linha nos diz o teste estatístico que foi realizado: um teste de proporções com uma amostra (a distribuição de uma variável). Em seguida, o R informa os dados utilizados – `tab.r` – e a proporção contra a qual contrastou a proporção observada – 0.5. Ou seja, o R comparou a distribuição de proporções de aproximadamente 68%-32% com uma proporção de 50%-50%. Este caso é semelhante ao da moeda, que vimos na lição passada! Se a hipótese nula prevê que a distribuição deveria ter sido meio a meio, qual é a probabilidade de se ter observado uma distribuição de 68% pra 32%?

Na terceira linha, o R informa que tal probabilidade é extremamente pequena: 2.2e-16. (Veremos daqui a pouco o que são os valores X-squared (= qui-quadrado) e df (= graus de liberdade).) A quarta linha é a afirmação que expressa a hipótese alternativa, H1: "a verdadeira p (= proporção) não é igual a 0.5". Comparando-se então o valor-*p* gerado (2.2e-16) e a hipótese alternativa, o que deve fazer o pesquisador?

- Rejeitar a hipótese nula e acatar a hipótese alternativa
- Rejeitar a hipótese alternativa e acatar a hipótese nula
- Refazer o teste mais algumas vezes, para verificar se o resultado é o mesmo

Continuemos com a leitura dos resultados. Após a hipótese alternativa, o R indica quais são os valores dentro do intervalo de confiança de 95%. Esses valores são 64,8% e 71,7%, que se referem ao primeiro nível da variável `r`. Entre r0 e r1, qual é o primeiro nível? Dica: olhe o gráfico de barras!

- r0
- r1

O R contrastou a proporção observada do primeiro nível da variável, tomada como valor de referência – r0 = 68% – com a proporção esperada – r0 = 50%. Como 50% não está contido no intervalo de confiança, entre 64,8% e 71,7%, a estimativa de probabilidade de que o verdadeiro parâmetro da distribuição é 50% está abaixo de 5%, pois está efetivamente fora dos 95% do nível de confiança.

Se esse raciocínio pareceu complicado, lembre-se do exemplo da moeda: se se espera que haja 50% de caras e 50% de coroas sob a hipótese nula, ter observado 68% de uma das faces é pouco provável, a depender do número de tentativas. A diferença aqui é que estamos falando de apagamento e de realização de /r/ em vez de cara ou coroa.

O R usou o intervalo de confiança de 95% como *default*, pois não especificamos nenhum outro valor. Mas, como visto na lição anterior, o pesquisador pode determinar outro nível α para seu teste (diferente do valor *default* 5%) e, consequentemente, mudar o nível de confiança. Veja na ajuda da função `prop.test()` qual argumento precisaríamos ter especificado para operar com um IC de 99%.

```
?prop.test
```

N.B.: Resultado aqui omitido.

O argumento é conf.level, cujo valor *default* é 0.95. Então se quiséssemos realizar um teste com 99% de nível de confiança, bastaria especificar conf.level = 0.99. Por fim, o R apresenta a estimativa do primeiro nível da variável que, como já visto, é 68% de r0.

A ajuda de `prop.test()` também informa outros argumentos possíveis da função. No exemplo acima, contrastamos a proporção 68%-32% com 50%-50%, mas também poderíamos ter contrastado com outra proporção. Imagine que um estudo prévio sobre a variável r no inglês de Nova Iorque houvesse notado uma proporção de 70% de apagamento, e Labov quisesse saber se a proporção de 68% de apagamento diferia significativamente. Neste caso, seria necessário especificar p = 0.7 na função `prop.test()`. Faça isso agora com a tabela `tab.r`. Não se preocupe em guardar o resultado em um objeto, pois queremos visualizá-lo imediatamente.

```
prop.test(tab.r, p = 0.7)
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  tab.r, null probability 0.7
## X-squared = 0.86269, df = 1, p-value = 0.353
## alternative hypothesis: true p is not equal to 0.7
## 95 percent confidence interval:
##  0.6482366 0.7169260
## sample estimates:
##         p
## 0.6835616
```

O R agora apresenta os resultados em relação a uma probabilidade esperada de 70%. Qual é o intervalo de confiança neste caso?

- de 0,35 a 0,86
- de 0,64 a 0,71
- de 0,68 a 0,86

Você deve ter notado que se trata exatamente do mesmo intervalo de confiança para o teste com probabilidade esperada de 50%. O que mudou agora, no entanto, é que a probabilidade 0,7 está dentro do intervalo de confiança, entre 0,64 e 0,71, de modo que

o valor-*p* agora está acima do nível α de 5%. Qual é a probabilidade de se ter observado 68% de r0 sob a hipótese nula de que o verdadeiro parâmetro é 70%?

- 0,35
- 0,68
- 0,7
- 0,86
- 1

No exemplo da moeda, também vimos dois cenários: um em que você avaliava a hipótese de eu estar roubando e outro em que um juiz imparcial avaliava se um dos jogadores estava roubando. Você sabia não estar roubando, de modo que podia estabelecer uma hipótese unidirecional, enquanto o juiz deveria estabelecer uma hipótese bidirecional. Na função `prop.test()`, o *default* é a realização de um teste bidirecional, que simplesmente avalia se as proporções diferem, independentemente de o valor de referência estar acima ou abaixo do valor esperado. Contudo, também se pode estabelecer um teste unidirecional com o argumento alternative. Dê uma olhada na ajuda da função para ver como especificar esse argumento.

Nos dados das lojas de departamento, se o pesquisador tem evidências de que a proporção de apagamento de /r/ está diminuindo em Nova Iorque, ele pode estabelecer uma hipótese unidirecional, em que a proporção esperada é *menor* do que 0.7. Aplique este teste sobre os mesmos dados acima, com a adição de alternative = “less”.

```
prop.test(tab.r, p = 0.70, alternative = “less”)
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  tab.r, null probability 0.7
## X-squared = 0.86269, df = 1, p-value = 0.1765
## alternative hypothesis: true p is less than 0.7
## 95 percent confidence interval:
##  0.0000000 0.7118195
## sample estimates:
##         p
## 0.6835616
```

Veja que, neste caso, mudam as estimativas do intervalo de confiança, a hipótese alternativa e as medidas estatísticas de qui-quadrado, graus de liberdade e valor-*p*. Por curiosidade, faça o teste agora com alternative = "greater".

```
prop.test(tab.r, p = 0.70, alternative = "greater")
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  tab.r, null probability 0.7
## X-squared = 0.86269, df = 1, p-value = 0.8235
## alternative hypothesis: true p is greater than 0.7
## 95 percent confidence interval:
##  0.6539154 1.0000000
## sample estimates:
##         p
## 0.6835616
```

Este último também foi um teste unidirecional, mas que testou a hipótese de que a proporção verdadeira é *maior* do que 70%. Compare os valores do intervalo de confiança, a enunciação da hipótese alternativa e as medidas de qui-quadrado, graus de liberdade e valor-*p* em relação ao teste unidirecional anterior. Diante da hipótese alternativa de que a proporção fosse 70% ou mais, a proporção de 68% torna a hipótese nula (= proporção não é maior do que 70%) mais provável, com probabilidade igual a 82%.

Até agora, só comparamos a proporção da VD com outras proporções esperadas. Mas é provável que, em seus dados, você queira comparar proporções entre dois grupos. A partir do dataframe `ds`, crie um novo dataframe chamado `prop.word`, computando as frequências de word e r (nessa ordem), agrupando os dados pela variável word, e computando as proporções de word numa variável chamada prop com a função `prop.table()`.

```
prop.word <- ds %>%
  count(word, r) %>%
  group_by(word) %>%
  mutate(prop = prop.table(n)) %>%
  print()
```

```
## # A tibble: 4 × 4
## # Groups:   word [2]
##   word   r         n  prop
##   <fct>  <fct> <int> <dbl>
```

```
## 1 fouRth r0      295 0.770
## 2 fouRth r1       88 0.230
## 3 flooR  r0      204 0.588
## 4 flooR  r1      143 0.412
```

Visualize essas proporções com um gráfico de barras (Figura 9.2). Coloque na função `ggplot()` o nome do dataframe relevante, e as variáveis x, y e fill.

```
ggplot(prop.word, aes(x = word, y = prop, fill = r)) +
  geom_bar(stat = "identity")
```

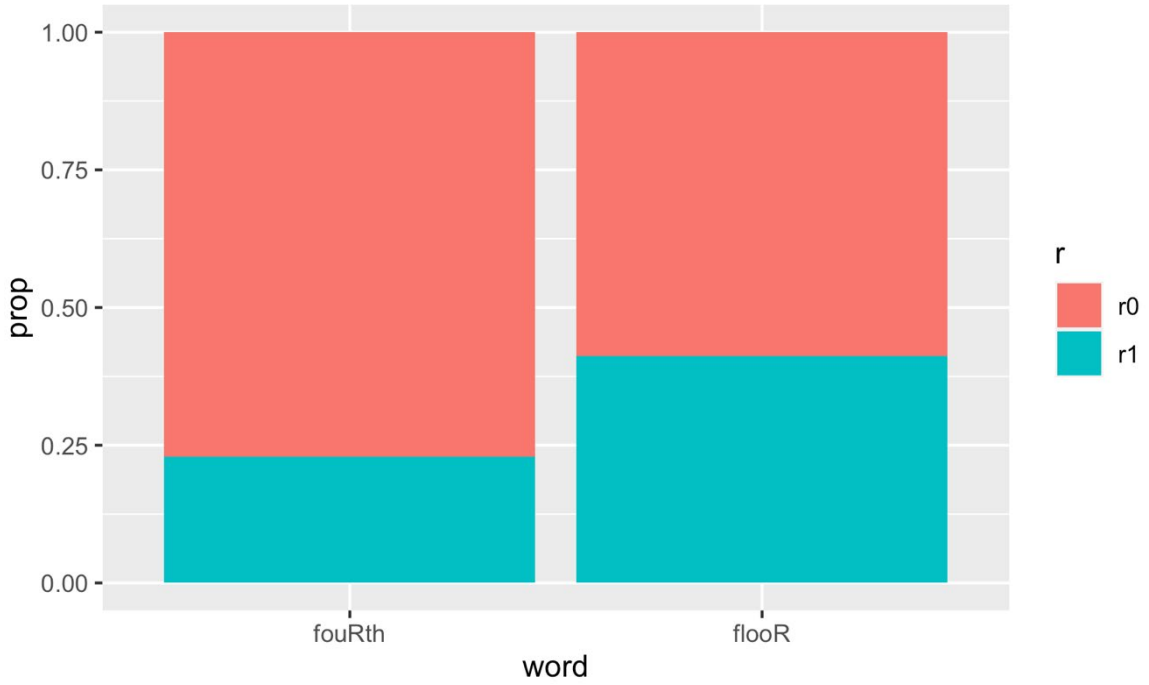

Figura 9.2: Distribuição das variantes de /r/ pós-vocálico por palavra, nos dados de Labov (1972). Fonte: própria.

Faça também uma tabela de frequências da variável word pela variável dependente r com as funções da instalação base do R, e guarde o resultado em um objeto chamado `tab.word`.

```
tab.word <- with(ds, table(word, r))
```

Inspecione agora a tabela `tab.word`.

```
tab.word

##         r
## word      r0  r1
##   fouRth 295  88
##   flooR  204 143
```

Novamente, refizemos os passos de lições anteriores, com a criação de tabelas e gráficos. Isso porque os testes estatísticos relevantes são sugeridos justamente por essas primeiras inspeções dos dados. Na figura, vemos que as barras de proporção de apagamento de /r/ nas palavras floor e fourth parecem ser diferentes. Será que as proporções são as mesmas ou diferem significativamente?

Para fazer a comparação de proporções entre dois grupos – aqui, entre os dois itens lexicais –, usamos a função `chisq.test()`. Esta função toma como argumento uma tabela de *frequências* (assim como `prop.test()`). Aplique então `chisq.test()` à tabela `tab.word` e guarde o resultado em um objeto chamado `x2.word`.

```
x2.word <- chisq.test(tab.word)
```

Veja agora o resultado do teste de qui-quadrado.

```
x2.word

##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tab.word
## X-squared = 27.147, df = 1, p-value = 1.886e-07
```

Voilà! Este é um teste de qui-quadrado. Sua aplicação é extremamente fácil, não? Faz-se uma tabela de frequências e aplica-se o teste sobre ela. Na prática, é isso que você vai fazer com suas variáveis nominais: tabela, gráfico, teste.

Mas importa saber não apenas como aplicá-lo no R, mas também *quando* aplicá-lo e *como ler os resultados*. Você sabe agora quando aplicá-lo: quando se tem uma VD nominal e se quer comparar proporções entre grupos (os níveis de uma VI também nominal). O resultado do teste de qui-quadrado é bastante simples e direto: o R informa o teste realizado (teste de qui-quadrado de Pearson), o conjunto de dados (`tab.word`), e valores de qui-quadrado, graus de liberdade e valor-*p*.

Você já sabe interpretar o valor-*p*: a probabilidade de se ter observado tal distribuição dos dados em caso de a hipótese nula ser verdadeira. Aqui, o teste de qui-quadrado indica que as proporções entre os grupos (59% de r0 em "floor" e 77% em "fourth") são significativamente diferentes, pelo valor-*p* abaixo de 5%. Mas o que querem dizer as medidas de qui-quadrado e de graus de liberdade?

Apesar de o resultado do teste de qui-quadrado ser bastante simples e direto, o R na verdade computou outros valores com o teste. Verifique a estrutura do objeto `x2.word` por meio da função `str()`.

```
str(x2.word)
```

```
## List of 9
##  $ statistic: Named num 27.1
##   ..- attr(*, “names”)= chr “X-squared”
##  $ parameter: Named int 1
##   ..- attr(*, “names”)= chr “df”
##  $ p.value  : num 1.89e-07
##  $ method   : chr “Pearson's Chi-squared test with Yates' continuit
y correction”
##  $ data.name: chr “tab.word”
##  $ observed : 'table' int [1:2, 1:2] 295 204 88 143
##   ..- attr(*, “dimnames”)=List of 2
##   .. ..$ word: chr [1:2] “fouRth” “flooR”
##   .. ..$ r   : chr [1:2] “r0” “r1”
##  $ expected : num [1:2, 1:2] 262 237 121 110
##   ..- attr(*, “dimnames”)=List of 2
##   .. ..$ word: chr [1:2] “fouRth” “flooR”
##   .. ..$ r   : chr [1:2] “r0” “r1”
##  $ residuals: 'table' num [1:2, 1:2] 2.05 -2.16 -3.02 3.17
##   ..- attr(*, “dimnames”)=List of 2
##   .. ..$ word: chr [1:2] “fouRth” “flooR”
##   .. ..$ r   : chr [1:2] “r0” “r1”
##  $ stdres   : 'table' num [1:2, 1:2] 5.29 -5.29 -5.29 5.29
##   ..- attr(*, “dimnames”)=List of 2
##   .. ..$ word: chr [1:2] “fouRth” “flooR”
##   .. ..$ r   : chr [1:2] “r0” “r1”
##  - attr(*, “class”)= chr “htest”
```

O objeto `x2.word` é uma lista que contém outros valores, e que podem ser acessados por meio do operador `$`. Já usamos esse operador para acessar colunas de um dataframe. Aqui, vamos usá-lo para acessar valores da lista. O que veremos na sequência é como o R computou os valores de qui-quadrado e de graus de liberdade, para mais bem compreender essas medidas estatísticas. A rigor, você não precisa saber disso para saber aplicar o teste – que já foi feito acima. Mas é importante saber um pouco do raciocínio por trás desse teste para saber como foram computados esses valores.

Um dos itens da lista são os valores observados, que são acessados por `x2.word$observed`. Guarde os valores observados em um objeto chamado `O` (“o” maiúsculo).

```
O <- x2.word$observed
```

Faça o mesmo com os valores esperados (x2.word$expected), guardando-os num objeto chamado `E`.

```
E <- x2.word$expected
```

Inspecione agora o objeto `O`, criado acima.

```
O
```

```
##         r
## word      r0  r1
##   fouRth 295  88
##   flooR  204 143
```

Ora, nada mais é do que a tabela de frequências, `tab.word`, que você já havia criado, não? Inspecione agora o objeto `E`.

```
E
```

```
##         r
## word           r0       r1
##   fouRth 261.8041 121.1959
##   flooR  237.1959 109.8041
```

Esta tabela é nova. O que são esses números? Para saber como computá-los, adicione as margens na tabela de valores observados `O`, por meio da função `addmargins()`.

```
addmargins(O)
```

```
##         r
## word      r0  r1 Sum
##   fouRth 295  88 383
##   flooR  204 143 347
##   Sum    499 231 730
```

Com `addmargins()`, temos os totais de linhas e colunas. Os valores esperados são obtidos por meio da conta: (T-Linha * T-Coluna) / T-Geral. Rode o comando (499 * 383) / 730 – ou seja, o total da primeira coluna vezes o total da primeira linha, dividido pelo total geral de dados – para ver o resultado.

```
(499 * 383) / 730
```

```
## [1] 261.8041
```

O valor corresponde exatamente àquele na primeira linha e primeira coluna da tabela E. Compute agora o valor esperado para a segunda linha da primeira coluna.

```
(499 * 347) / 730
```

```
## [1] 237.1959
```

Calcule o valor esperado para a primeira linha da segunda coluna.

```
(231 * 383) / 730
## [1] 121.1959
```

E calcule o valor esperado para a segunda linha da segunda coluna.

```
(231 * 347) / 730
## [1] 109.8041
```

Ok, agora você sabe computar os valores esperados a partir dos valores observados. O que são esses valores? Faça a divisão do valor esperado de r0 para a palavra fourth (E[1, 1]) pelo total de dados de fourth (383), linha de comando que já está no *script*.

```
E[1, 1] / 383
## [1] 0.6835616
```

Faça também a divisão do valor esperado de r0 para a palavra floor (E[2, 1]) pelo total de dados de floor (347).

```
E[2, 1] / 347
## [1] 0.6835616
```

Ambos dão o valor de 0,683561. Você se lembra onde já viu esse número?

Sim, é a proporção geral de r0 na amostra! Os valores esperados são justamente aqueles que se esperariam caso não houvesse diferença entre as proporções (no exemplo, entre as proporções de r0 nas palavras fourth e floor) – que seriam, então, iguais à proporção geral de r0 na amostra (68%).

O teste de qui-quadrado compara valores observados com valores esperados de acordo com a hipótese nula. Foi o mesmo que fizemos no teste de proporções, certo? Aliás, essencialmente, *todo* teste estatístico faz isso: compara valores observados com valores esperados.

O valor de qui-quadrado é calculado a partir dos valores observados e esperados, por meio da fórmula na sequência. Não se assuste com a notação matemática, pois essa fórmula não é nada complicada. Ela define o valor de qui-quadrado como a soma das

diferenças entre valores observados e esperados elevadas ao quadrado, divididas pelos valores esperados.

$$\chi^2 = \Sigma \frac{(O-E)^2}{E}$$

No R, qual fórmula a seguir expressa a fórmula do qui-quadrado?

- `(O - E)^2 / E`
- `sum(O ^ 2 - E ^ 2 / E)`
- `sum((O - E) ^ 2 / E)`
- `sum(O - E)^2 / E`

Acima, havíamos definido os valores observados e esperados nos objetos `O` e `E` respectivamente. Portanto, podemos aplicar a fórmula `sum((O - E) ^ 2 / E)` para calcular o valor de qui-quadrado. Faça isso agora.

```
sum((O - E) ^ 2 / E)
```

```
## [1] 27.98314
```

O R fornece o valor de qui-quadrado = 27,98314. Como esse valor se compara com aquele do teste feito acima? Inspecione novamente o objeto `x2.word`.

```
x2.word
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tab.word
## X-squared = 27.147, df = 1, p-value = 1.886e-07
```

No teste, o valor de qui-quadrado foi de 27,147, e no nosso teste foi 27,983. Parecido, mas não igual. Isso porque o teste de qui-quadrado no R tem como valor *default* uma correção feita para tabelas 2 x 2 (2 linhas e 2 colunas), como é o caso do nosso exemplo. Esse argumento pode ser definido como correct = F para que o R não faça a correção. Aplique a função `chisq.test()` sem correção da medida de qui-quadrado à tabela `tab.word`. Não se preocupe em guardar o resultado.

```
chisq.test(tab.word, correct = F)
```

```
##
##  Pearson's Chi-squared test
##
```

```
## data:  tab.word
## X-squared = 27.983, df = 1, p-value = 1.224e-07
```

O mesmíssimo valor que computamos! Agora você sabe que o valor de qui-quadrado é uma medida da diferença entre valores observados e esperados em uma distribuição. Quanto mais próximo o qui-quadrado estiver de zero, mais os valores observados se aproximam dos valores esperados – e, portanto, maior a chance de se ter observado tal distribuição em caso de a hipótese nula ser verdadeira (i.e. maiores valores de *p*!). No entanto, a interpretação do valor de qui-quadrado depende dos graus de liberdade, pois tabelas maiores tendem a gerar valores de qui-quadrado maiores.

O cálculo dos graus de liberdade é bastante simples. A fórmula é (n-linhas - 1) * (n-colunas - 1). Faça essa conta para nossa tabela de frequências `tab.word`, que tem 2 linhas e 2 colunas.

```
(2 - 1) * (2 - 1)
```

```
## [1] 1
```

Para tabelas 2 x 2, os graus de liberdade são sempre 1. Você pode entender o valor de graus de liberdade como o número de células de que você precisa, junto com os valores totais de linhas e colunas, para conseguir deduzir os demais valores. Isso está ilustrado na Tabela 9.1. Com apenas uma das células e os totais, você conseguiria definir quais foram as demais frequências.

Tabela 9.1: Graus de liberdade em tabelas 2 x 2.

	r0	r1	soma
flooR	?	?	347
fouRth	295	?	383
soma	499	231	730

Fonte: própria.

Com os valores de qui-quadrado e graus de liberdade, o pesquisador pode consultar uma tabela de distribuição de qui-quadrado para determinar o valor-*p*, a

probabilidade de se ter observado tal distribuição em caso de a hipótese nula ser verdadeira.

A Tabela 9.2 é uma tabela de qui-quadrado. As linhas apresentam os valores de qui-quadrado de 1 a 10 graus de liberdade; as colunas indicam as probabilidades associadas a cada valor de qui-quadrado.

Tabela 9.2: Tabela de distribuição de qui-quadrado.

	Probabilidade										
df	0,95	0,90	0,80	0,70	0,50	0,30	0,20	0,10	0,05	0,01	0,001
1	0,004	0,02	0,06	0,15	0,46	1,07	1,64	2,71	3,84	6,64	10,83
2	0,10	0,21	0,45	0,71	1,39	2,41	3,22	4,60	5,99	9,21	13,82
3	0,35	0,58	1,01	1,42	2,37	3,66	4,64	6,25	7,82	11,34	16,27
4	0,71	1,06	1,65	2,20	3,36	4,88	5,99	7,78	9,49	13,28	18,47
5	1,14	1,61	2,34	3,00	4,35	6,06	7,29	9,24	11,07	15,09	20,52
6	1,63	2,20	3,07	3,83	5,35	7,23	8,56	10,64	12,59	16,81	22,46
7	2,17	2,83	3,82	4,67	6,35	8,38	9,80	12,02	14,07	18,48	24,32
8	2,73	3,49	4.59	5,53	7,34	9,52	11,03	13,36	15,51	20,09	26,12
9	3,32	4,17	5.38	6,39	8,34	10,66	12,24	14,68	16,92	21,67	27,88
10	3,94	4,86	6,18	7,27	9,34	11,78	15,99	15,99	18,31	23,21	29,59

Caso o R não houvesse calculado a significância pra nós, consultaríamos a primeira linha da tabela até encontrar o valor $\chi^2 = 27{,}983$. Da esquerda pra direita, vemos que os valores de qui-quadrado aumentam. O valor $\chi^2 = 27{,}983$, portanto, está à direita da última coluna. Vemos também que os valores de probabilidade, da esquerda pra direita, diminuem. Dessa forma, o valor-*p* para $\chi^2 = 27{,}983$, com 1 grau de liberdade, é menor do que 0,001. No teste de qui-quadrado, o R nos forneceu esse valor com maior precisão: $p = 1.224e\text{-}07$.

Vejamos agora outro exemplo, com uma tabela maior do que 2 x 2. No caso acima, em que testamos a diferença entre proporções para os itens floor e fourth, sabemos que, se há diferença, só pode ser entre esses itens. Mas e se estivermos tratando de uma variável com três ou mais níveis? Numa variável com os fatores A, B e C, eventuais diferenças verificadas podem estar entre A-B, A-C, B-C ou entre todos eles.

Façamos novamente os passos da análise. Primeiro, faça um novo dataframe (`prop.store`) que computa as frequências e as proporções de store por r.

```
prop.store <- ds %>%
  count(store, r) %>%
  group_by(store) %>%
  mutate(prop = prop.table(n)) %>%
  print()

## # A tibble: 6 × 4
## # Groups:   store [3]
##   store r          n   prop
##   <fct> <fct> <int>  <dbl>
## 1 Saks  r0       93 0.522
## 2 Saks  r1       85 0.478
## 3 Macys r0      211 0.628
## 4 Macys r1      125 0.372
## 5 Klein r0      195 0.903
## 6 Klein r1       21 0.0972
```

Plote o gráfico de barras (Figura 9.3).

```
ggplot(prop.store, aes(x = store, y = prop, fill = r)) +
  geom_bar(stat = "identity")
```

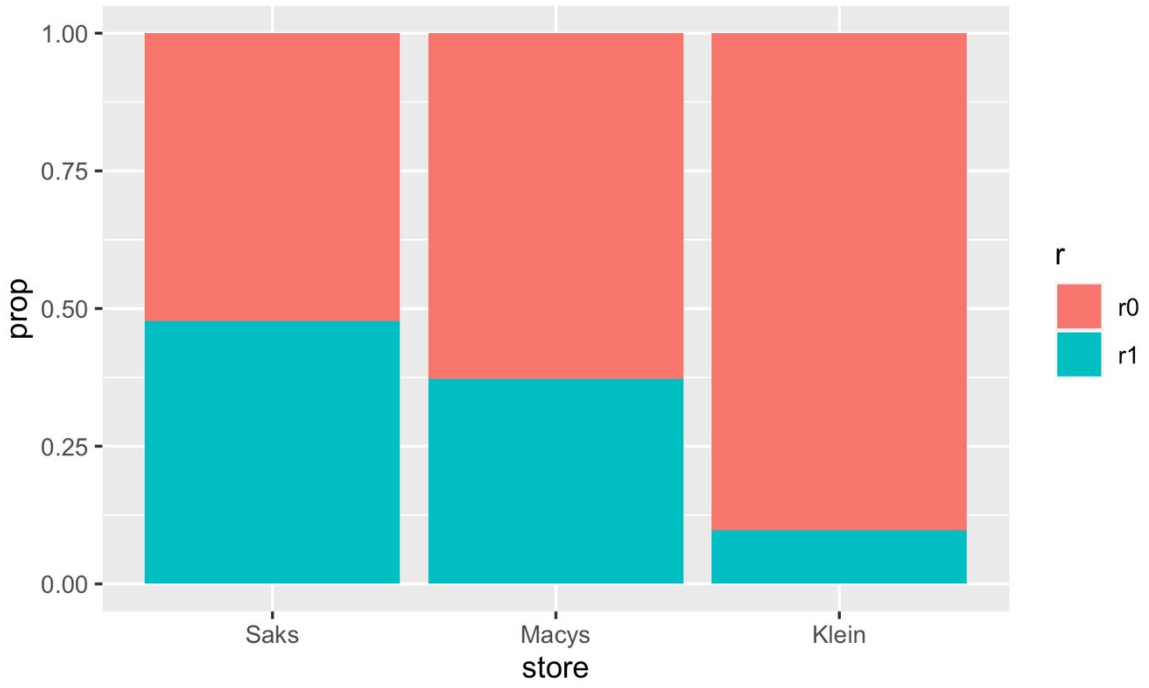

Figura 9.3: Distribuição das variantes de /r/ pós-vocálico por loja, nos dados de Labov (1972). Fonte: própria.

Crie a tabela de frequências da variável store pela VD r com as funções da instalação base do R e guarde-a num objeto chamado `tab.store`.

```
tab.store <- with(ds, table(store, r))
```

Inpecione o objeto `tab.store`.

```
tab.store
```

```
##        r
## store    r0  r1
##   Saks   93  85
##   Macys 211 125
##   Klein 195  21
```

Como já visto nas Lições 4 e 5, há mais apagamento de /r/ na S. Klein (90%), relativamente menos na Macy's (63%) e ainda menos na Saks (52%). Aplique a função de qui-quadrado à tabela `tab.store` e guarde o resultado num objeto chamado `x2.store`.

```
x2.store <- chisq.test(tab.store)
```

Veja o resultado do teste de qui-quadrado.

```
x2.store
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab.store
## X-squared = 74.142, df = 2, p-value < 2.2e-16
```

O R nos informa que o qui-quadrado é 74,1, com 2 graus de liberdade e significância menor do que 2.2e-16. Isso, contudo, é uma medida global da distribuição – lembre-se que o qui-quadrado é a *soma* das diferenças entre valores observados e esperados elevadas ao quadrado e divididas pelo valor esperado.

Vamos guardar novamente os valores observados e esperados em dois objetos. Guarde os valores observados do teste `x2.store` num objeto chamado `O`.

```
O <- x2.store$observed
```

Guarde os valores esperados do teste `x2.store` num objeto chamado `E`.

```
E <- x2.store$expected
```

Com a fórmula sum((O-E)^2/E), computamos acima o valor de qui-quadrado da tabela. Se tirarmos a função `sum()` da fórmula, teremos o valor de qui-quadrado por célula. Digite então `(O - E) ^ 2 / E` para ver o resultado.

```
(O - E) ^ 2 / E
```

```
##        r
## store          r0        r1
##   Saks   6.757375 14.597101
##   Macys  1.518742  3.280745
##   Klein 15.185220 32.802705
```

Quanto maior o valor de qui-quadrado, maior é a diferença entre o valor observado e o valor esperado. Para a distribuição dos dados de /r/ pelas lojas, vemos que a maior diferença está na S. Klein (valores acima de $\chi^2 = 15$), e a segunda maior diferença está na Saks (valores acima de $\chi^2 = 6$).

Para mais bem visualizar essa diferença, vamos fazer uma linha horizontal no gráfico de barras que indica a proporção esperada de r1, 32%. Ao comando já usado para plotar o presente gráfico, vamos adicionar a função `geom_hline()`, que plota uma linha horizontal na figura. Como argumento de `geom_hline()`, estabeleça yintercept = 0.32 (para que seja uma linha horizontal na altura 0.32).

```
ggplot(prop.store, aes(x = store, y = prop, fill = r)) +
  geom_bar(stat = "identity") +
  geom_hline(yintercept = 0.32)
```

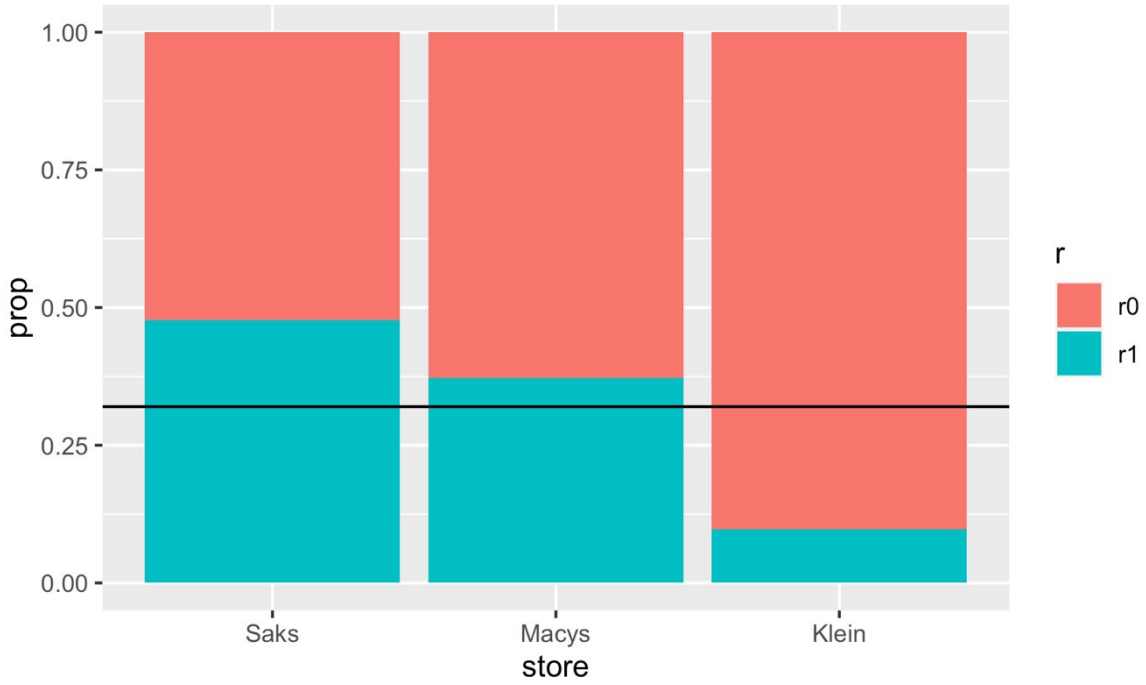

Figura 9.4: Distribuição das variantes de /r/ pós-vocálico por loja, nos dados de Labov (1972), com indicação da proporção geral de r1. Fonte: própria.

Pela , também vemos que as maiores diferenças estão nas proporções na S. Klein e na Saks, e que a proporção de r1 na Macy's se aproxima mais da proporção esperada. Os valores de qui-quadrado expressam justamente essas diferenças.

Outro modo de medir essas diferenças é pelo valor dos resíduos. Os resíduos são a diferença entre o valor observado e o valor esperado, e também foram guardados no resultado do teste. Para ver esses valores, digite `x2.store$residuals`.

```
x2.store$residuals
```

```
##        r
## store          r0        r1
##   Saks  -2.599495  3.820615
##   Macys -1.232372  1.811283
##   Klein  3.896822 -5.727365
```

Os valores de resíduos aqui são calculados pelo método de Pearson – `(O - E) / sqrt(E)`. Como esses valores não são elevados ao quadrado, mantém-se o sinal positivo ou negativo da diferença. Veja que o valor do resíduo de r1 para S. Klein é -5,727365, ou seja, o valor observado foi abaixo do esperado, enquanto os valores de resíduos de r1 para Macy's e Saks são positivos (1,811283 e 3,820615), acima da proporção esperada.

Tantos os valores de qui-quadrado por célula quanto os resíduos indicam que o maior responsável pelas diferenças são as proporções de S. Klein. Pela (e pelas estatísticas), vemos que a diferença de proporções entre Saks e Macy's é relativamente menor. Poderíamos nos perguntar se há diferença significativa entre essas duas lojas.

Acima, fizemos o teste de qui-quadrado sobre diferenças entre as três lojas por meio de `chisq.test(tab.store)`. Com que linha de comando podemos testar se há diferença apenas entre Saks e Macy's?

- `chisq.test(tab.store[1:2, ])`
- `chisq.test(tab.store[1:2])`
- `chisq.test(tab.store[1,2])`

Como `tab.store` é uma tabela, ela tem linhas e colunas, que podem ser acessadas por meio dos colchetes [ ]. Da tabela, podemos analisar apenas as duas primeiras linhas 1:2, referentes a Saks e a Macy's. Deixamos o índice de coluna vazio. Faça então o teste de qui-quadrado apenas com as proporções das lojas Saks e Macy's. Não se preocupe em guardar o resultado num objeto.

```
chisq.test(tab.store[1:2, ])
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tab.store[1:2, ]
## X-squared = 4.9323, df = 1, p-value = 0.02636
```

Pelo resultado, o que o pesquisador pode concluir?

- Há diferença significativa entre as proporções de r0 de Macys e de Saks.
- Não há diferença significativa entre as proporções de r0 de Macys e de Saks.

Qual é o valor de qui-quadrado?

```
4.9323
```

```
## [1] 4.9323
```

Quantos graus de liberdade há na tabela `tab.store[1:2, ]`?

```
1
```

```
## [1] 1
```

Volte à tabela de distribuição do qui-quadrado. Em que ponto se localiza o valor χ^2 = 4,9323 para um grau de liberdade?

- entre as probabilidades 0,01 e 0,001
- entre as probabilidades 0,05 e 0,01
- entre as probabilidades 0,10 e 0,05

Faz sentido, não? O R calculou a probabilidade 0,02636, que está justamente entre 0,05 e 0,01 com um grau de liberdade.

É importante ainda mencionar que os valores de qui-quadrado e de significância são sensíveis ao tamanho da amostra. Digamos que a amostra de dados tivesse sido 20 vezes menor. Digite `tab.store/20` para ver como ficaria a distribuição dos dados de /r/ por loja neste cenário.

```
tab.store/20
```

```
##        r
## store      r0    r1
##   Saks   4.65  4.25
##   Macys 10.55  6.25
##   Klein  9.75  1.05
```

As proporções continuariam as mesmas, pois dividimos todos os valores igualmente por 20. Mas faça o teste de qui-quadrado sobre este conjunto de dados, tab.store/20.

```
chisq.test(tab.store/20)
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab.store/20
## X-squared = 3.7071, df = 2, p-value = 0.1567
```

Embora as proporções sejam as mesmas, o resultado agora é a uma diferença não significativa entre as lojas. O motivo para isso é simples: com um menor número de dados, a chance de aleatoriedade é muito maior, de modo que é mais difícil rejeitar a hipótese nula.

Por fim, é importante saber como apresentar os resultados de testes de qui-quadrado. A notação convencional é $\chi^2 = 74{,}14(2)$, $p < 0{,}001$, que se lê: "Qui-quadrado igual a 74,14, com dois graus de liberdade e p menor do que 0,001." O símbolo χ deve ser representado pela letra grega chi e o quadrado é o número 2 sobrescrito. Ainda que o valor de significância fornecido pelo R seja um número muito menor – aqui, 2.2e-16 –, não é necessário reportar a significância com tanta precisão. Qualquer valor abaixo de 0,001 pode ser reportado com $p < 0{,}001$. Textualmente, o resultado pode ser assim descrito: "Um teste de qui-quadrado, com o objetivo de verificar se há diferenças entre as proporções de apagamento de /r/ nas três lojas de departamento em Nova Iorque, indica que há diferenças significativas entre as lojas ($\chi^2 = 74{,}14(2)$, $p < 0{,}001$)." O pesquisador então pode se dedicar à explicação de tal fato.

Para saber mais

Recomendo a leitura do capítulo 8 de Dalgaard (2008), das páginas 150–177 de Gries (2019) e do capítulo 9 de Levshina (2015).

Exercícios

1. Carregue o pacote tidyverse.
2. Defina como diretório de trabalho aquele que contém a planilha DadosRT.csv.
3. Carregue os dados da planilha DadosRT.csv num objeto chamado `dados` com a função `read_csv()`. Para tanto, defina as variáveis como factor, exceto IDADE (definida como integer), e INDICE.SOCIO e FREQUENCIA (definidas como double).
4. Reorganize os níveis da variável ORIGEM.PAIS, colocando "SPcapital" como primeiro nível. Não se esqueça de guardar o resultado no mesmo vetor.
5. Cheque os níveis da variável ORIGEM.PAIS.
6. Cheque a estrutura de dados para verificar se foram carregados corretamente.
7. A planilha DadosRT.csv contém dados sobre a variação na pronúncia de /r/ em coda silábica (como em "porta" e "mulher") na fala de 118 paulistanos. Foram excluídos os dados de apagamento de /r/ e mantidos os dados das variantes retroflexa e tepe. Aplique a função `View()` sobre o dataframe e dedique um tempo para se familiarizar com o conjunto de dados.
8. No R, ao criar um objeto, é possível visualizá-lo imediatamente colocando-se parênteses () em volta de toda a expressão. Faça uma tabela de frequência da variável dependente (VD), usando as funções da instalação base do R, e guarde-a num objeto chamado `tab.RT`. Em seguida, coloque parênteses em volta de toda a linha de comando para visualizar o conteúdo de `tab.RT`.
9. Faça uma tabela de proporções chamada `prop.RT` com as proporções da variável dependente, usando a função da instalação base do R. Envolva a linha de comando com parênteses para visualizar o conteúdo de `prop.RT`.
10. Faça um teste de proporções dos dados de VD sob a hipótese nula de que a proporção de *tepes* é igual a 80%.

11. Faça uma tabela de frequências dos dados da variável SEXO.GENERO por VD, usando as funções da instalação base do R. Guarde-a num objeto chamado `tab.sexo`. Visualize o conteúdo de `tab.sexo` imediatamente.
12. Faça uma tabela de proporções por linha dos dados da variável SEXO.GENERO por VD, usando a função da instalação base do R, e guarde-a num objeto chamado `prop.sexo`. Visualize o conteúdo de `prop.sexo` imediatamente.
13. Com uso das funções do tidyverse, faça um gráfico de barras das proporções de uso de retroflexo e de tepe por parte de homens e mulheres, e que contenha uma linha horizontal que representa a proporção de tepes na comunidade como um todo. Para tanto, com auxílio do pipe e a partir do dataframe `dados`, (i) compute as frequências da VD pela VI; (ii) agrupe os dados pela variável SEXO.GENERO; (iii) compute as proporções de retroflexo e de tepe por sexo do falante, nomeando a coluna de proporções como prop; (iv) defina os parâmetros estéticos x, y e fill dentro da função `ggplot()`; (v) use a geometria de barras apenas com o argumento stat = "identity"; e (vi) adicione a função `geom_hline()`, com argumento yintercept = 0.72.
14. Faça um teste de qui-quadrado para verificar se a diferença no uso de retroflexo entre homens e mulheres é significativa. Guarde os resultados do teste num objeto chamado `x2.sexo`.
15. Visualize o resultado do teste de qui-quadrado feito acima.
16. Qual é o valor de qui-quadrado?
17. O que significa df?
    a. degrees of freedom
    b. degrees of fame
    c. degrees of force
    d. degrees of fashion
18. Qual é o valor de significância calculado para este teste?
    a. $p < 0{,}001$
    b. $0{,}001 < p < 0{,}01$

c. $0{,}05 > p > 0{,}01$

d. $p > 0{,}1$

19. A qual conclusão o pesquisador *não* pode chegar diante desse resultado?
    a. a diferença entre homens e mulheres no uso de retroflexo não é significativa
    b. homens paulistanos tendem a usar retroflexos mais frequentemente do que as mulheres paulistanas
    c. o uso de retroflexos por parte de mulheres paulistanas é significativamente mais baixo que o dos homens paulistanos
    d. homens e mulheres da cidade de São Paulo empregam o retroflexo em proporções diferentes
20. Faça uma tabela de frequências dos dados da variável ORIGEM.PAIS por VD, usando as funções da instalação base do R, e guarde-a num objeto chamado `tab.origem`. Visualize a tabela.
21. Nesta distribuição, há quantos graus de liberdade?
22. Faça uma tabela de proporções por linha dos dados da variável ORIGEM.PAIS por VD, usando a função da instalação base do R, e guarde-a num objeto chamado `prop.origem`. Visualize a tabela.
23. Usando as funções do tidyverse, faça um gráfico de barras das proporções de uso de retroflexo e de tepe, de acordo com a origem dos pais, com as mesmas especificações do gráfico plotado para a variável SEXO.GENERO.
24. Da , quais falantes mais usam retroflexo?
    a. paulistanos filhos de paulistanos
    b. paulistanos filhos de interioranos
    c. paulistanos filhos de nordestinos
    d. paulistanos filhos de estrangeiros
    e. paulistanos filhos de pais de origem mista

25. Faça um teste de qui-quadrado sobre os dados de ORIGEM.PAIS por VD, e guarde os resultados num objeto chamado `x2.origem`. Visualize os resultados.
26. A qual conclusão um pesquisador pode chegar a partir do gráfico de barras e do resultado do teste acima?
    a. a variação na pronúncia de /r/ em coda, por parte de paulistanos, correlaciona-se com a origem dos pais
    b. paulistanos cujos pais vieram do interior são os que usam o retroflexo mais frequentemente
    c. a diferença entre o uso de retroflexo por parte de filhos de interioranos e filhos de paulistanos é significativa
    d. a proporção de uso do retroflexo por parte de filhos de estrangeiros é acima da média dos paulistanos
27. Da figura, percebe-se que as barras de proporção para paulistanos cujos pais são de origem mista e de origem paulistana são bastante próximas. Faça um teste de qui-quadrado para verificar se tal diferença é significativa. Não se preocupe em guardar o resultado.
28. A diferença das proporções de retroflexos entre os dois grupos (paulistanos cujos pais são de origem mista e paulistana) é significativa? Explique sua resposta.

Lição 10: Teste-t

Na Lição 9, vimos testes estatísticos univariados para variáveis nominais. Nesta lição, veremos um teste que se aplica a variáveis numéricas: o teste-t.

Antes de mais nada, carregue o pacote tidyverse.

```
library(tidyverse)
```

Vamos trabalhar com os dados das vogais pretônicas. Rode as linhas de comando para deixá-lo disponível, definindo como diretório de trabalho a pasta que contém o arquivo Pretonicas.csv.

```
# Definir diretório de trabalho

#setwd()

pretonicas <- read_csv("Pretonicas.csv",
                       col_types = cols(
                  AMOSTRA = col_factor(levels = c("PBSP", "SP2010")),
                  VOGAL = col_factor(levels = c("i", "e", "a", "o", "u
"))
                                          )
                       )
```

Cheque a estrutura desse dataframe.

```
str(pretonicas)

## spec_tbl_df [2,415 × 27] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ PALAVRA      : chr [1:2415] "fazer" "quatorze" "casou" "casado"
...
##  $ Transc.Fon   : chr [1:2415] "f<a>-'zer" "k<a>-'tor-ze" "k<a>-'zo
w" "k<a>-'za-do" ...
##  $ VOGAL        : Factor w/ 5 levels "i","e","a","o",..: 3 3 3 3 3
3 4 3 3 3 ...
##  $ F1           : num [1:2415] 487 686 731 621 845 ...
##  $ F2           : num [1:2415] 1666 1414 1168 1275 1574 ...
##  $ F1.NORM      : num [1:2415] 397 476 494 450 540 ...
##  $ F2.NORM      : num [1:2415] 1517 1386 1258 1314 1469 ...
##  $ CONT.PREC    : chr [1:2415] "f" "k" "k" "k" ...
##  $ CONT.SEG     : chr [1:2415] "z" "t" "z" "z" ...
##  $ VOGAL.SIL.SEG: chr [1:2415] "e" "o" "ow" "a" ...
##  $ F1.SIL.SEG   : num [1:2415] 498 462 529 842 509 ...
##  $ F2.SIL.SEG   : num [1:2415] 2001 1126 1009 1239 2351 ...
##  $ F1.SEG.NORM  : num [1:2415] 328 317 338 433 331 ...
##  $ F2.SEG.NORM  : num [1:2415] 1518 1095 1038 1149 1687 ...
##  $ VOGAL.TONICA : chr [1:2415] "e" "o" "ow" "a" ...
##  $ DIST.TONICA  : num [1:2415] 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ ESTR.SIL.PRET: chr [1:2415] “CV” “CV” “CV” “CV” ...
##  $ Begin.Time.s : num [1:2415] 20.4 20.6 33.6 36.5 40.3 ...
##  $ End.Time.s   : num [1:2415] 20.4 20.6 33.6 36.5 40.4 ...
##  $ Duration.ms  : num [1:2415] 19.1 20.2 40.7 25.2 34.7 ...
##  $ AMOSTRA      : Factor w/ 2 levels “PBSP”,”SP2010”: 1 1 1 1 1 1 1
1 1 1 ...
##  $ PARTICIPANTE : chr [1:2415] “MartaS” “MartaS” “MartaS” “MartaS”
...
##  $ SEXO         : chr [1:2415] “feminino” “feminino” “feminino” “fe
minino” ...
##  $ IDADE        : num [1:2415] 32 32 32 32 32 32 32 32 32 32 ...
##  $ IDADE.CHEGADA: num [1:2415] 18 18 18 18 18 18 18 18 18 18 ...
##  $ ANOS.SP      : num [1:2415] 14 14 14 14 14 14 14 14 14 14 ...
##  $ CONTEXTO     : chr [1:2415] “ai aqui j\u0087 tem treze ano vai f
azer quatorze” “ai aqui j\u0087 tem treze ano vai fazer quatorze” “a\u
0092 depois ele voltou a gente casou e viemos” “que l\u0087 voc\u0090
s\u0097 podia sair se fosse casado n\u008e se fosse pra” ...
##  - attr(*, “spec”)=
##   .. cols(
##   ..   PALAVRA = col_character(),
##   ..   Transc.Fon = col_character(),
##   ..   VOGAL = col_factor(levels = c(“i”, “e”, “a”, “o”, “u”), orde
red = FALSE, include_na = FALSE),
##   ..   F1 = col_double(),
##   ..   F2 = col_double(),
##   ..   F1.NORM = col_double(),
##   ..   F2.NORM = col_double(),
##   ..   CONT.PREC = col_character(),
##   ..   CONT.SEG = col_character(),
##   ..   VOGAL.SIL.SEG = col_character(),
##   ..   F1.SIL.SEG = col_double(),
##   ..   F2.SIL.SEG = col_double(),
##   ..   F1.SEG.NORM = col_double(),
##   ..   F2.SEG.NORM = col_double(),
##   ..   VOGAL.TONICA = col_character(),
##   ..   DIST.TONICA = col_double(),
##   ..   ESTR.SIL.PRET = col_character(),
##   ..   Begin.Time.s = col_double(),
##   ..   End.Time.s = col_double(),
##   ..   Duration.ms = col_double(),
##   ..   AMOSTRA = col_factor(levels = c(“PBSP”, “SP2010”), ordered =
FALSE, include_na = FALSE),
##   ..   PARTICIPANTE = col_character(),
##   ..   SEXO = col_character(),
##   ..   IDADE = col_double(),
##   ..   IDADE.CHEGADA = col_double(),
##   ..   ANOS.SP = col_double(),
##   ..   CONTEXTO = col_character()
##   .. )
##  - attr(*, “problems”)=<externalptr>
```

Nesta lição, vamos trabalhar apenas com os dados da vogal /e/ pretônica. Por vezes será nos dados apenas de paraibanos ou de paulistanos, por vezes será para ambos

os grupos. Crie então um subconjunto de dados chamado `PBSP_e`, com os dados da VOGAL /e/ na AMOSTRA PBSP.

```
PBSP_e <- filter(pretonicas, VOGAL == "e" & AMOSTRA == "PBSP")
```

Crie também um subconjunto de dados chamado `SP2010_e`, com os dados da VOGAL /e/ na AMOSTRA SP2010.

```
SP2010_e <- filter(pretonicas, VOGAL == "e" & AMOSTRA == "SP2010")
```

E crie um subconjunto de dados da vogal /e/ (com dados de PBSP e SP2010), chamado `VOGAL_e`, que vamos usar mais adiante.

```
VOGAL_e <- filter(pretonicas, VOGAL == "e")
```

Vamos agora fazer dois histogramas das medidas de F1.NORM (=altura das vogais) em uma mesma figura para compará-los. Complete a linha de comando neste ponto do *script*, substituindo, com calma, os valores df, VAR e VAR.

O dataframe é `VOGAL_e` – que contém dados de ambas as amostras –, o valor da variável em x é F1.NORM e da variável em `facet_grid()` é AMOSTRA, para que cada faceta contenha os dados de um dos grupos. A variável AMOSTRA vem antes de ~ para ordenar os gráficos um em cima do outro, e não lado a lado. O resultado se encontra na Figura 10.1.

```
ggplot(VOGAL_e, aes(x = F1.NORM)) +
  geom_histogram() +
  facet_grid(AMOSTRA ~ .)
```

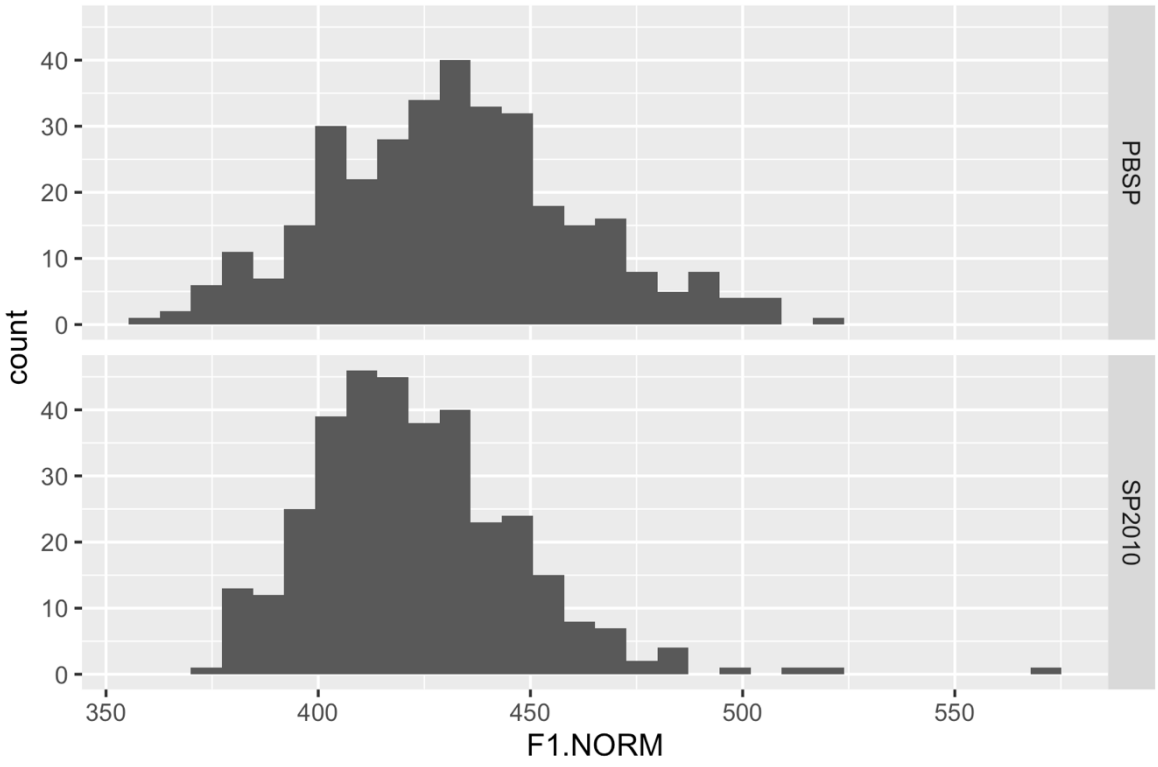

Figura 10.1: Distribuição das medidas de F1 normalizado da vogal /e/ nas amostras PBSP e SP2010. Fonte: própria.

Na figura, as medidas de F1.NORM para os paraibanos parecem ser, em geral, mais altas e mais dispersas do que para os paulistanos. Calcule a média e o desvio padrão de F1.NORM para PBSP e SP2010, e guarde os resultados num novo dataframe chamado `estatisticas`.

```
estatisticas <- VOGAL_e %>%
  group_by(AMOSTRA) %>%
  summarize(media_F1 = mean(F1.NORM),
            desvio_F1 = sd(F1.NORM)) %>%
  print()
```

```
## # A tibble: 2 × 3
##   AMOSTRA media_F1 desvio_F1
##   <fct>      <dbl>     <dbl>
## 1 PBSP        432.      29.5
## 2 SP2010      423.      24.9
```

Compare as médias de F1.NORM entre paraibanos e paulistanos: qual é maior?

- a média de F1.NORM de paraibanos
- a média de F1.NORM de paulistanos

Compare os valores de desvio padrão de F1.NORM entre paraibanos e paulistanos: qual é maior?

- o desvio padrão de F1.NORM de paraibanos
- o desvio padrão de F1.NORM de paulistanos

Como vimos na Lição 7, outro modo de visualizar a dispersão dos dados é por meio de boxplots. Faça um boxplot das medidas de F1.NORM por AMOSTRA no subconjunto de dados da VOGAL /e/ (Figura 10.2).

```
VOGAL_e %>%
  ggplot(., aes(x = AMOSTRA, y = F1.NORM)) +
  geom_boxplot(notch = TRUE) +
  scale_y_reverse()
```

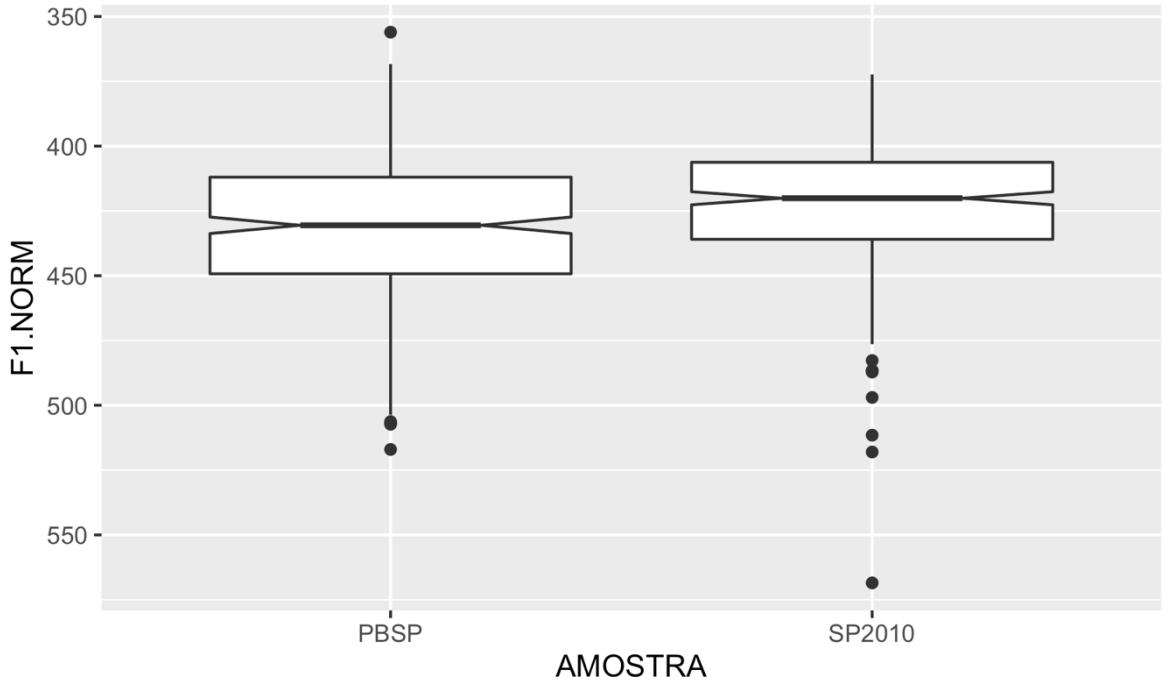

Figura 10.2: Boxplots das medidas de F1 normalizado da vogal /e/ nas amostras PBSP e SP2010. Fonte: própria.

Até aqui, refizemos os passos da Lição 7, pois esta teria sido sua trajetória de análise: começar com cálculos de medidas estatísticas básicas e visualização dos dados para estabelecer hipóteses. A partir dos histogramas e do boxplot, das medidas de média e desvio padrão, podemos querer verificar se as vogais médias pretônicas /e/ para paraibanos são significativamente mais baixas do que as dos paulistanos. Pelos entalhes

nos boxplots, temos uma boa evidência de que as médias são significativamente diferentes; cabe agora um teste estatístico para reforçar essa evidência.

O teste-t – no R, `t.test()` – compara médias e variâncias de uma distribuição com uma distribuição esperada ou com a distribuição de outro grupo. Ele tem, contudo, um requisito: a distribuição dos dados deve seguir a distribuição normal (o formato da curva de sino), ou deve ser aplicada a uma amostra com grande número de dados. Caso a distribuição não seja normal ou a amostra seja pequena, aplica-se uma variante do teste-t, chamada Teste de Wilcoxon – no R, `wilcox.test()`. Aqui, vamos seguir a primeira condição – seguir a distribuição normal ou não. O teste-t é um teste paramétrico e o teste de Wilcoxon é um teste não paramétrico.

Olhando para os histogramas (volte a eles com as flechinhas azuis), você acha que a distribuição das medidas de F1.NORM da vogal /e/ para paraibanos segue a distribuição normal?

- sim
- não
- não sei!

E para os paulistanos?

- sim
- não
- não sei!

A inspeção gráfica é um bom primeiro passo, mas deixa margem para muita dúvida. O teste de Shapiro permite obter uma medida mais acurada do quanto a distribuição se aproxima da distribuição normal. No R, ele é feito com a função `shapiro.test()`, que se aplica a um vetor de dados numéricos. Aplique a função aos dados de F1.NORM do subconjunto `PBSP_e`.

```
shapiro.test(PBSP_e$F1.NORM)

##
##  Shapiro-Wilk normality test
##
```

```
## data:  PBSP_e$F1.NORM
## W = 0.99263, p-value = 0.09217
```

E aplique-a também aos de F1.NORM do subconjunto `SP2010_e`.

```
shapiro.test(SP2010_e$F1.NORM)

##
##  Shapiro-Wilk normality test
##
## data:  SP2010_e$F1.NORM
## W = 0.94706, p-value = 8.416e-10
```

Para aplicação do teste de Shapiro, usamos a função da instalação base do R. Só por curiosidade, também seria possível fazer isso por meio do dplyr; no entanto, como o resultado do teste de Shapiro é uma lista, também seria necessário guardar o resultado em uma lista (não um dataframe). A linha de comando neste ponto do *script*, já pronta, mostra como isso poderia ser feito, aplicando-se a função `list()` sobre o teste de Shapiro. Rode-a para ver o resultado.

```
shapiro <- pretonicas %>%
  filter(VOGAL == "e") %>%
  group_by(AMOSTRA) %>%
  summarize(res = list(shapiro.test(F1.NORM)))
```

O resultado pode ser acessado por `shapiro$res`, em que `shapiro` é o objeto criado pelo comando acima e `res` é onde guardamos o resultado do teste de Shapiro. Rode essa linha de comando.

```
shapiro$res

## [[1]]
##
##  Shapiro-Wilk normality test
##
## data:  F1.NORM
## W = 0.99263, p-value = 0.09217
##
##
## [[2]]
##
##  Shapiro-Wilk normality test
##
## data:  F1.NORM
## W = 0.94706, p-value = 8.416e-10
```

O teste de Shapiro testa a hipótese *nula* que os dados vêm de uma distribuição normal. Isso significa que um $p < 0{,}05$ indica que a distribuição provavelmente *não* é

normal. Veja o resultado do teste de Shapiro nos dados de `PBSP_e`. A que conclusão o pesquisador pode chegar?

- a distribuição é normal
- a distribuição não é normal

Qual teste então pode ser aplicado aos dados de `PBSP_e`?

- `t.test()`
- `wilcox.test()`

E sobre o resultado do teste de Shapiro nos dados de `SP2010_e`?

- a distribuição é normal
- a distribuição não é normal

Qual teste então pode ser aplicado aos dados de `SP2010_e`?

- `t.test()`
- `wilcox.test()`

Façamos isso. Vamos aplicar o teste-t à distribuição de F1.NORM dos dados de `PBSP_e`. Em sua forma mais simples, assim como no teste de proporções (Lição 9), pode-se verificar se a média dos dados observados é igual a uma média esperada. Neste caso, a função `t.test()` tem dois argumentos: (i) o vetor de dados numéricos com os dados observados; e (ii) o argumento mu, que especifica o valor esperado. Suponhamos que um estudo prévio sobre vogais médias pretônicas na fala de paraibanos tenha indicado uma média de 440 Hz para a vogal /e/; vamos usar esse valor para comparação.

```
t.test(PBSP_e$F1.NORM, mu = 440)
```

```
##
##  One Sample t-test
##
## data:  PBSP_e$F1.NORM
## t = -5.1475, df = 339, p-value = 4.482e-07
## alternative hypothesis: true mean is not equal to 440
## 95 percent confidence interval:
##  428.6064 434.9065
## sample estimates:
## mean of x
##  431.7564
```

Vejamos o resultado de um teste-t. Primeiro o R informa o teste que foi realizado (teste-t de uma amostra) e, logo em seguida, o conjunto de dados (`PBSP_e$F1.NORM`). A terceira linha informa a medida t, os graus de liberdade e o valor de significância. Isso é paralelo ao teste de qui-quadrado, visto na Lição 9, com a diferença que, em vez de uma tabela de distribuição de qui-quadrado, consulta-se uma tabela de distribuição t. Como variáveis numéricas podem ter um número muito maior de graus de liberdade, ela não é mostrada aqui (como na última lição), mas você pode facilmente encontrá-la na Internet.

Logo em seguida o R enuncia a hipótese alternativa e o intervalo de confiança da distribuição (*default* = 95%). Se quisesse mudar o intervalo de confiança, bastaria especificar o argumento conf.level. Veja que o intervalo de confiança do teste acima (428,6 Hz a 434,9 Hz) não contém a média esperada (mu = 440), de modo que a hipótese nula (= a média é igual a 440) pode ser rejeitada e a hipótese alternativa (= a média não é igual a 440) pode ser acatada. Por último, o R mostra a média da amostra (431,7 Hz), que é justamente o que havíamos calculado acima para as medidas de F1.NORM da vogal /e/ entre os paraibanos.

Assim como o teste de qui-quadrado, a aplicação do teste-t é extremamente simples. Mas não importa apenas saber como aplicá-lo, mas também *quando* aplicá-lo (a uma variável numérica) e *de onde saíram* as medidas estatísticas.

Para sua curiosidade, o valor-*t* é calculado de acordo com a fórmula na sequência. $\bar{X}$ é o valor da média da amostra (para nós, `estatisticas$media_F1[1]`) e μ é a média esperada (aqui, 440); S é o valor de desvio padrão da amostra (`estatisticas$desvio_F1[1]`) e n é o número de observações.

$$t = \frac{\bar{X} - \mu}{\frac{S}{\sqrt{n}}}$$

No R, como seria escrita a fórmula da figura de acordo com nossos dados?

- `(estatisticas$media_F1[1] - 440) / (estatisticas$desvio_F1[1] / sqrt(length(PBSP_e$F1.NORM)))`
- `(estatisticas$media_F1[1] - 440) / (estatisticas$desvio_F1[1] / sqrt(PBSP))`

- `(440 - estatisticas$media_F1[1]) / (estatisticas$desvio_F1[1] / length(PBSP_e$F1.NORM))`
- `estatisticas$media_F1[1] - 440 / estatisticas$desvio_F1[1] / sqrt(length(PBSP_e$F1.NORM))`

Execute a linha de comando `(estatisticas$media_F1[1] - 440) / (estatisticas$desvio_F1[1] / sqrt(length(PBSP_e$F1.NORM)))` para calcular o valor-*t*.

```
(estatisticas$media_F1[1] - 440) / (estatisticas$desvio_F1[1] / sqrt(l
ength(PBSP_e$F1.NORM)))
```

```
## [1] -5.147498
```

Compare este valor com o valor-*t* calculado no teste-t acima. É o mesmo, certo? Quanto aos graus de liberdade, ele é o número de observações - 1. Como há 340 dados de F1.NORM da vogal /e/ para paraibanos, df = 339. (Para outros tipos de teste-t que veremos adiante, o cálculo do valor-*t* e dos graus de liberdade é um pouco diferente. Para mais informações, veja http://www.statisticshowto.com/t-test/.)

Assim como no teste de proporções, é possível estabelecer hipóteses bi ou unidirecionais. Para as vogais médias pretônicas /e/ na fala de migrantes paraibanos, a expectativa é que esses valores se aproximem mais das medidas de F1 dos paulistanos – vogais relativamente mais altas, ou F1 mais baixo. Neste caso, o pesquisador poderia estabelecer uma hipótese unidirecional: a média de F1.NORM para paraibanos em São Paulo é mais baixa do que a média de F1.NORM para paraibanos não migrantes. No R, como esta hipótese seria colocada da função `t.test()`?

- `t.test(PBSP_e$F1.NORM, mu = 440, alternative = “less”)`
- `t.test(PBSP_e$F1.NORM, mu = 440, alternative = “greater”)`
- `t.test(PBSP_e$F1.NORM, mu = 440, alternative = “two.sided”)`

Faça agora o teste unidirecional, com alternative = “less”.

```
t.test(PBSP_e$F1.NORM, mu = 440, alternative = “less”)
```

```
##
##  One Sample t-test
##
## data:  PBSP_e$F1.NORM
## t = -5.1475, df = 339, p-value = 2.241e-07
```

```
## alternative hypothesis: true mean is less than 440
## 95 percent confidence interval:
##      -Inf 434.3978
## sample estimates:
## mean of x
##  431.7564
```

Evidentemente, a escolha de um teste uni ou bidirecional depende de suas questões de pesquisa, hipóteses e expectativas. O conhecimento necessário aí é a sua formação em Linguística!

Vamos agora aplicar o teste para as vogais dos paulistanos. Lembre-se que, neste caso, como a distribuição não é normal, devemos aplicar o teste de Wilcoxon, cuja função no R é `wilcox.test()` – atenção para o nome da função, que é diferente do nome do teste! Como primeiro argumento, use o vetor de dados de F1.NORM de `SP2010_e`, e como segundo argumento uma média hipotética mu = 410.

```
wilcox.test(SP2010_e$F1.NORM, mu = 410)

##
##  Wilcoxon signed rank test with continuity correction
##
## data:  SP2010_e$F1.NORM
## V = 46146, p-value < 2.2e-16
## alternative hypothesis: true location is not equal to 410
```

O resultado da função `wilcox.test()` é mais resumido, mas a interpretação é semelhante. O teste nos informa que as médias não são iguais, com probabilidade de se ter observado tal distribuição em caso de a hipótese nula ser verdadeira abaixo de 0,001. Para esta função, o *default* é não exibir os valores do intervalo de confiança e a média da amostra; caso queira vê-los, é necessário especificar o argumento conf.int = T. Faça isso agora a partir da linha de comando acima.

```
wilcox.test(SP2010_e$F1.NORM, mu = 410, conf.int = T)

##
##  Wilcoxon signed rank test with continuity correction
##
## data:  SP2010_e$F1.NORM
## V = 46146, p-value < 2.2e-16
## alternative hypothesis: true location is not equal to 410
## 95 percent confidence interval:
##  418.9230 424.0106
## sample estimates:
```

```
## (pseudo)median
##       421.4194
```

Note, contudo, que como se trata de uma distribuição não normal, as estimativas são menos precisas. Assim como na função `t.test()`, você também poderia estabelecer uma hipótese unidirecional com o argumento alternative, e mudar o intervalo de confiança com o argumento conf.level.

Nos exemplos acima, empregamos uma média esperada mu hipoteticamente vinda de algum estudo prévio, e analisamos os dados de `PBSP_e` e `SP2010_e` separadamente. Mas as funções `t.test()` e `wilcox.test()` também podem comparar a média e a variância/desvio padrão de dois grupos, como é justamente o nosso caso. Quando se quer comparar dois grupos, o primeiro argumento de ambas as funções é uma fórmula no formato y ~ x, em que y é a variável dependente e x é a variável independente. Memorize essa notação, pois ela será vista novamente mais adiante no curso. Aqui, queremos comparar F1.NORM ~ AMOSTRA. O segundo argumento é o conjunto de dados – para nós, `VOGAL_e`.

Para decidir qual dos testes aplicar (teste-t ou teste de Wilcoxon), façamos um teste de Shapiro sobre os dados de F1.NORM do subconjunto `VOGAL_e`.

```
shapiro.test(VOGAL_e$F1.NORM)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  VOGAL_e$F1.NORM
## W = 0.97831, p-value = 1.505e-08
```

O teste nos informa que a distribuição dos dados não é normal e, portanto, é mais recomendado aplicar o teste de Wilcoxon. Essa decisão também poderia ter sido tomada pelo simples fato de que sabemos que a distribuição de uma das amostras (a dos paulistanos) não tem distribuição normal.

Digite então `wilcox.test(F1.NORM ~ AMOSTRA, data = VOGAL_e, conf.int = T)`.

```
wilcox.test(F1.NORM ~ AMOSTRA, data = VOGAL_e, conf.int = T)
```

```
##
##  Wilcoxon rank sum test with continuity correction
```

```
##
## data:  F1.NORM by AMOSTRA
## W = 70330, p-value = 9.2e-06
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:
##   5.181038 13.209983
## sample estimates:
## difference in location
##               9.148978
```

O resultado de um teste de Wilcoxon de duas amostras (= dois grupos) segue a mesma estrutura que já vimos acima; no entanto, uma diferença é que a estatística gerada é um valor W. Outra é o fato que a medida da comparação não são as médias em si, mas sim se a diferença entre elas é igual ou não a zero. É nesse sentido que se deve entender a hipótese alternativa (true location shift is not equal to 0) e os valores do intervalo de confiança. No teste feito acima, o intervalo de confiança, de 5,18 a 13,21, não contém zero, o que nos leva à rejeição da hipótese nula, com $p < 0{,}001$. A medida de estimativa ao final (9,15) é o valor da diferença estimada entre os grupos.

Veja que a estimativa da diferença entre as médias é um pouco diferente da diferença entre a média de PBSP e de SP2010. Digite `estatisticas$media_F1[1] - estatisticas$media_F1[2]` para comparar.

```
estatisticas$media_F1[1] - estatisticas$media_F1[2]
```

```
## [1] 8.790398
```

A diferença observada entre as médias é um pouco menor. Como os dados advêm de uma distribuição não normal, o teste de Wilcoxon ajustou a estimativa da diferença.

Por curiosidade, vamos aplicar a função `t.test()` ao mesmo conjunto de dados, para familiarização com o resultado fornecido pelo R. Digite `t.test(F1.NORM ~ AMOSTRA, data = VOGAL_e)`.

```
t.test(F1.NORM ~ AMOSTRA, data = VOGAL_e)
```

```
##
##  Welch Two Sample t-test
##
## data:  F1.NORM by AMOSTRA
## t = 4.2125, df = 660.96, p-value = 2.877e-05
## alternative hypothesis: true difference in means between group PBSP
and group SP2010 is not equal to 0
## 95 percent confidence interval:
```

```
##   4.692926 12.887869
## sample estimates:
##   mean in group PBSP mean in group SP2010
##             431.7564             422.9660
```

O teste-t de duas amostras também mede a diferença entre as médias e testa a hipótese nula que a diferença é igual a zero. Os valores do intervalo de confiança estimam que a diferença entre as medidas de F1.NORM para paraibanos e paulistanos está entre 4,7 Hz e 12,9 Hz, intervalo que não inclui zero – daí a diferença entre as médias ser significativa. Ao final, o R apresenta as médias do grupo 1 e do grupo 2.

Assim como fizemos no teste de uma amostra, podemos estabelecer uma hipótese unidirecional. Para as vogais pretônicas /e/ de paraibanos e de paulistanos, pode-se esperar que a média de F1.NORM de paraibanos seja maior que a de paulistanos, ou que a média de F1.NORM de paulistanos seja menor que a de paraibanos. Como você acha que o R determina a direção da comparação?

- pela ordem alfabética dos níveis
- pela ordem dos níveis no dataframe
- é uma ordem aleatória, que muda a cada teste

Exato! Pela ordem dos níveis, que pode ter sido definida pelo usuário ao importar os dados com `read_csv()`, ou, caso não tenham sido feitas modificações, a ordem em que os níveis aparecem no dataframe. É importante, portanto, checar a ordem dos níveis de AMOSTRA para bem interpretar o resultado do teste-t. Aplique a função `levels()` a esse vetor para termos certeza da ordem dos níveis de AMOSTRA.

```
levels(VOGAL_e$AMOSTRA)
## [1] "PBSP"   "SP2010"
```

Pela ordem no dataframe, o R vai comparar PBSP com SP2010, e não o contrário. Como esperamos que a média de F1.NORM seja maior para PBSP do que para SP2010, devemos estabelecer alternative = "greater" nas funções `wilcox.test()` ou `t.test()`. Copie a última linha de comando em que usamos a função `wilcox.test()`, cole-a no script e acrescente o argumento alternative = "greater".

```
wilcox.test(F1.NORM ~ AMOSTRA, data = VOGAL_e, conf.int = T, alternati
ve = “greater”)

##
##  Wilcoxon rank sum test with continuity correction
##
## data:  F1.NORM by AMOSTRA
## W = 70330, p-value = 4.6e-06
## alternative hypothesis: true location shift is greater than 0
## 95 percent confidence interval:
##  5.834992      Inf
## sample estimates:
## difference in location
##               9.148978
```

Fizemos isso apenas para que você tenha em conta que sempre pode realizar testes bi ou unidirecionais. Na verdade, o resultado significativo do teste unidirecional era previsível. Lembra-se do exemplo da moeda (Lição 8)? Lá vimos que o teste bidirecional (como era o caso do juiz imparcial) é mais rigoroso para rejeição da hipótese nula do que o teste unidirecional (como era o seu caso). Se a hipótese nula foi rejeitada num teste bidirecional, ela também será no teste unidirecional que prevê a direção dos dados observados.

Um terceiro tipo de teste-t/teste de Wilcoxon é de amostras pareadas. Aqui, permita-me usar um novo conjunto de dados, pois os dados de vogais pretônicas de paraibanos e paulistanos não são adequados para um teste pareado. Vamos usar um exemplo retirado do curso Statistical Inference, do swirl – outro curso que recomendo fazer –, mas que não tem nada a ver com Linguística, infelizmente.

Um exemplo linguístico em que se poderia aplicar um teste-t pareado seria testar a eficácia de determinado método de ensino. Digamos que um grupo de 10 alunos de um curso de inglês fizesse uma prova, depois passasse por uma aula, e em seguida fizesse nova prova sobre o mesmo assunto. O pesquisador quer comparar se o desempenho do aluno – medido pelas provas – melhorou depois da aula. O pesquisador tem em mãos 20 notas de provas, mas cada uma dessas notas não é uma observação independente vinda da população. É mais provável que alunos que já tinham tido uma nota boa na primeira prova continuem com uma boa nota na segunda prova. Cada nota da prova 1 está

associada a uma nota da prova 2. É aí que cabe fazer uma comparação de médias em que os dados são pareados.

No conjunto de dados não linguísticos com que vamos trabalhar, tem-se os resultados de um estudo médico sobre o sono, que testa o efeito de duas drogas que foram ministradas a 10 pacientes. Para mais informações, visite https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/sleep.html. Este conjunto de dados está disponível no dataframe sleep, que faz parte da instalação base do R. Digite `sleep` para vê-lo no Console.

```
sleep
```

```
##    extra group ID
## 1    0.7     1  1
## 2   -1.6     1  2
## 3   -0.2     1  3
## 4   -1.2     1  4
## 5   -0.1     1  5
## 6    3.4     1  6
## 7    3.7     1  7
## 8    0.8     1  8
## 9    0.0     1  9
## 10   2.0     1 10
## 11   1.9     2  1
## 12   0.8     2  2
## 13   1.1     2  3
## 14   0.1     2  4
## 15  -0.1     2  5
## 16   4.4     2  6
## 17   5.5     2  7
## 18   1.6     2  8
## 19   4.6     2  9
## 20   3.4     2 10
```

A primeira coluna mostra o resultado obtido após a aplicação das drogas; a segunda coluna indica qual foi a droga (grupo 1 ou grupo 2); e a terceira coluna identifica o paciente (de 1 a 10). Se fizermos simplesmente a média das medições da droga 1 e da droga 2, não podemos saber efetivamente se cada paciente melhorou ou não.

Para fazer um teste-t pareado, precisamos de dois vetores de mesma extensão e mesma ordem, para que os pares estejam devidamente ordenados. Crie primeiro um vetor chamado `g1`, com as medições da primeira coluna, das linhas 1 a 10.

```
g1 <- sleep[1:10, 1]
```

Crie agora um vetor chamado g2, com as medições da primeira coluna, das linhas 11 a 20.

```
g2 <- sleep[11:20, 1]
```

Aplique o teste de Shapiro a g1 para verificar se a distribuição é normal.

```
shapiro.test(g1)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  g1
## W = 0.92581, p-value = 0.4079
```

Aplique o teste de Shapiro a g2 para verificar se a distribuição é normal.

```
shapiro.test(g2)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  g2
## W = 0.9193, p-value = 0.3511
```

Como ambas as amostras têm distribuição normal, podemos aplicar `t.test()`. O teste-t pareado toma como argumentos os dois vetores e mais o argumento paired = T. Digite então `t.test(g1, g2, paired = T)` para ver o resultado.

```
t.test(g1, g2, paired = T)
```

```
##
##  Paired t-test
##
## data:  g1 and g2
## t = -4.0621, df = 9, p-value = 0.002833
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.4598858 -0.7001142
## sample estimates:
## mean of the differences
##                   -1.58
```

O R indica que a diferença é significativa (p = 0,0028), e que a diferença g1 - g2 é em média -1,58. Para comparar, veja o resultado de um teste-t caso não se especificasse que as amostras são pareadas. Digite `t.test(g1, g2)` – o que seria o mesmo que `t.test(extra ~ group, data = sleep)`.

```
t.test(g1, g2)
```

```
##
##  Welch Two Sample t-test
##
## data:  g1 and g2
## t = -1.8608, df = 17.776, p-value = 0.07939
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.3654832  0.2054832
## sample estimates:
## mean of x mean of y
##      0.75      2.33
```

Neste caso, o intervalo de confiança vai de -3,37 a 0,21, um intervalo que inclui o zero, de modo que a diferença entre as amostras poderia ter sido nula. A significância reflete isso: o valor-*p* calculado está acima de 0,05. Veja que deixar de especificar o fato de que os dados são pareados pode mudar o resultado do teste: no primeiro caso, o teste indicou que houve diferença, e no segundo caso não.

Ao reportar os resultados de um teste-t ou teste de Wilcoxon, é importante indicar qual medida estatística foi gerada (t ou W), os graus de liberdade, o valor-*p*, o tipo de teste (uma amostra, duas amostras, duas amostras pareadas), se o teste foi uni ou bicaudal, junto com os valores de médias e de erro padrão de cada distribuição. Não há uma função específica na instalação base do R para calcular o erro padrão, mas ele é facilmente calculável. Como vimos na Lição 6, essa medida é o valor do desvio padrão dividido pela raiz quadrada do número de observações – no R: `sd(x)/sqrt(length(x))`. O resultado do teste de Wilcoxon sobre a diferença entre a média de F1.NORM entre as amostras pode ser assim reportado: "Um teste de Wilcoxon bicaudal foi feito para comparar as médias de F1 normalizadas da vogal pretônica /e/ nas amostras de paraibanos e de paulistanos. Em média, a vogal /e/ de paraibanos é significativamente mais baixa (média = 431,7Hz, erro padrão = 1,60) do que a de paulistanos (média = 423Hz, erro padrão = 1,33), W = 70,330, $p < 0{,}001$.".

Vamos fazer uma pequena revisão antes de concluir esta lição. Quando se aplica um teste-t ou teste de Wilcoxon?

- quando se tem uma variável fatorial e se quer comparar as médias de dois grupos

- quando se tem uma variável nominal e se quer comparar as médias de dois grupos
- quando se tem uma variável numérica e se quer comparar as médias de dois grupos

Quando se aplica um teste-t (e não um teste de Wilcoxon)?

- quando a distribuição dos dados segue a distribuição normal
- quando a distribuição dos dados não segue a distribuição normal
- quando a distribuição dos dados segue a distribuição binomial
- quando a distribuição dos dados segue a distribuição de qui-quadrado

Além da inspeção de histogramas, como se pode determinar se uma distribuição é normal?

- por meio do teste-t, com a função `t.test()`
- por meio do teste de Shapiro, com a função `shapiro.test()`
- por meio do teste de Wilcoxon, com a função `wilcox.test()`

Quando se aplica um teste-t/teste de Wilcoxon de uma amostra?

- quando se quer comparar uma distribuição com uma média conhecida
- quando se quer comparar as distribuições entre dois grupos diferentes de nossos dados
- quando se quer comparar as distribuições entre dois grupos pareados de nossos dados

Quando se aplica um teste-t/teste de Wilcoxon de duas amostras?

- quando se quer comparar uma distribuição com uma média conhecida
- quando se quer comparar as distribuições entre dois grupos diferentes de nossos dados
- quando se quer comparar as distribuições entre dois grupos pareados de nossos dados

Quando se aplica um teste-t/teste de Wilcoxon pareado?

- quando se quer comparar uma distribuição com uma média conhecida
- quando se quer comparar as distribuições entre dois grupos diferentes de nossos dados
- quando se quer comparar as distribuições entre dois grupos pareados de nossos dados

Para saber mais

Recomendo a leitura do capítulo 5 de Dalgaard (2008) e do capítulo 5 de Levshina (2015) para reforçar os conceitos aqui aprendidos.

Exercícios

Para estes exercícios, você vai precisar do arquivo DadosRT-percepcao.csv. Os dados dessa planilha foram adaptados de um experimento de percepções sociolinguísticas sobre a pronúncia variável de /r/ em coda silábica como tepe ou como retroflexo. Todos os participantes eram habitantes da cidade de São Paulo (nativos e não nativos). A eles foram tocados quatro pequenos excertos de áudio, em que ouviam uma pessoa falando; sua tarefa era a de imaginar quem era o falante e responder um questionário sobre as características que imaginaram. Sete dessas características deveriam ser assinaladas dentro de uma escala de dez pontos, em que um extremo significa "pouco" e o outro extremo significa "muito": extrovertido; escolarizado; inteligente; formal; amigável; paulistano; ter sotaque. Havia também outra escala, em que um extremo significava morar num bairro mais periférico e o outro significava morar num bairro mais central. Os participantes foram divididos em dois grupos. Metade dos participantes ouviram os falantes Antonio e Luisa usando o retroflexo, e Daniela e Paulo usando o tepe. A outra metade ouviu os falantes Antonio e Luisa usando o tepe, e Daniela e Paulo usando o retroflexo. O interesse geral da pesquisa foi o de verificar se as percepções sobre um mesmo falante seriam alteradas ao serem ouvidos com o tepe ou com o retroflexo. Para mais detalhes, ver Oushiro (2015, 2019).

1. Carregue os dados de DadosRT-percepcao.csv em um dataframe chamado `dados`. Defina as colunas VARIANTE.OUVIDA e FALANTE como factor.
2. Cheque a estrutura dos `dados` para ver se foram carregados corretamente.
3. Aplique a função `View()` para visualizar a planilha de dados e se familiarizar com ela. Note, em especial, as colunas de variáveis numéricas: EXTROVERSAO, ESCOLARIZACAO, INTELIGENCIA etc. A planilha também contém variáveis que identificam a variante que foi ouvida, quem era o falante, e características sociais dos ouvintes/participantes.
4. Carregue o pacote tidyverse.
5. Faça um subconjunto de dados chamado `dados_R` para quando se ouviu a variante retroflexa. Veja o resultado de `str()` e de `View()` para saber como foram nomeadas a variável e a variante.
6. Faça um subconjunto de dados chamado `dados_T` para quando se ouviu a variante tepe. Veja o resultado de `str()` e de `View()` para saber como foram nomeadas a variável e a variante.
7. Faça um gráfico com dois histogramas da distribuição de "graus de paulistanidade" (PAULISTANIDADE) atribuídos aos falantes quando se ouviu o retroflexo e quando se ouviu o tepe. Para tanto, a partir do dataframe `dados`, (i) defina o parâmetro estético para x; (ii) use a geometria para histogramas com bindwidth = 1, alpha = 0.7 e cor das bordas das barras "gray"; e (iii) use `facet_grid()`, de modo que um histograma fique em cima do outro.
8. Examine os histogramas plotados. Em média, foram atribuídas maiores notas de PAULISTANIDADE quando se ouviu qual variante? Justifique sua resposta.
    a. tepe
    b. retroflexo
9. Os dados de PAULISTANIDADE quando se ouviu o retroflexo parecem seguir uma distribuição normal? Aplique o teste de Shapiro sobre esses dados.

10. De acordo com o teste acima, a distribuição das notas de PAULISTANIDADE quando se ouviu o retroflexo seguem uma distribuição normal? Justifique sua resposta.
11. Aplique o teste de Shapiro aos dados de PAULISTANIDADE quando se ouviu o tepe.
12. De acordo com o teste acima, a distribuição das notas de PAULISTANIDADE quando se ouviu o tepe seguem uma distribuição normal? Justifique sua resposta.
13. Com `ggplot()`, faça boxplots simples da distribuição dos dados de PAULISTANIDADE por VARIANTE.OUVIDA. Use o conjunto geral de `dados` e notch = T.
14. Calcule a média de notas atribuídas na escala de PAULISTANIDADE por VARIANTE.OUVIDA. Nomeie a coluna da medida como media.
15. Faça um teste estatístico para testar a hipótese nula de que a média da nota atribuída na escala de PAULISTANIDADE quando se ouviu o retroflexo é 5. Escolha o teste apropriado, segundo o resultado do teste de Shapiro, bem como o conjunto de dados. Estabeleça conf.int = T.
16. De acordo com o teste acima, qual deve ser a decisão do pesquisador?
    a. Rejeitar a hipótese alternativa e acatar a hipótese nula.
    b. Rejeitar a hipótese nula e acatar a hipótese alternativa.
    c. Refazer o teste.
17. Faça um teste estatístico para testar a hipótese nula de que média da nota atribuída na escala de PAULISTANIDADE é a mesma para quando se ouviu o tepe ou o retroflexo. Escolha o teste apropriado, segundo o resultado do teste de Shapiro, bem como o conjunto de dados. Estabeleça conf.int = T.
18. De acordo com o teste acima, a diferença entre as médias de PAULISTANIDADE para quando se ouviu tepe ou retroflexo é zero? Justifique sua resposta.
19. Qual é a diferença estimada entre as médias?
    a. 9.664e-15

b. 0

c. 1,199951

d. 1,599976

e. 1,900033

20. Com `ggplot()`, faça boxplots da distribuição das notas atribuídas ao grau de SOTAQUE por VARIANTE.OUVIDA e FALANTE. Para tanto, (i) escolha o dataframe apropriado; (ii) defina os parâmetros gráficos x e y; (iii) use a geometria de boxplots com notch = T; e (iv) use `facet_grid()` de modo que os falantes apareçam em facetas lado a lado no gráfico.
21. A figura permite comparar as notas atribuídas ao grau de SOTAQUE quando se ouviu o tepe ou o retroflexo para cada um dos quatro falantes. Para qual dos quatro falantes parece realmente não haver uma diferença significativa entre a nota atribuída a SOTAQUE para retroflexo ou tepe?
    a. Antonio
    b. Daniela
    c. Luisa
    d. Paulo
22. Na sequência, faremos quatro testes para analisar se a diferença entre as notas atribuídas ao grau de SOTAQUE é significativa a depender da VARIANTE.OUVIDA, para cada um dos quatro falantes. Como o teste se repetirá quatro vezes, é importante antes tomar uma importante medida. Qual é ela?
    a. fazer a correção de Bonferroni, ajustando o valor-*p* para 0,05/4
    b. fazer o teste-t para verificar se os dados seguem a distribuição normal
    c. fazer o teste de qui-quadrado para determinar os graus de liberdade
23. Calcule qual deve ser o valor-*p* para repetir o teste quatro vezes.
24. De acordo o valor-*p* determinado acima, qual deve ser o nível de confiança?
25. Faça um teste de Wilcoxon para testar a diferença entre as notas atribuídas ao grau de SOTAQUE a depender da VARIANTE.OUVIDA, apenas para o falante Antonio. Para tanto, com auxílio do pipe e a partir do dataframe dados, (i)

crie um subconjunto apenas dos dados do falante Antonio; e (ii) aplique o teste de Wilcoxon com a fórmula SOTAQUE ~ VARIANTE.OUVIDA e data = . (o ponto final, que indica o conjunto de dados criado à esquerda do pipe). Estabeleça o nível de confiança igual ao valor que você calculou na questão acima.

26. Faça o mesmo teste acima para a falante Daniela.
27. Faça o mesmo teste acima para o falante Paulo.
28. Faça o mesmo teste acima para a falante Luisa.
29. De acordo com os quatro testes acima, para qual ou quais falantes não há diferença significativa para o grau de SOTAQUE atribuído a depender da VARIANTE.OUVIDA?

Se quiser praticar mais, use os dados `pretonicas` ou os dados desta aula para elaborar novos testes. Para as pretônicas, por exemplo, você pode fazer e interpretar os testes para a vogal /o/. Para os dados do teste de percepções sociolinguísticas, você pode testar os efeitos de tepe/retroflexo sobre outras variáveis (ESCOLARIZACAO, CENTRALIDADE.BAIRRO etc.).

Lição 11: Correlação e Regressão

N.B.: Rode as linhas de comando a seguir antes de iniciar esta lição.

```
idade <- c(1, 2, 3, 4, 5, 5, 5, 6, 7, 8, 8, 9, 11, 12, 12)

altura <- c(60, 65, 97, 98, 100, 105, 107, 105, 119, 122,
            125, 132, 142, 147, 153)

criancas <- as.data.frame(cbind(idade, altura))
```

Na Lição 9, vimos testes estatísticos que se aplicam a variáveis nominais; na Lição 10, vimos testes que se aplicam a variáveis numéricas, quando queremos comparar dois grupos ou comparar uma média com outra média conhecida. Nesta lição, veremos outros testes estatísticos que se aplicam a variáveis numéricas, em relação a outra variável também numérica.

Você já sabe o que fazer no início da sessão: carregar pacotes necessários! Carregue o pacote tidyverse.

```
library(tidyverse)
```

E, nesta sessão, também vamos usar outro pacote! Carregue o pacote chamado GGally.

```
library(GGally)
```

Comecemos com um exemplo não linguístico. Existe correlação entre a idade e a altura de crianças?

- sim
- não
- não sei

Sabemos, por simples observação cotidiana, que sim! Vamos analisar este caso com alguns dados. Deixei disponíveis para você dois vetores numéricos no dataframe `criancas`. Inspecione-o.

```
criancas
```

```
##    idade altura
## 1      1     60
```

```
## 2       2     65
## 3       3     97
## 4       4     98
## 5       5    100
## 6       5    105
## 7       5    107
## 8       6    105
## 9       7    119
## 10      8    122
## 11      8    125
## 12      9    132
## 13     11    142
## 14     12    147
## 15     12    153
```

Esse dataframe contém as idades (em anos) e as alturas (em centímetros) de 15 crianças. Vamos agora fazer um gráfico de dispersão que relaciona idade e altura, por meio da função `geom_point()`, que já havíamos visto na Lição 7. Para as coordenadas, use `idade` para o eixo x e `altura` para o eixo y (Figura 11.1).

```
ggplot(criancas, aes(x = idade, y = altura)) +
  geom_point()
```

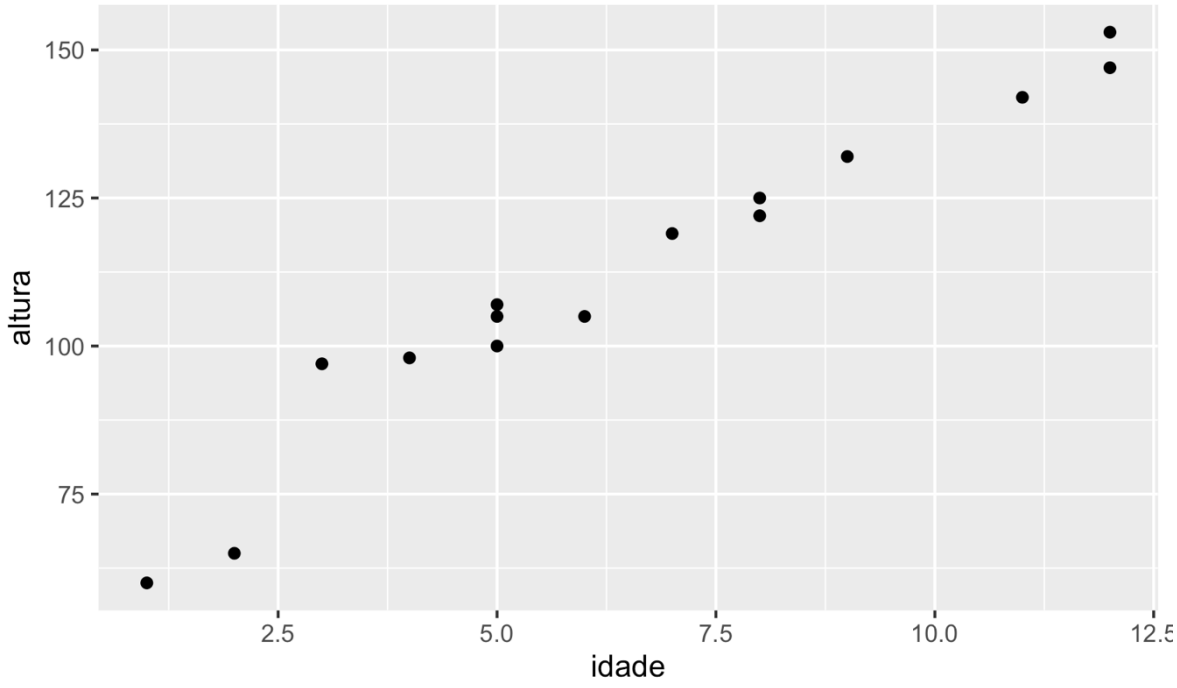

Figura 11.1: Gráfico de dispersão da altura por idade das crianças. Fonte: própria.

A figura plotada mostra claramente a correlação entre idade e altura das crianças: quanto mais velha a criança, maior sua altura. A disposição dos pontos lembra uma reta

inclinada, e é justamente a inclinação da reta que nos faz concluir que existe uma correlação entre essas duas variáveis.

A inspeção gráfica é sempre um excelente ponto de partida, mas os testes estatísticos nos permitem chegar a estimativas mais precisas sobre se há correlação ou não entre duas variáveis. Quando temos duas variáveis numéricas, o teste indicado é o teste de correlação, que no R é feito por meio da função `cor.test()`. Esta função toma como argumentos dois vetores numéricos de igual extensão. Aplique-a então aos vetores idade e altura do dataframe `criancas`.

```
cor.test(criancas$idade, criancas$altura)
```

```
##
##  Pearson's product-moment correlation
##
## data:  criancas$idade and criancas$altura
## t = 14.693, df = 13, p-value = 1.782e-09
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9132826 0.9906151
## sample estimates:
##       cor
## 0.9711854
```

Faz diferença se trocarmos a ordem dos vetores? Vamos testar. Aplique a função `cor.test()` com os vetores na ordem inversa daquela que você usou na linha anterior.

```
cor.test(criancas$altura, criancas$idade)
```

```
##
##  Pearson's product-moment correlation
##
## data:  criancas$altura and criancas$idade
## t = 14.693, df = 13, p-value = 1.782e-09
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9132826 0.9906151
## sample estimates:
##       cor
## 0.9711854
```

Vemos que o resultado é idêntico. Essa é uma primeira propriedade do teste de correlação de Pearson: ele não pressupõe uma direção da correlação; apenas avalia se duas variáveis x e y se correlacionam. Vejamos então o resultado de um teste de Pearson, cuja estrutura, a esta altura, já deve ser familiar a você.

As duas primeiras linhas indicam o teste realizado e o conjunto de dados a que se aplicou o teste. Em seguida, o R informa o valor-*t*, os graus de liberdade e o valor de significância, de modo muito semelhante ao resultado do teste-t. Numa correlação de Pearson, os graus de liberdade são o número de pares (x, y) – aqui, 15 –, menos 2: n - 2 = 13. As linhas seguintes enunciam a hipótese alternativa, o intervalo de confiança e um valor de correlação.

O valor de correlação gerado pelo teste de correlação de Pearson é chamado de *r*. Ele é calculado a partir da fórmula que aparece no *script* do Anexo A, e leva em conta o número de observações n, a soma simples e dos quadrados de x, a soma simples e dos quadrados de y, e o produto x * y. Após esta lição, separe um momento para estudá-la.

Para nossos propósitos práticos, o resultado da fórmula de *r* é sempre um número que vai de -1 a +1, que indica não só se há correlação entre x e y, mas também a força da correlação. Um valor de -1, ou próximo dele, indica uma forte correlação negativa: quanto mais x, menos y. Um valor de +1, ou próximo dele, indica uma forte correlação positiva: quanto mais x, mais y. Um valor próximo a zero indica ausência de correlação.

O nosso teste de correlação entre idade e altura das crianças dá um valor de r de Pearson igual a 0,97, o que indica forte correlação positiva entre as duas variáveis. O intervalo de confiança de 95% estima que o r de Pearson poderia ter sido entre 0,91 e 0,99, um intervalo que não inclui zero, daí o valor de significância ser abaixo de 0,05.

Assim como o teste-t, o teste de correlação tem uma versão paramétrica e uma não paramétrica, a depender de se as distribuições seguem ou não a distribuição normal. Aplique o teste de Shapiro ao vetor `idade` para verificar sua normalidade.

```
shapiro.test(criancas$idade)

##
##  Shapiro-Wilk normality test
##
## data:  criancas$idade
## W = 0.95595, p-value = 0.6225
```

Como $p > 0{,}05$, podemos assumir que a distribuição dos valores de `idade` segue a distribuição normal. Aplique agora o teste de Shapiro ao vetor `altura`.

```
shapiro.test(criancas$altura)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  criancas$altura
## W = 0.95079, p-value = 0.537
```

Ambas as variáveis seguem a distribuição normal, de modo que o teste de correlação de Pearson foi apropriado. Se este não tivesse sido o caso, a solução seria aplicar uma variante do teste de correlação, adicionando novo argumento a `cor.test()`: method = "spearman", para aplicar o teste de correlação de Spearman. Apenas como prática, aplique o teste de correlação com o método de Spearman às variáveis `idade` e `altura` para ver o resultado.

```
cor.test(criancas$idade, criancas$altura, method = "spearman")

##
##  Spearman's rank correlation rho
##
## data:  criancas$idade and criancas$altura
## S = 8.5479, p-value = 2.96e-11
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.9847359
```

Começamos esta lição com a pergunta: "Existe correlação entre a idade e a altura das crianças?", e vimos, pelo teste de correlação, que existe uma forte correlação entre elas. Mas o que significa "haver correlação" entre duas variáveis? Um dos interesses em encontrar correlações reside no fato de que, se duas variáveis se correlacionam, podemos estimar o valor de uma se temos o valor da outra. Por exemplo, quando vemos uma criança, já temos uma ideia de qual é a sua idade. De modo semelhante, também podemos estimar qual é a altura de uma criança se sabemos quantos anos ela tem.

Quando sabemos que há correlação entre duas variáveis, podemos formalizar tal conhecimento dentro de um *modelo estatístico*, que nos ajuda a chegar a estimativas mais precisas. Façamos isso com nossos dados de idade e altura das crianças. Quando duas variáveis têm uma relação linear, como é o caso dessas duas variáveis, podemos criar um *modelo linear* por meio da função `lm()` – linear model. Essa função toma como primeiro argumento uma fórmula no formato y ~ x, que já usamos na função `t.test()`. Nesta

fórmula, y é uma variável *dependente*, x é uma variável *independente* e ~ pode ser glosado como "explicado por"; em outras palavras, estamos testando se y pode ser explicado por ou depende de ou tem correlação com x. Entre altura e idade, qual é a variável *dependente*?

- altura, pois a altura depende da idade
- idade, pois a idade depende da altura

Sim, é a altura que depende da idade! Na função `cor.test()`, não precisamos definir a direção da correlação, mas na função `lm()` isso é necessário. Aplique então a função `lm()` à fórmula altura ~ idade. Como segundo argumento, informe o conjunto de dados com data = criancas. Guarde o resultado em um objeto chamado `modelo`.

```
modelo <- lm(altura ~ idade, data = criancas)
```

Vejamos o resultado da função. Digite `modelo` no Console.

```
modelo
```

```
##
## Call:
## lm(formula = altura ~ idade, data = criancas)
##
## Coefficients:
## (Intercept)        idade
##      62.504        7.545
```

O resultado de um modelo linear gera dois coeficientes: um valor de interceptação (também chamado de coeficiente linear) e um coeficiente angular. Em nosso modelo, o coeficiente linear é 62,5 e o coeficiente angular é 7,5. O que são esses números?

Para bem entendê-los, vamos dar um passeio em suas memórias das aulas de Matemática... vai aqui uma questão fácil: y + 3 = 5. Qual é o valor de y? (Desculpe-me se estou ofendendo sua inteligência... prometo que isso vai chegar a um lugar!)

- 2
- 3
- 4
- depende!

E em y - 10 = 15, qual é o valor de y?

- 25
- 30
- 35
- depende!

E em x + y = 30, qual é o valor de y?

- 2
- 20
- 40
- depende!

Quando temos duas variáveis, x e y, o valor de y depende do valor de x. A relação entre duas variáveis é chamada de função, que pode ser denotada genericamente pela expressão y = f(x).

Uma reta no plano cartesiano segue uma função de primeiro grau no formato y = a + bx. Há aí dois novos termos, a e b. O primeiro, "a", é o valor de y quando x é igual a zero (experimente colocar x = 0 na expressão y = a + bx). "a", portanto, é o ponto em que a reta intersecciona o eixo y.

O termo "b" denota em quanto aumenta ou diminui o valor de y a cada unidade de x. Se b = 0, o valor de y é uma constante (ou seja, não é uma variável); se b > 0, a reta é ascendente; se b < 0, a reta é descendente.

Os dois valores gerados em nosso modelo linear denotam justamente os valores de "a" e "b", respectivamente o coeficiente linear e o coeficiente angular. Nosso modelo então segue a função y = 62,5 + 7,5x. Vamos visualizar isso no gráfico. A linha de comando neste ponto do *script* é a mesma que rodamos anteriormente para plotar o gráfico, com a adição da função `geom_smooth()`, que usa justamente um modelo linear `lm` para plotar a linha. Rode-a agora para visualizar a Figura 11.2.

```
ggplot(criancas, aes(x = idade, y = altura)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "lightgrey")
```

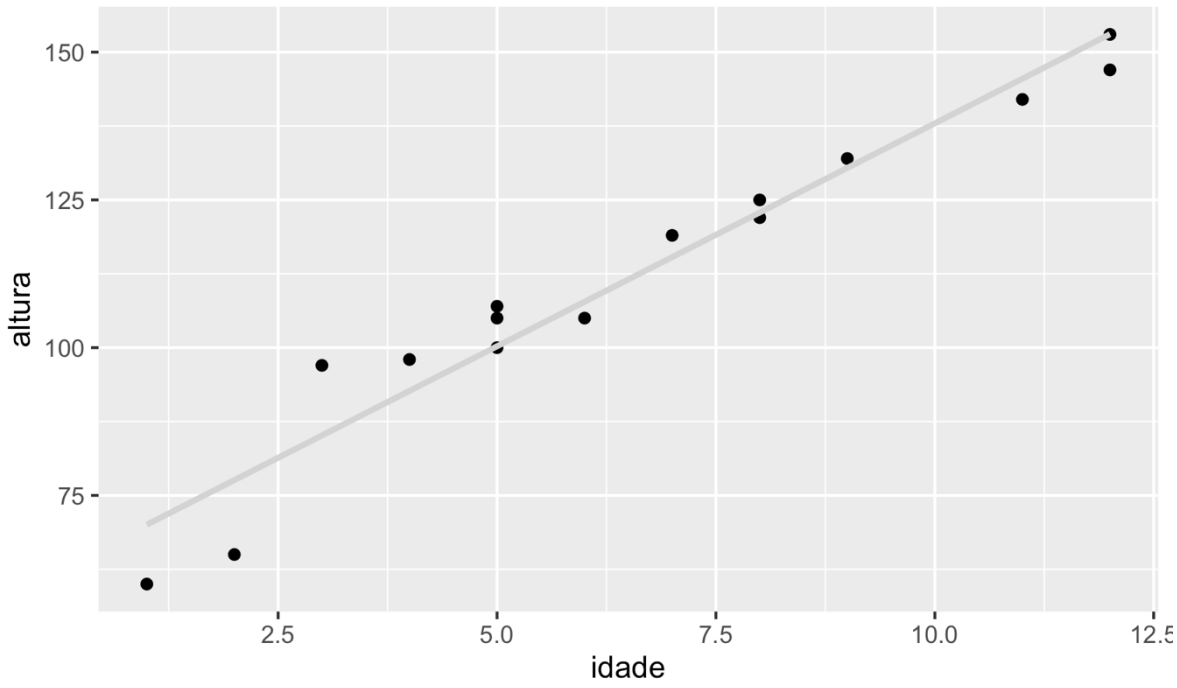

Figura 11.2: Gráfico de dispersão da altura por idade das crianças, com linha de regressão. Fonte: própria.

A linha plotada no gráfico é uma linha de regressão. Ela representa os pontos pelos quais deve passar a reta de modo que a soma das distâncias entre as observações (os pontos) e a linha seja a menor possível. Ela cruza o eixo y, altura, em 62,5cm e aumenta 7,5cm a cada unidade de x. Isso significa que nosso modelo prevê que as crianças nascem com cerca de 62,5 cm e que crescem cerca de 7,5 cm por ano.

Você pode ter achado a altura de 62,5 cm muito grande para um recém-nascido. Mas veja que isso é um modelo, não a realidade! Em parte, nosso modelo é menos preciso do que poderia ser porque estamos levando em conta apenas uma variável – a idade – para prever a altura. Certamente há outras que influenciam a altura das crianças: a altura dos pais, a alimentação etc. Adiante, no curso, veremos como incluir mais variáveis num modelo estatístico, o que faz com que os parâmetros sejam ajustados. Mas mesmo com a inclusão de outras tantas variáveis, o modelo nunca vai se equiparar com a realidade.

Um pequeno conto de Jorge Luis Borges, chamado Del Rigor en la Ciencia, ilustra bem este ponto: “En aquel Imperio, el Arte de la Cartografía logró tal Perfección que el mapa de una sola Provincia ocupaba toda una Ciudad, y el mapa del Imperio, toda una

Provincia. Con el tiempo, estos Mapas Desmesurados no satisficieron y los Colegios de Cartógrafos levantaron un Mapa del Imperio, que tenía el tamaño del Imperio y coincidía puntualmente con él. Menos Adictas al Estudio de la Cartografía, las Generaciones Siguientes entendieron que ese dilatado Mapa era Inútil y no sin Impiedad lo entregaron a las Inclemencias del Sol y los Inviernos. En los desiertos del Oeste perduran despedazadas Ruinas del Mapa, habitadas por Animales y por Mendigos; en todo el País no hay otra reliquia de las Disciplinas Geográficas. Suárez Miranda, Viajes de Varones Prudentes, Libro Cuarto, Cap. XLV, Lérida, 1658."

Um modelo é necessariamente uma representação simplificada da realidade – caso contrário, corre-se o risco de ter um modelo inútil, tal como o mapa que ocupa todo o Império. Ainda que o modelo se distancie da realidade, ele é útil para apreender o todo: para descrevê-lo, explicá-lo e fazer previsões.

Vejamos então o que mais o modelo nos fornece. No R, o resultado da função `lm()` é, minimamente, os dois coeficientes necessários para plotar a linha de regressão, mas o R também gera outros valores dentro do modelo. Aplique a função `str()` a `modelo` para ver essas outras informações.

```
str(modelo)
```

```
## List of 12
##  $ coefficients : Named num [1:2] 62.5 7.55
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "idade"
##  $ residuals    : Named num [1:15] -10.049 -12.595 11.86 5.315 -0.2
31 ...
##   ..- attr(*, "names")= chr [1:15] "1" "2" "3" "4" ...
##  $ effects      : Named num [1:15] -433 97.72 15.47 8.28 2.09 ...
##   ..- attr(*, "names")= chr [1:15] "(Intercept)" "idade" "" "" ...
##  $ rank         : int 2
##  $ fitted.values: Named num [1:15] 70 77.6 85.1 92.7 100.2 ...
##   ..- attr(*, "names")= chr [1:15] "1" "2" "3" "4" ...
##  $ assign       : int [1:2] 0 1
##  $ qr           :List of 5
##   ..$ qr   : num [1:15, 1:2] -3.873 0.258 0.258 0.258 0.258 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:15] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:2] "(Intercept)" "idade"
##   .. ..- attr(*, "assign")= int [1:2] 0 1
##   ..$ qraux: num [1:2] 1.26 1.26
##   ..$ pivot: int [1:2] 1 2
##   ..$ tol  : num 1e-07
##   ..$ rank : int 2
```

```
##   ..- attr(*, “class”)= chr “qr”
##  $ df.residual  : int 13
##  $ xlevels      : Named list()
##  $ call         : language lm(formula = altura ~ idade, data = cria
ncas)
##  $ terms        :Classes 'terms', 'formula'  language altura ~ idad
e
##   .. ..- attr(*, “variables”)= language list(altura, idade)
##   .. ..- attr(*, “factors”)= int [1:2, 1] 0 1
##   .. .. ..- attr(*, “dimnames”)=List of 2
##   .. .. .. ..$ : chr [1:2] “altura” “idade”
##   .. .. .. ..$ : chr “idade”
##   .. ..- attr(*, “term.labels”)= chr “idade”
##   .. ..- attr(*, “order”)= int 1
##   .. ..- attr(*, “intercept”)= int 1
##   .. ..- attr(*, “response”)= int 1
##   .. ..- attr(*, “.Environment”)=<environment: R_GlobalEnv>
##   .. ..- attr(*, “predvars”)= language list(altura, idade)
##   .. ..- attr(*, “dataClasses”)= Named chr [1:2] “numeric” “numeric
”
##   .. .. ..- attr(*, “names”)= chr [1:2] “altura” “idade”
##  $ model        :'data.frame':   15 obs. of  2 variables:
##   ..$ altura: num [1:15] 60 65 97 98 100 105 107 105 119 122 ...
##   ..$ idade : num [1:15] 1 2 3 4 5 5 5 6 7 8 ...
##   ..- attr(*, “terms”)=Classes 'terms', 'formula'  language altura
~ idade
##   .. .. ..- attr(*, “variables”)= language list(altura, idade)
##   .. .. ..- attr(*, “factors”)= int [1:2, 1] 0 1
##   .. .. .. ..- attr(*, “dimnames”)=List of 2
##   .. .. .. .. ..$ : chr [1:2] “altura” “idade”
##   .. .. .. .. ..$ : chr “idade”
##   .. .. ..- attr(*, “term.labels”)= chr “idade”
##   .. .. ..- attr(*, “order”)= int 1
##   .. .. ..- attr(*, “intercept”)= int 1
##   .. .. ..- attr(*, “response”)= int 1
##   .. .. ..- attr(*, “.Environment”)=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, “predvars”)= language list(altura, idade)
##   .. .. ..- attr(*, “dataClasses”)= Named chr [1:2] “numeric” “nume
ric”
##   .. .. .. ..- attr(*, “names”)= chr [1:2] “altura” “idade”
##  - attr(*, “class”)= chr “lm”
```

Assim como na função de qui-quadrado (Lição 9), o resultado da função `lm()` é uma lista com diversas medidas estatísticas, que podem ser acessadas pelo operador `$`. Digite `modelo$fitted.values` e guarde o resultado em um objeto chamado `previsao`.

```
previsao <- modelo$fitted.values
```

Visualize o vetor `previsao`.

```
previsao
```

```
##         1         2         3         4         5         6
7
##  70.04928  77.59459  85.13990  92.68521 100.23052 100.23052 100.230
52
##         8         9        10        11        12        13
14
## 107.77583 115.32114 122.86645 122.86645 130.41176 145.50238 153.047
69
##        15
## 153.04769
```

No modelo linear, os valores previstos são aqueles que coincidiriam com a linha de regressão. Na figura, vemos que a maior parte das observações não coincide exatamente com a linha. A diferença entre os valores observados e os valores previstos é o resíduo. Acesse os valores de resíduos com `modelo$residuals` e guarde-os em um objeto chamado `residuos`.

```
residuos <- modelo$residuals
```

Visualize o vetor `residuos`.

```
residuos
```

```
##            1            2            3            4            5
## -10.04928458 -12.59459459  11.86009539   5.31478537  -0.23052464
##            6            7            8            9           10
##   4.76947536   6.76947536  -2.77583466   3.67885533  -0.86645469
##           11           12           13           14           15
##   2.13354531   1.58823529  -3.50238474  -6.04769475  -0.04769475
```

Veja que os resíduos são exatamente a diferença entre o valor observado – a altura – e os valores previstos no modelo. Digite `criancas$altura - modelo$fitted.values` para obter o mesmo resultado acima.

```
criancas$altura - modelo$fitted.values
```

```
##            1            2            3            4            5
## -10.04928458 -12.59459459  11.86009539   5.31478537  -0.23052464
##            6            7            8            9           10
##   4.76947536   6.76947536  -2.77583466   3.67885533  -0.86645469
##           11           12           13           14           15
##   2.13354531   1.58823529  -3.50238474  -6.04769475  -0.04769475
```

Os primeiros dois valores são negativos porque os valores observados estão abaixo dos valores previstos. O terceiro e o quarto valores, por sua vez, são positivos, e

assim por diante. Rode a próxima linha de comando, já pronta, que plota segmentos que indicam as diferenças entre valores observados e previstos (Figura 11.3).

```
ggplot(criancas, aes(x = idade, y = altura)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "lightgrey") +
  geom_segment(aes(xend = idade, y = altura, yend = previsao), alpha =
.2)
```

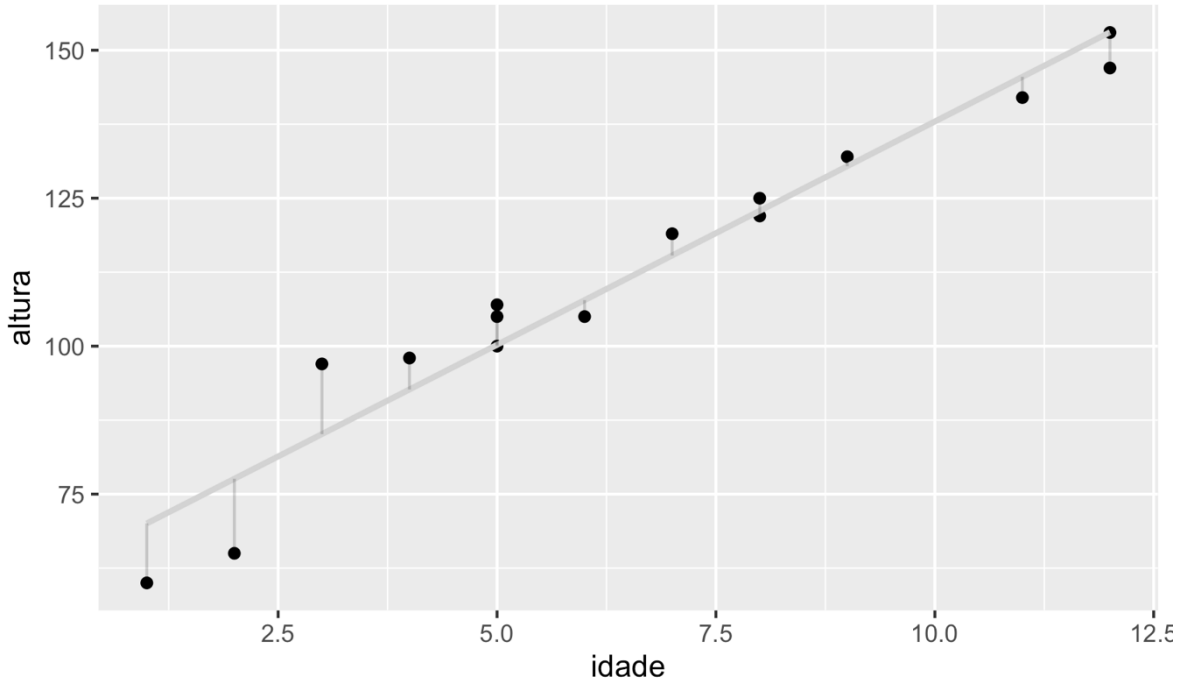

Figura 11.3: Gráfico de dispersão da altura por idade das crianças, com linha de regressão e diferenças entre valores observados e previstos. Fonte: própria.

Vamos ver os demais resultados da função linear por meio de `summary()`, para obter o resumo. Aplique essa função a `modelo`.

```
summary(modelo)
```

```
##
## Call:
## lm(formula = altura ~ idade, data = criancas)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.5946  -3.1391  -0.0477   4.2242  11.8601
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  62.5040     3.7691   16.58 3.98e-10 ***
## idade         7.5453     0.5135   14.69 1.78e-09 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.651 on 13 degrees of freedom
## Multiple R-squared:  0.9432, Adjusted R-squared:  0.9388
## F-statistic: 215.9 on 1 and 13 DF,  p-value: 1.782e-09
```

O resumo do modelo traz, em primeiro lugar, a fórmula que usamos na função `lm()` – altura ~ idade. Em seguida, traz uma visão geral dos resíduos: o valor mínimo, o primeiro quartil (1Q), a mediana, o terceiro quartil (3Q) e o valor máximo. Vimos esses termos da Lição 7, quando falamos de boxplots. Ao inspecionar os resíduos, verifique se o valor da mediana está próximo de zero e se os valores min-max e 1Q-3Q são razoavelmente simétricos. Em nosso `modelo`, estamos dentro dessa expectativa, pois a mediana (-0,05) está bem próxima de zero, os valores mínimo (-12,6) e máximo (11,8) são semelhantes em valores absolutos, assim como os valores 1Q (-3,14) e 3Q (4,22). Colocado em outros termos, nossos resíduos seguem uma distribuição normal.

Por que esperamos simetria em torno de zero para os resíduos? Os resíduos são a diferença entre os valores observados e os valores previstos pelo modelo, certo? Visto de outro modo, os resíduos são os valores que nosso modelo não foi capaz de prever perfeitamente – o que já é esperado, pois, como visto, modelos não são a realidade. Contudo, se nosso modelo "erra", é bom que ele "erre" tanto para mais quanto para menos, pois daí podemos ter certa segurança de que nossas estimativas não estão muito longe do que se poderia observar. Veja que, em nossos dados, não temos nenhuma observação para uma criança com 10 anos; no entanto, esperamos que a estimativa do modelo – i.e., y = 62,5 + (7,5 * 10) = 137,5 cm – não esteja tão longe da realidade.

Em seguida, o resumo mostra os coeficientes em uma tabela. As estimativas, na primeira coluna, já são suas conhecidas. O valor 'Intercept' é o coeficiente linear, ou seja, o valor de y quando x é zero. O valor na segunda linha é a estimativa para a idade, o coeficiente angular, que é quanto muda o valor do eixo y a cada unidade de x.

A segunda coluna se refere ao erro padrão da estimativa (ver Lições 6 e 10). Esta é uma medida da precisão das previsões: quanto menor esse valor, maior é o grau de precisão do modelo.

O valor-*t*, na terceira coluna, é computado pela divisão Estimate / Std.Error. Faça essa conta: divida o valor da estimativa do coeficiente linear, 62.5040, por seu erro padrão, 3.7691.

```
62.504 / 3.7691
```

```
## [1] 16.58327
```

Exatamente o valor-*t* calculado, certo? E faça a divisão do valor da estimativa do coeficiente angular, 7.5453, por seu erro padrão, 0.5135.

```
7.5453 / 0.5135
```

```
## [1] 14.69387
```

Como vimos na Lição 10, sobre teste-t, o valor-*t* pode ser consultado numa tabela de distribuição t para determinar a significância. Em nosso modelo, ambas as estimativas têm valor de significância $p < 0{,}05$. Mas, para poder interpretar esse valor de significância, precisamos saber qual era a hipótese alternativa e a hipótese nula sob teste!

O modelo linear testa a hipótese nula de que a estimativa tem valor zero. Em nosso modelo, 62,5 cm é bastante diferente de zero e, levando em conta o erro padrão, o valor-*p* reflete isso. Mas peraí! 62,5 cm é a estimativa do tamanho das crianças quando elas nascem. Você esperaria que um bebê nascesse com 0 centímetros? Claro que não! Você nunca levantaria essa hipótese! Isso significa que esse valor de significância, para nós, não quer dizer *na-da*!

É importante reforçar este ponto: quem faz os testes e os interpreta é o pesquisador. O R simplesmente gera os valores que está programado para gerar. No modelo linear, ele vai gerar valores de significância para avaliar o quanto cada estimativa difere de zero. Mas cabe a você saber o que é verdadeiramente relevante ou não!

Vamos para a próxima estimativa e sua significância. Aqui, o R avaliou a hipótese nula de que o coeficiente angular – 7,5 cm – difere de zero. Pare um pouco para pensar o que significaria um coeficiente zero neste modelo: isso implicaria que as crianças não crescem, pois a cada ano teriam a mesma altura; isso também implicaria que altura e idade não se correlacionam, pois a altura seria a mesma não importa a idade da criança, e não seria possível estimar o valor de uma variável a partir de outra. O valor de

significância, aqui, mostra que há uma correlação significativa entre idade e altura. Esta sim é uma medida relevante para nós.

Logo abaixo da tabela de coeficientes, o R mostra o significado dos asteriscos, segundo os níveis α mais comuns: 0,001, 0,01 e 0,05. Três asteriscos indicam $p < 0{,}001$, dois asteriscos indicam p entre 0,001 e 0,01, e um asterisco indica p entre 0,01 e 0,05. O ponto final indica um valor pouco acima de 0,05. Como já vimos na Lição 9, os valores de significância são sensíveis ao tamanho da amostra. Um valor pouco acima de 0,05 pode indicar, por exemplo, que o resultado do teste pode mudar se o pesquisador obtiver mais dados ou, ainda, se o pesquisador "limpar" os dados de valores atípicos, que podem estar causando ruído na distribuição.

Ao pé do resultado, o R mostra o valor do erro padrão dos resíduos de acordo com os graus de liberdade (cf. `sqrt(sum(modelo$residuals^2)/13)`); um valor de zero aqui significaria que o modelo prevê as observações perfeitamente (o que quase nunca é o caso).

O R também fornece dois valores chamados R^2 "R ao quadrado", o segundo dos quais "ajustado". Sua interpretação é o quanto de variabilidade na variável dependente é explicada pelas variáveis incluídas no modelo; aqui, temos que a idade explica cerca de 94% da variação em altura. Este valor nada mais é do que o valor de r de Pearson elevado ao quadrado. Você se lembra do r de Pearson calculado acima, para a mesma correlação entre idade e altura? Ele também pode ser acessado por meio de `cor.test(criancas$idade, criancas$altura)$estimate`. Faça isso agora.

```
cor.test(criancas$idade, criancas$altura)$estimate
```

```
##       cor
## 0.9711854
```

Agora calcule o quadrado da expressão acima. Copie o último comando e adicione ^ 2 à expressão acima para calcular o R^2.

```
cor.test(criancas$idade, criancas$altura)$estimate ^ 2
```

```
##      cor
## 0.943201
```

Este é o valor de R^2 do modelo. O valor de R^2 ajustado é um ajuste de R^2 a depender do número de variáveis independentes incluídas no modelo (aqui, apenas uma) e o tamanho da amostra (aqui, 15). A fórmula de R^2 ajustado está no *script* do Anexo A.

A estatística-F, por fim, é uma medida útil quando se tem interesse em comparar diferentes modelos estatísticos, a fim de determinar qual deles mais bem explica a variação na variável dependente. Ela é usada para calcular um valor de significância do modelo como um todo, para além da significância de variáveis independentes específicas.

Que tal agora aplicar esse conhecimento a fenômenos linguísticos? Rode a linha de comando a seguir para carregar o dataframe `cov`, com os dados da planilha Covariaveis.csv. Para tanto, defina como diretório de trabalho aquele que contém essa planilha em seu computador.

```
# Definir diretório de trabalho

#setwd()

# Carregar dados

cov <- read_csv("Covariaveis.csv")
```

Veja a estrutura do dataframe `cov`.

```
str(cov)

## spec_tbl_df [118 × 15] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ PARTICIPANTE : chr [1:118] "AdolfoF" "AdrianaP" "AmandaA" "Amara
lM" ...
##  $ SEXO.GENERO  : chr [1:118] "masculino" "feminino" "feminino" "ma
sculino" ...
##  $ IDADE        : num [1:118] 21 24 20 60 73 32 45 61 70 63 ...
##  $ FAIXA.ETARIA : chr [1:118] "1a" "1a" "1a" "3a" ...
##  $ ESCOLARIDADE : chr [1:118] "EnsSuperior" "EnsSuperior" "EnsSuper
ior" "EnsMedio" ...
##  $ REGIAO       : chr [1:118] "central" "central" "periferica" "per
iferica" ...
##  $ ZONA         : chr [1:118] "centro" "centro" "leste" "norte" ...
##  $ CLASSE.SOCIAL: chr [1:118] "b2" "c1" "c1" "c2-d" ...
##  $ ORIGEM.PAIS  : chr [1:118] "interior" "mista" "paulistanos" "int
erior" ...
##  $ EN           : num [1:118] 16 16 58 6 48 58 68 62 6 14 ...
##  $ RT           : num [1:118] 44 4 34 80 18 54 28 20 42 12 ...
##  $ R0           : num [1:118] 47 38 53 61 55 52 39 49 46 59 ...
##  $ CN           : num [1:118] 45 6 7 12 1 15 4 0 17 7 ...
##  $ CV3PP        : num [1:118] 8 8 19 14 7 18 8 5 24 23 ...
##  $ CV1PP        : num [1:118] 0 0 0 0 0 0 0 0 40 0 ...
```

```
##  - attr(*, "spec")=
##   .. cols(
##   ..   PARTICIPANTE = col_character(),
##   ..   SEXO.GENERO = col_character(),
##   ..   IDADE = col_double(),
##   ..   FAIXA.ETARIA = col_character(),
##   ..   ESCOLARIDADE = col_character(),
##   ..   REGIAO = col_character(),
##   ..   ZONA = col_character(),
##   ..   CLASSE.SOCIAL = col_character(),
##   ..   ORIGEM.PAIS = col_character(),
##   ..   EN = col_double(),
##   ..   RT = col_double(),
##   ..   R0 = col_double(),
##   ..   CN = col_double(),
##   ..   CV3PP = col_double(),
##   ..   CV1PP = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Veja também a planilha de dados por meio da função `View()`.

```
View(cov)
```

N.B.: Resultado aqui omitido.

`cov` contém dados de 118 falantes paulistanos, com suas respectivas características sociais e proporções de emprego de variantes de seis variáveis sociolinguísticas: (i) ditongação de /e/ nasal [~ej], como em "fazenda" (por oposição à variante monotongada [~e]; (ii) realização de /r/ em coda, como em "mulher", como retroflexo (por oposição ao tepe); (iii) apagamento de /r/ em coda (por oposição à sua realização como tepe ou retroflexo); (iv) concordância nominal não padrão, como em "os menino" (por oposição à variante padrão); (v) concordância de 3PP não padrão, como "eles foi" (por oposição à variante padrão); e (vi) concordância de 1PP não padrão, como "nós vai" (por oposição à variante padrão). Embora sejam todas variáveis nominais, elas estão aqui representadas pelas proporções de uso de uma das variantes, de modo que se tornaram numéricas.

Esses dados foram assim organizados para investigar se certas variáveis sociolinguísticas "andam juntas", ou seja, se falantes que tendem a empregar a variante x de uma variável A também tendem a empregar a variante y de uma variável B. Por exemplo: falantes que tendem a não fazer a concordância nominal padrão também

tendem a fazer a concordância não padrão de 3PP? (ver Oushiro, 2016) Visualize a distribuição desses dados no gráfico de dispersão (Figura 11.4).

```
ggplot(cov, aes(x = CV3PP, y = CN)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "lightgrey")
```

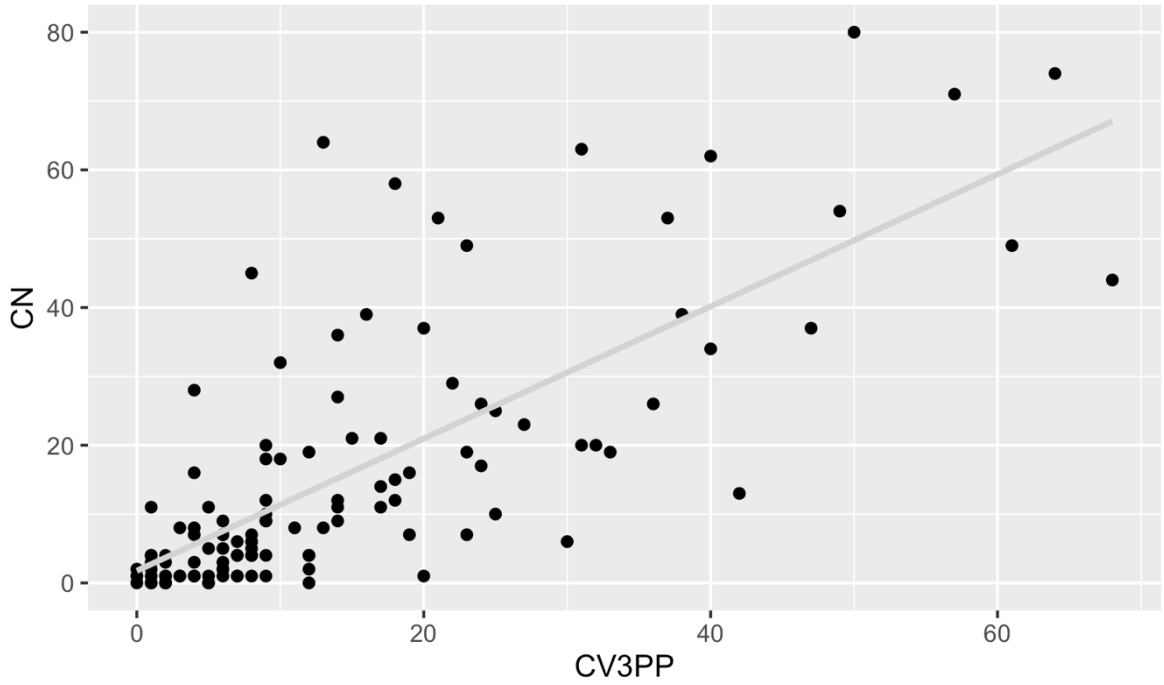

Figura 11.4: Gráfico de dispersão das proporções de uso de concordância verbal não padrão e de concordância nominal não padrão na amostra de Oushiro (2016). Fonte: própria.

Faça um teste de Shapiro para testar se a distribuição de dados de `cov$CV3PP` segue a distribuição normal.

```
shapiro.test(cov$CV3PP)

##
##  Shapiro-Wilk normality test
##
## data:  cov$CV3PP
## W = 0.81503, p-value = 7.137e-11
```

Faça um teste de Shapiro para testar se a distribuição de dados de `cov$CN` segue a distribuição normal.

```
shapiro.test(cov$CN)

##
##  Shapiro-Wilk normality test
```

```
##
## data:  cov$CN
## W = 0.78386, p-value = 6.832e-12
```

Como a distribuição de ambos não é normal, faça um teste de correlação com a adição do argumento method = "spearman".

```
cor.test(cov$CV3PP, cov$CN, method = "spearman")

##
##  Spearman's rank correlation rho
##
## data:  cov$CV3PP and cov$CN
## S = 66236, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.7581044
```

A que conclusão o pesquisador pode chegar a partir do teste acima?

- existe correlação entre o uso de CN e CV3PP
- não existe correlação entre o uso de CN e CV3PP

O teste indica que existe correlação entre o uso de CN e CV3PP, com $\rho = 0{,}75$ e $p < 0{,}001$.

Crie um modelo linear, chamado `modelo1`, com a fórmula CN ~ CV3PP como primeiro argumento, e data = cov como segundo argumento.

```
modelo1 <- lm(CN ~ CV3PP, data = cov)
```

Acima, colocamos CN ~ CV3PP, o que assume que a proporção de emprego de CN não padrão depende da proporção de emprego de CV3PP não padrão. No entanto, isso não necessariamente é verdade. Fizemos isso por exigência do formato de fórmula para o primeiro argumento de `lm()`.

Visualize o resultado de `modelo1` por meio de `summary()`.

```
summary(modelo1)

##
## Call:
## lm(formula = CN ~ CV3PP, data = cov)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.081  -6.188  -2.716   2.203  49.763
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.75595    1.64045    1.07    0.287
## CV3PP        0.96011    0.07913   12.13   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.56 on 116 degrees of freedom
## Multiple R-squared:  0.5593, Adjusted R-squared:  0.5555
## F-statistic: 147.2 on 1 and 116 DF,  p-value: < 2.2e-16
```

A que conclusão o pesquisador pode chegar a partir do teste acima?

- existe correlação entre o uso de CN e CV3PP
- não existe correlação entre o uso de CN e CV3PP

O `modelo1` prevê que, quando a proporção de emprego de CV3PP não padrão é zero, a proporção de CN não padrão é 1,8%, e que a cada 1% de aumento da proporção de emprego de CV3PP não padrão, a proporção de CN não padrão também aumenta em cerca de 1% (os valores das estimativas). Também vemos que os resíduos não se distribuem de modo muito simétrico; compare, por exemplo, o valor mínimo (-29) ao valor máximo (49). Isso pode ser visto no gráfico: diferentemente do caso da correlação entre idade e altura das crianças, aqui os pontos se dispersam bem mais e alguns deles se distanciam mais da linha de regressão – sobretudo para cima. O valor máximo de 49 é a maior distância entre o valor observado e o valor previsto pelo modelo.

Podemos repetir o teste acima para cada par de variáveis, o que daria um total de 15 testes (EN-RT, EN-R0, EN-CN etc.). Uma função muito útil para calcular uma série de correlações entre pares de variáveis é `ggpairs()`, do pacote GGally, que carregamos no início da lição.

A função `ggpairs()` toma como argumento o dataframe. É possível também definir um subconjunto de colunas, o que, em nosso caso, vai ser necessário, pois nem todas as variáveis do dataframe `cov` são numéricas. As proporções de uso das variantes /~e/ ditongado, /r/ retroflexo, apagamento de /r/, CN não padrão, CV3PP não padrão e CV1PP não padrão se encontram nas colunas 10 a 15 (Figura 11.5).

```
ggpairs(cov, columns = 10:15)
```

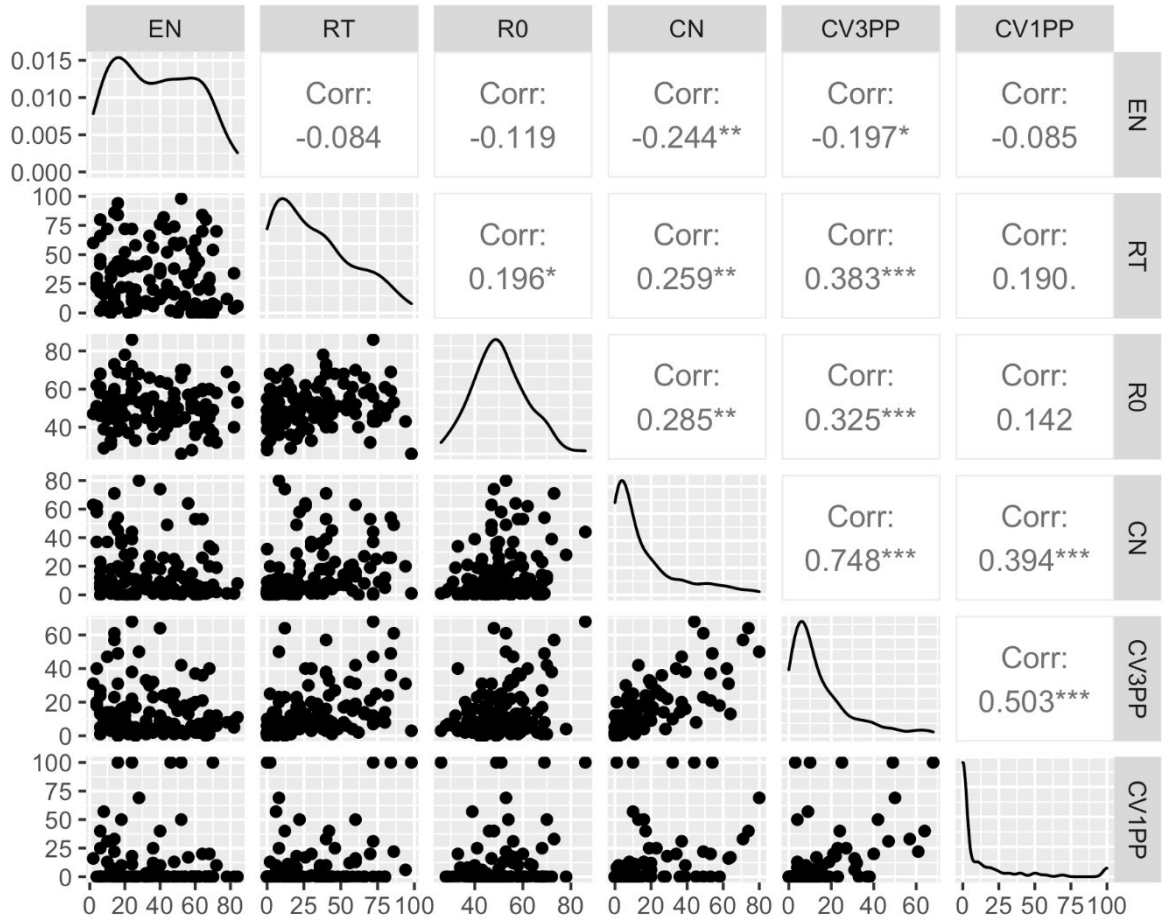

Figura 11.5: Distribuição dos dados de ditongação de /e/ nasal, /r/ retroflexo, apagamento de /r/, CN não padrão, CV3PP não padrão e CV1PP não padrão nos dados de Oushiro (2016). Fonte: própria.

A figura mostra, na linha diagonal, o gráfico de densidade das proporções das variantes indicadas nas colunas/linhas. Vemos aí claramente que a maioria das variáveis não tem uma distribuição normal. Na parte superior, a função calculou o r de Pearson, com os respectivos valores de significância, para cada par de variáveis – valores que se encontram no cruzamento das respectivas linhas. Na parte inferior, visualizam-se os gráficos de dispersão.

Da figura, qual par de variáveis apresenta a correlação mais forte?

- CN e CV3PP
- CV3PP e CV1PP
- EN e CN

- RT e R0

Da figura, qual dos pares a seguir não apresenta uma correlação significativa?

- CV3PP e CV1PP
- EN e R0
- R0 e CN
- R0 e CV3PP

Da figura, a distribuição de qual das variáveis mais se aproxima da distribuição normal?

- EN
- CN
- CV1PP
- R0

Com a função `ggpairs()`, também é possível adicionar uma variável fatorial para visualizar a distribuição para diferentes grupos. A partir da última linha de comando, acrescente o argumento `ggplot2::aes(color = REGIAO))` para definir as cores do gráfico.

```
ggpairs(cov, columns = 10:15, ggplot2::aes(color = REGIAO))
```

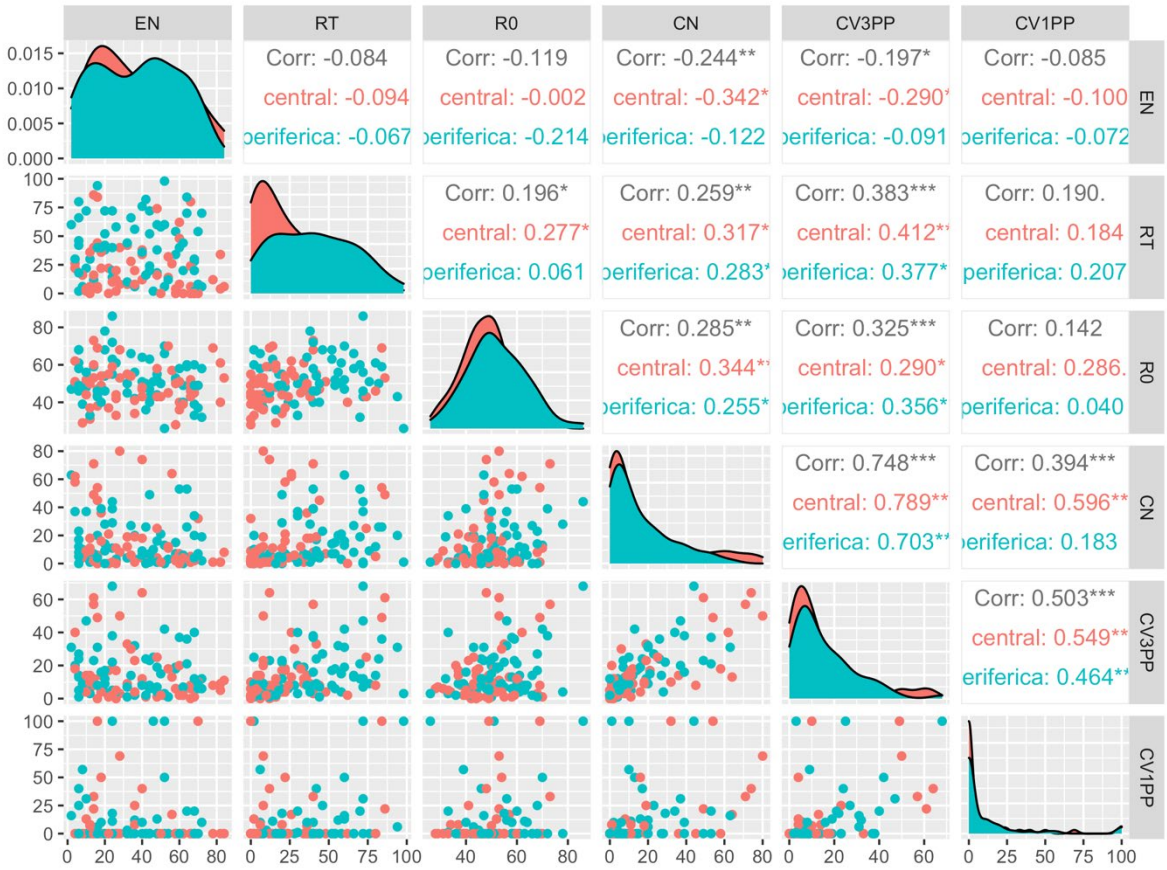

Figura 11.6: Distribuição dos dados de ditongação de /e/ nasal, /r/ retroflexo, apagamento de /r/, CN não padrão, CV3PP não padrão e CV1PP não padrão nos dados de Oushiro (2016), com falantes separados por região de residência. Fonte: própria.

A função plotou agora os gráficos de dispersão, de densidade e os valores de r de Pearson para central e periferica, o que permite comparar o comportamento desses dois grupos de falantes. Dica: clique em Zoom para ter uma melhor visualização dos dados.

Ao reportar testes de correlação, informe o valor de r de Pearson ou de Spearman, os graus de liberdade e o valor-*p*. Em modelos lineares, as medidas estatísticas do `summary()` são relevantes. Na próxima lição, veremos com mais detalhes como reportar esses dados.

Após tantos testes de correlação e modelos lineares, cabe ressaltar um mantra da Estatística: *correlação não é sinônimo de motivação*! Ainda que você tenha encontrado uma correlação significativa, seja por meio do teste de correlação, seja por meio de um modelo linear, isso não significa que y é motivado por x. A relação (ou não) entre duas

variáveis só pode ser explicada pelo pesquisador, que deve ser maximamente crítico quanto aos resultados. Um site que bem ilustra a afirmação acima é o Spurious Correlations (http://tylervigen.com/old-version.html), que traz uma série de correlações absurdas.

Os resultados de testes estatísticos não provam nada. Eles são apenas uma ferramenta auxiliar do pesquisador para explicar os fenômenos sob análise, mas todo o processo – desde o levantamento de hipóteses até seu teste e interpretação – deve ser guiado pelo bom senso.

Para saber mais

Recomendo a leitura do capítulo 6 de Dalgaard (2008), do capítulo 6 de Levshina (2015) e das páginas 138–146 de Gries (2019) para fixar o conteúdo visto aqui. No *script*, deixei um código para fazer gráficos de pares por meio do pacote languageR (Baayen, 2008).

Exercícios

Observe os gráficos da Figura 11.7 para responder as questões.

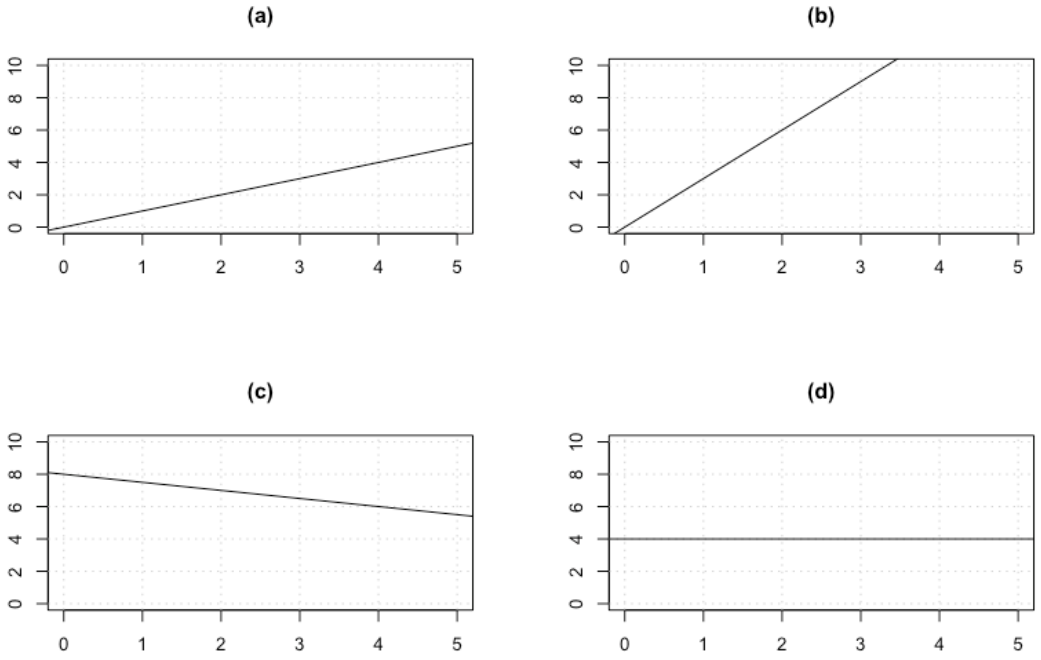

Figura 11.7: Gráficos para leitura de coeficientes linear e angular. Fonte: própria.

1. Quais são os valores dos coeficientes linear e angular na figura (a)?
    a. 0, 0
    b. 0, 1
    c. 0, 2
    d. 4, 2
2. Qual é o coeficiente angular da figura (b)?
3. Qual é o coeficiente angular da figura (c)?
4. Qual é o coeficiente linear da figura (d)?
5. Carregue o pacote tidyverse.
6. As próximas questões se baseiam nos dados de vogais médias pretônicas. Defina como diretório de trabalho a pasta que, em seu computador, tem o arquivo Pretonicas.csv.
7. Carregue os dados num dataframe chamado `pretonicas`. Defina a variável VOGAL como factor.

8. Cheque a estrutura do dataframe `pretonicas`.
9. Crie um subconjunto de dados da VOGAL "o" da AMOSTRA "PBSP", e guarde-o num dataframe chamado `PBSP_o`.
10. O abaixamento de vogais médias pretônicas, em palavras como "relógio" e "romã", é descrito como um fenômeno de harmonia vocálica: a vogal pretônica abaixa quando a vogal da sílaba seguinte é baixa [ɛ, a, ɔ]. Você tem interesse em testar a hipótese de que a altura da vogal /o/, de acordo com a medida de F1 normalizada (a variável F1.NORM), correlaciona-se com a altura da vogal da sílaba seguinte (a variável F1.SEG.NORM). O primeiro passo é plotar o gráfico de dispersão. Qual das duas variáveis deve ser colocada no eixo x (i.e., qual é a variável independente)? Justifique sua resposta.
    a. F1.SEG.NORM
    b. F1.NORM
    c. tanto faz!
11. Plote um gráfico simples de dispersão das medidas de F1.NORM e F1.SEG.NORM da vogal /o/ na fala de paraibanos com os eixos x e y devidamente especificados. Adicione a função `geom_smooth()`, com argumento method = "lm", para já plotar uma linha de regressão.
12. Você quer fazer um teste de correlação. Para decidir qual teste aplicar, o que deve fazer antes?
    a. aplicar o teste de Shapiro para verificar se as variáveis seguem a distribuição normal
    b. aplicar o teste-t para ver se as variáveis podem ser correlacionadas
    c. aplicar o teste de correlação de Pearson para determinar o r
13. Aplique o teste de Shapiro no vetor com as medidas de F1.NORM da vogal /o/.
14. Aplique o teste de Shapiro no vetor com as medidas de F1.SEG.NORM da vogal /o/.
15. Pelo resultado dos testes de Shapiro acima, qual teste é mais adequado para se aplicar aos dados?

a. teste-t
b. teste de qui-quadrado
c. teste de correlação de Spearman
d. teste de correlação de Pearson

16. Aplique o teste identificado na questão 15 para testar a hipótese de que as medidas de F1 da vogal pretônica e da vogal da sílaba seguinte se correlacionam.
17. A que conclusão o pesquisador pode chegar a partir do teste acima?
    a. não há correlação significativa entre F1.NORM e F1.SEG.NORM, com r = 0,27.
    b. há correlação significativa entre F1.NORM e F1.SEG.NORM, com r = 0,27.
    c. não há correlação significativa entre F1.NORM e F1.SEG.NORM, com r = 4,07.
    d. há correlação significativa entre F1.NORM e F1.SEG.NORM, com r = 4,69.
18. Crie um modelo linear, chamado `modelo`, para testar se há correlação entre F1.NORM e F1.SEG.NORM nos dados da vogal /o/ na fala de paraibanos.
19. Visualize o resumo dos resultados do modelo gerado.
20. De acordo com os resultados do modelo, qual afirmação seguinte não é verdadeira?
    a. A distribuição dos resíduos, neste modelo, não segue uma distribuição normal.
    b. A variável F1.SEG.NORM explica 6,7% da variação nos valores de F1.NORM da vogal /o/.
    c. Quando F1.SEG.NORM é igual a zero, a estimativa do valor de F1.NORM é 355,7 Hz.
    d. A cada unidade de F1.SEG.NORM, o valor de F1.NORM cresce 0,23 Hz.
    e. A significância da estimativa de F1.SEG.NORM está entre 0,01 e 0,05.
    f. A variável F1.SEG.NORM se correlaciona significativamente com a variável F1.NORM.
21. Qual é o valor do coeficiente linear nesse modelo?

22. Qual é o valor do coeficiente angular nesse modelo?
23. O que significa o valor-p = 4.08e-06 para F1.SEG.NORM?
    a. é baixa a probabilidade de que a verdadeira estimativa seja zero
    b. é baixa a probabilidade de que a diferença seja significativa
    c. é baixa a probabilidade que F1.SEG.NORM correlacione-se com F1.NORM
24. Aplique o teste de Shapiro aos resíduos do modelo para verificar a normalidade ou não da distribuição.
25. O que informa o teste de Shapiro sobre os resíduos do modelo?
    a. a distribuição não é normal
    b. a distribuição é normal

Lição 12: Regressão Linear Parte 1

N.B.: Rode as linhas de comando a seguir antes de iniciar esta lição. Defina como diretório de trabalho aquele que contém o arquivo Pretonicas.csv.

```
# Definir diretório de trabalho

#setwd()

# Importar planilha de dados

pretonicas <- read_csv("Pretonicas.csv",
                       col_types = cols(.default = col_factor(),
                                        VOGAL = col_factor(levels = c(
"i", "e", "a", "o", "u")),
                                        F1 = col_double(),
                                        F2 = col_double(),
                                        F1.NORM = col_double(),
                                        F2.NORM = col_double(),
                                        F1.SIL.SEG = col_double(),
                                        F2.SIL.SEG = col_double(),
                                        F1.SEG.NORM = col_double(),
                                        F2.SEG.NORM = col_double(),
                                        DIST.TONICA = col_double(),
                                        Begin.Time.s = col_double(),
                                        End.Time.s = col_double(),
                                        Duration.ms = col_double(),
                                        IDADE = col_integer(),
                                        IDADE.CHEGADA = col_integer(),
                                        ANOS.SP = col_integer()
                                        )
                       )

pretonicas$CONT.PREC <-  fct_collapse(pretonicas$CONT.PREC,
                          dental.alveolar = c("t", "d", "n", "l"),
                          labial = c("p", "b", "m", "f", "v"),
                          palatal.sibilante = c("S", "Z", "L", "s", "z
"),
                          velar = c("k", "g"),
                          vibrante = c("h", "R")
                                    )

pretonicas$CONT.PREC <- fct_relevel(pretonicas$CONT.PREC, "dental.alve
olar", "labial", "palatal.sibilante", "velar", "vibrante")

pretonicas$CONT.SEG <- fct_collapse(pretonicas$CONT.SEG,
                     dental.alveolar = c("t", "d", "n", "l"),
                     labial = c("p", "b", "m", "f", "v"),
                     palatal.sibilante = c("S", "Z", "L", "N", "s", "z
```

```
"),
                      velar = c("k", "g"),
                      vibrante = c("r", "h", "R")
                                  )

pretonicas$CONT.SEG <- fct_relevel(pretonicas$CONT.SEG, "dental.alveol
ar", "labial", "palatal.sibilante", "velar", "vibrante")

### Criar subconjunto de dados da vogal /e/ pretonica

VOGAL_e <- filter(pretonicas, VOGAL == "e") %>%
  droplevels()
```

Nas Lições 10 e 11, vimos testes estatísticos que se aplicam a variáveis dependentes numéricas: o teste-t e sua versão não paramétrica podem ser usados quando se quer comparar as distribuições de dois grupos – ou seja, aplicam-se quando se tem uma VD numérica e uma VI nominal binária (ou uma distribuição conhecida); o teste de correlação, por sua vez, aplica-se quando se tem uma VD numérica e uma VI também numérica. Esses testes só permitem testar correlações entre uma VD e uma única VI – são análises *univariadas*. Nesta e nas próximas aulas veremos quais análises podem ser feitas quando se trabalha com mais de uma variável independente – *modelos multivariados*.

Na última lição, também vimos que uma função linear pode ser denotada pela expressão y = a + bx, em que "a" é o coeficiente linear e "b" é o coeficiente angular. Análises multivariadas seguem uma estrutura semelhante, mas que inclui novos coeficientes angulares, um para cada variável independente no modelo: y = a + bx1 + cx2 + dx3... b, c e d, nesse exemplo, são coeficientes angulares das variáveis x1, x2 e x3. Em análises multivariadas, a variável dependente y é chamada de *variável resposta*, e as variáveis independentes x1, x2 etc. são chamadas de *variáveis previsoras*. Mais adiante, veremos por que os termos "variável dependente" e "variável independente" não são adequados para modelos de regressão.

Para começar, carregue o pacote tidyverse.

```
library(tidyverse)
```

Carregue também o pacote effects, que usaremos para visualizar resultados de modelos de regressão.

```
library(effects)
```

Vamos trabalhar aqui com os dados da vogal /e/ pretônica, na fala de migrantes paraibanos residentes em São Paulo e na fala de paulistanos nativos. No *script*, deixei essas linhas de comando, mas você não precisará rodá-las (ficam apenas para você saber como o dataframe foi criado). Esses dados foram guardados no df `VOGAL_e`. Aplique a função `str()` para inspecioná-lo. Em especial, veja as variáveis AMOSTRA, SEXO, F1.SEG.NORM, CONT.PREC e CONT.SEG, com as quais trabalharemos nesta e na próxima lição.

```
str(VOGAL_e)
```

```
## spec_tbl_df [686 × 27] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ PALAVRA      : Factor w/ 259 levels “diferente”,”melhor”,..: 1 1
2 3 1 4 5 6 7 8 ...
##  $ Transc.Fon   : Factor w/ 259 levels “d<i>-f<e>-'RE-te”,..: 1 1 2
3 1 4 5 6 7 8 ...
##  $ VOGAL        : Factor w/ 1 level “e”: 1 1 1 1 1 1 1 1 1 1 ...
##  $ F1           : num [1:686] 613 656 573 735 567 ...
##  $ F2           : num [1:686] 2014 1848 2413 1656 1375 ...
##  $ F1.NORM      : num [1:686] 447 464 431 496 429 ...
##  $ F2.NORM      : num [1:686] 1698 1611 1905 1512 1366 ...
##  $ CONT.PREC    : Factor w/ 5 levels “dental.alveolar”,..: 2 2 2 5
2 4 2 3 2 1 ...
##  $ CONT.SEG     : Factor w/ 5 levels “dental.alveolar”,..: 5 5 3 2
5 5 1 5 3 2 ...
##  $ VOGAL.SIL.SEG: Factor w/ 11 levels “a”,”aw”,”A”,”\u0097”,..: 5 5
4 7 5 5 1 5 1 5 ...
##  $ F1.SIL.SEG   : num [1:686] 569 524 686 652 661 ...
##  $ F2.SIL.SEG   : num [1:686] 1674 2428 1497 2159 1865 ...
##  $ F1.SEG.NORM  : num [1:686] 350 336 385 375 378 ...
##  $ F2.SEG.NORM  : num [1:686] 1360 1724 1274 1594 1452 ...
##  $ VOGAL.TONICA : Factor w/ 14 levels “e”,”o”,”ow”,”a”,..: 9 9 2 1
9 9 4 9 4 9 ...
##  $ DIST.TONICA  : num [1:686] 1 1 1 1 1 1 1 1 1 2 ...
##  $ ESTR.SIL.PRET: Factor w/ 5 levels “CV”,”CVs”,”CCV”,..: 1 1 1 1 1
1 1 1 1 1 ...
##  $ Begin.Time.s : num [1:686] 219 226 576 584 614 ...
##  $ End.Time.s   : num [1:686] 219 226 576 584 614 ...
##  $ Duration.ms  : num [1:686] 10.4 11.6 30.3 17.5 17.1 ...
##  $ AMOSTRA      : Factor w/ 2 levels “PBSP”,”SP2010”: 1 1 1 1 1 1 1
1 1 1 ...
##  $ PARTICIPANTE : Factor w/ 14 levels “MartaS”,”JosaneV”,..: 1 1 1
1 1 1 1 1 1 1 ...
##  $ SEXO         : Factor w/ 2 levels “feminino”,”masculino”: 1 1 1
1 1 1 1 1 1 1 ...
##  $ IDADE        : int [1:686] 32 32 32 32 32 32 32 32 32 32 ...
##  $ IDADE.CHEGADA: int [1:686] 18 18 18 18 18 18 18 18 18 18 ...
##  $ ANOS.SP      : int [1:686] 14 14 14 14 14 14 14 14 14 14 ...
```

```
##  $ CONTEXTO     : Factor w/ 632 levels "diferente o clima de eu com
ele",..: 1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   .default = col_factor(),
##   ..   PALAVRA = col_factor(levels = NULL, ordered = FALSE, include
_na = FALSE),
##   ..   Transc.Fon = col_factor(levels = NULL, ordered = FALSE, incl
ude_na = FALSE),
##   ..   VOGAL = col_factor(levels = c("i", "e", "a", "o", "u"), orde
red = FALSE, include_na = FALSE),
##   ..   F1 = col_double(),
##   ..   F2 = col_double(),
##   ..   F1.NORM = col_double(),
##   ..   F2.NORM = col_double(),
##   ..   CONT.PREC = col_factor(levels = NULL, ordered = FALSE, inclu
de_na = FALSE),
##   ..   CONT.SEG = col_factor(levels = NULL, ordered = FALSE, includ
e_na = FALSE),
##   ..   VOGAL.SIL.SEG = col_factor(levels = NULL, ordered = FALSE, i
nclude_na = FALSE),
##   ..   F1.SIL.SEG = col_double(),
##   ..   F2.SIL.SEG = col_double(),
##   ..   F1.SEG.NORM = col_double(),
##   ..   F2.SEG.NORM = col_double(),
##   ..   VOGAL.TONICA = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   DIST.TONICA = col_double(),
##   ..   ESTR.SIL.PRET = col_factor(levels = NULL, ordered = FALSE, i
nclude_na = FALSE),
##   ..   Begin.Time.s = col_double(),
##   ..   End.Time.s = col_double(),
##   ..   Duration.ms = col_double(),
##   ..   AMOSTRA = col_factor(levels = NULL, ordered = FALSE, include
_na = FALSE),
##   ..   PARTICIPANTE = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   SEXO = col_factor(levels = NULL, ordered = FALSE, include_na
= FALSE),
##   ..   IDADE = col_integer(),
##   ..   IDADE.CHEGADA = col_integer(),
##   ..   ANOS.SP = col_integer(),
##   ..   CONTEXTO = col_factor(levels = NULL, ordered = FALSE, includ
e_na = FALSE)
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Aplique também a função `View()` para se (re)familiarizar com a planilha.

```
View(VOGAL_e)
```

N.B.: Resultado aqui omitido.

A variável AMOSTRA é codificada de acordo com a amostra da qual vieram os dados: PBSP (migrantes paraibanos) ou SP2010 (paulistanos nativos). SEXO indica se o falante é do sexo feminino ou masculino. A variável F1.SEG.NORM contém as medições normalizadas da altura (F1) da vogal da sílaba seguinte – por exemplo, em "relógio", refere-se à medida de F1 da vogal /ɔ/. CONT.PREC e CONT.SEG codificam o segmento fonológico que precede ou antecede a vogal pretônica.

É importante reforçar que, antes de chegar ao ponto de análises multivariadas, idealmente o pesquisador já terá feito análises preliminares, por meio de tabelas e gráficos, e já terá uma boa ideia de como se dá a distribuição de seus dados: se há poucas ocorrências em certos fatores, se parece haver diferenças entre fatores de uma variável, se os testes univariados apontam para diferenças significativas ou não. Por outro lado, se o pesquisador está trabalhando com mais de uma VI/variável previsora, é *imprescindível* realizar análises multivariadas, pois o comportamento de certas variáveis pode não ser tão preponderante ou pode mudar em face de outras.

A função empregada para criar modelos multivariados para uma variável dependente numérica é `lm()`, que já vimos na Lição 11. Aqui vamos aplicá-la à variável resposta numérica altura da vogal /e/, em Hz: F1.NORM. Nesta lição, vamos criar modelos com apenas uma variável previsora a fim de treinar a leitura dos resultados. Contudo, o interesse principal nessa função é a modelagem multivariada – que faremos na próxima.

Em outras palavras, as tarefas desta lição têm objetivo puramente didático; as análises que você fará efetivamente com seus dados e que acabará reportando serão mais parecidas com o que faremos na Lição 13.

Crie um primeiro modelo para testar se há correlação entre a altura da vogal e a AMOSTRA, ou seja, para testar se a altura da vogal /e/ na fala de paraibanos e de paulistanos difere. O primeiro argumento de `lm()` é uma fórmula y ~ x, e o segundo argumento é o conjunto de dados. Digite então a linha de comando a seguir para criar o modelo `mod1`.

```
mod1 <- lm(F1.NORM ~ AMOSTRA, data = VOGAL_e)
```

Visualize agora o resumo do resultado de `mod1` com `summary()`.

```
summary(mod1)

##
## Call:
## lm(formula = F1.NORM ~ AMOSTRA, data = VOGAL_e)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -75.71 -18.31  -2.20  16.12 145.56
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    431.756      1.480 291.767  < 2e-16 ***
## AMOSTRASP2010   -8.790      2.084  -4.219 2.79e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.29 on 684 degrees of freedom
## Multiple R-squared:  0.02536,    Adjusted R-squared:  0.02394
## F-statistic:  17.8 on 1 and 684 DF,  p-value: 2.788e-05
```

Vamos examinar o resultado do modelo linear. O primeiro ponto a se checar é se os resíduos têm distribuição normal: valor de mediana próximo a zero e valores de mínimo-máximo e 1Q-3Q razoavelmente simétricos. Olhando os resíduos de `mod1`, qual é a sua avaliação?

- a distribuição dos resíduos segue a distribuição normal
- a distribuição dos resíduos não segue a distribuição normal

O valor máximo de resíduo, 145,56, difere muito do valor mínimo, -75,71 em valores absolutos. Isso é indicativo de que há valores atípicos na distribuição. Lembre-se que os resíduos são a diferença entre os valores observados e os valores previstos pelo modelo; um resíduo tão grande, de 145 Hz, refere-se a alguma observação muito acima do esperado.

Visualize a distribuição das medidas de F1.NORM por AMOSTRA nos dados `VOGAL_e`, por meio de um boxplot (Figura 12.1).

```
ggplot(VOGAL_e, aes(x = AMOSTRA, y = F1.NORM)) +
  geom_boxplot(notch = TRUE) +
  scale_y_reverse()
```

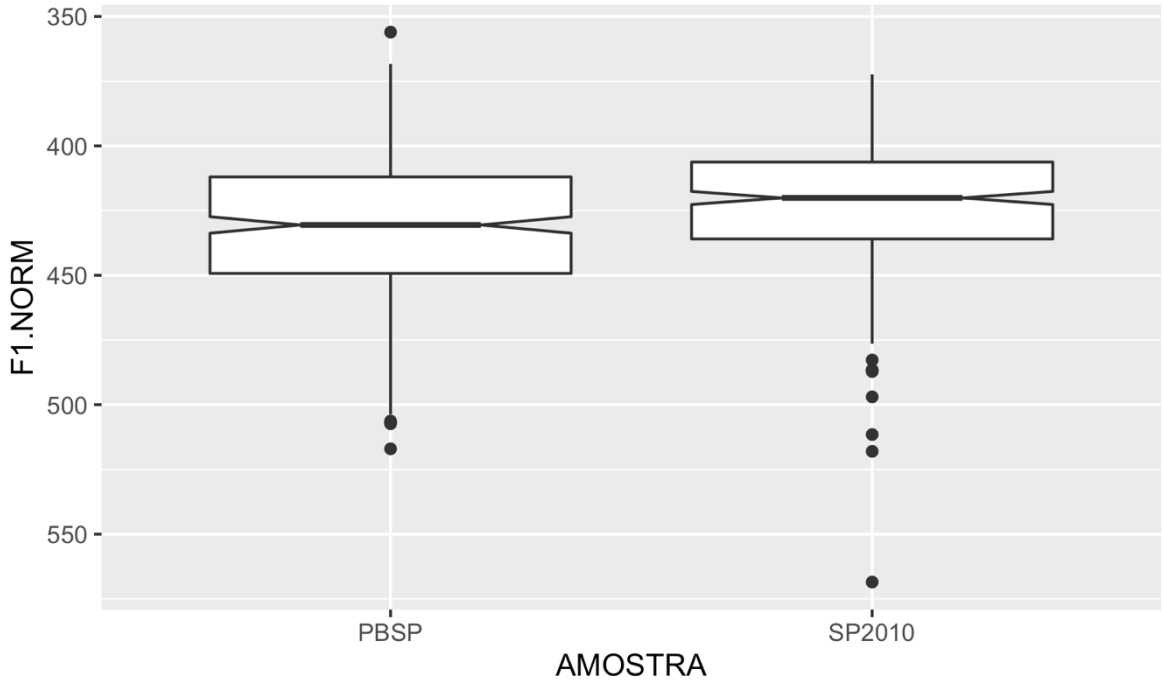

Figura 12.1: Boxplots das medidas de F1 normalizado da vogal /e/ nas amostras PBSP e SP2010. Fonte: própria.

Vemos que as medidas de F1.NORM se concentram entre 400 e 450 Hz, e que há alguns poucos valores atípicos acima de 500 Hz. Crie então um novo subconjunto de dados, chamado `VOGAL_e2`, que inclui os dados de `VOGAL_e` cuja medida de F1.NORM é abaixo de 500 Hz.

```
VOGAL_e2 <- filter(VOGAL_e, F1.NORM < 500)
```

Faça um novo modelo linear, chamado `mod2`, com a mesma fórmula de `mod1` e os dados de `VOGAL_e2`.

```
mod2 <- lm(F1.NORM ~ AMOSTRA, data = VOGAL_e2)
```

Visualize o resumo do resultado de `mod2`.

```
summary(mod2)

##
## Call:
## lm(formula = F1.NORM ~ AMOSTRA, data = VOGAL_e2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -74.357 -17.225  -1.261  16.848  74.960
##
## Coefficients:
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     430.406      1.391 309.423  < 2e-16 ***
## AMOSTRASP2010   -8.400      1.954  -4.298 1.97e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.42 on 675 degrees of freedom
## Multiple R-squared:  0.02664,    Adjusted R-squared:  0.0252
## F-statistic: 18.48 on 1 and 675 DF,  p-value: 1.975e-05
```

Comparativamente a `mod1`, os resíduos de `mod2` são mais simétricos: os valores absolutos de mínimo e máximo são em torno de 74; os valores absolutos de 1Q-3Q são próximos de 17, e a mediana está razoavelmente próxima de zero.

Na sequência do resultado, temos a tabela de coeficientes, com as estimativas, erro padrão, valor-*t* e significância. Assim como no modelo linear que vimos na Lição 11, a significância aqui testa a hipótese nula de que a estimativa é igual a zero. Pela tabela, ambas as estimativas diferem significativamente de zero. Como interpretar esses números?

Quando testamos a correlação entre duas variáveis numéricas (Lição 11), o coeficiente linear mediu a estimativa de y (a VD/variável resposta) quando x (a VI/variável previsora) = 0. Aqui, x se refere à variável AMOSTRA. O que significa AMOSTRA = 0? Trata-se do valor de referência da variável, que é seu primeiro nível (na ordem da tabela ou modificado pelo usuário). Nesse df, PBSP é o primeiro nível e, portanto, a estimativa de 430,4 Hz representa o valor médio esperado de F1.NORM para a vogal /e/ dos paraibanos em São Paulo. O valor-*p* abaixo de 0,05, para nós, nada significa aqui; a medida apenas indica que é baixa a probabilidade de se ter observado 430 caso o verdadeiro parâmetro seja zero, mas nunca esperaríamos que uma vogal /e/ tivesse 0 Hz de F1!

A estimativa para SP2010, -8,4 Hz, deve ser lida em relação ao coeficiente linear. O modelo estima que o valor de F1.NORM para paulistanos é 430,4 - 8,4 = 422 Hz – ou seja, em média, os valores de F1.NORM para paulistanos são menores, o que indica que a vogal pretônica /e/ de paulistanos tende a ser significativamente mais alta. Os resultados

de `mod2` podem ser colocados na forma da função F1.NORM = 430,4 - 8,4 * AMOSTRASP2010.

Se você se lembra do teste-t da Lição 10, vimos valores semelhantes para as médias de F1.NORM para cada amostra (430 Hz para PBSP e 422 Hz para SP2010). Há uma pequena diferença, que se deve ao fato de que aqui excluímos alguns valores atípicos para `mod2`. Faça esse teste: aplique um teste-t para testar a hipótese nula de que a diferença entre as médias de F1.NORM entre paraibanos e paulistanos é zero, no subconjunto de dados `VOGAL_e2`.

```
t.test(F1.NORM ~ AMOSTRA, data = VOGAL_e2)
```

```
##
##  Welch Two Sample t-test
##
## data:  F1.NORM by AMOSTRA
## t = 4.2863, df = 639.61, p-value = 2.097e-05
## alternative hypothesis: true difference in means between group PBSP
and group SP2010 is not equal to 0
## 95 percent confidence interval:
##   4.551609 12.247925
## sample estimates:
##   mean in group PBSP mean in group SP2010
##             430.4061             422.0064
```

As médias calculadas no teste-t correspondem exatamente às estimativas do modelo linear. Na prática, ao incluir apenas uma VI na função `lm()`, estamos efetivamente executando uma análise univariada como o teste-t. A diferença do modelo linear, nesse caso, é visualizar o resultado da estimativa em termos da *diferença* entre os níveis. Não subestime esse fato, pois isso é uma *grande* vantagem: isso torna mais fácil verificar se essa diferença é ou não zero.

As demais informações são interpretadas como já comentado na lição anterior: o erro padrão dos resíduos indica o quanto da variação o modelo não é capaz de explicar e R^2/R^2-ajustado indicam o quanto da variação nos dados é explicada pelas variáveis incluídas no modelo. A estatística-F permite avaliar a significância do modelo como um todo e comparar diferentes modelos.

O pacote effects permite plotar gráficos de efeitos a partir de modelos lineares e logísticos criados no R. Aplique a função `plot()` com os seguintes argumentos: (i) para

os dados a plotar, digite `effect("AMOSTRA", mod2)`; (ii) para os limites do eixo y, coloque ylim = c(450, 400) – para que os valores do eixo fiquem invertidos; e (iii) inclua o argumento grid = T (Figura 12.2).

```
plot(effect("AMOSTRA", mod2), ylim = c(450, 400), grid = T)
```

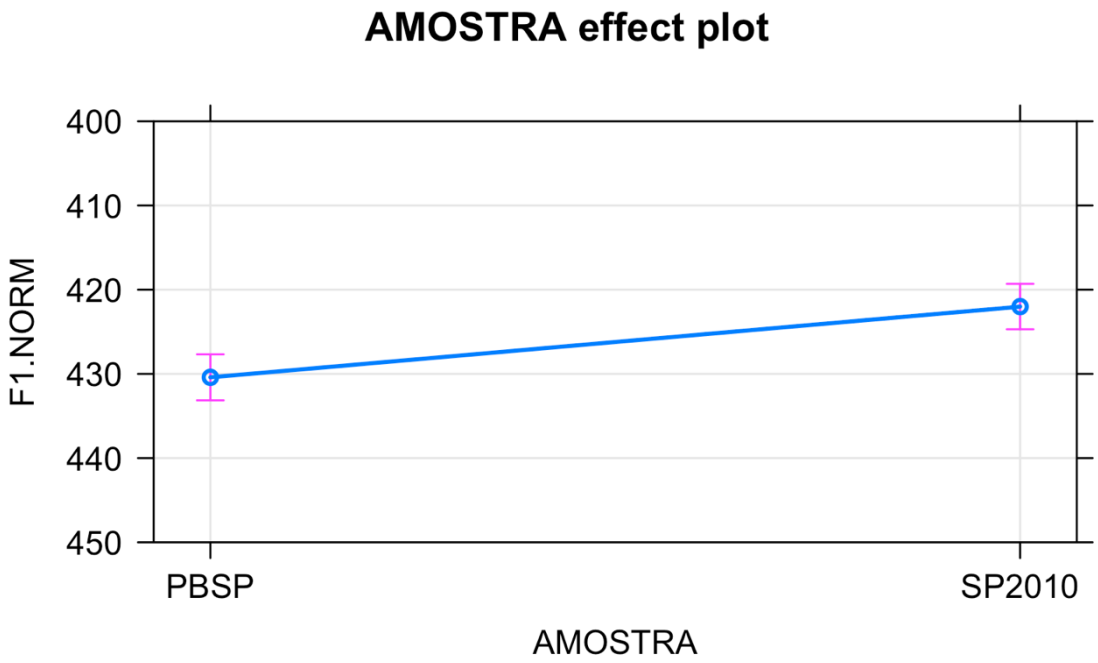

Figura 12.2: Gráfico de efeitos da variável Amostra para a altura da vogal /e/ pretônica. Fonte: própria.

O gráfico de efeitos plota as estimativas de cada um dos níveis da variável indicada dentro da função `effect()`, extraídas do modelo, junto com o intervalo de confiança indicado pelas barras. Vê-se que a diferença entre os níveis da variável AMOSTRA é significativa porque os intervalos não se sobrepõem: mesmo que a medida de F1.NORM para PBSP fosse mais alta, e a medida de F1.NORM para SP2010 fosse mais baixa, elas ainda assim não seriam iguais dentro do intervalo de confiança de 95% – o que significa que é menor do que 5% a probabilidade de que as médias sejam iguais, que não há diferença entre as amostras.

Aqui usamos a função `plot()`, da instalação base do R, para plotar esse gráfico. Também é possível fazer isso com o ggplot2, mas aí são necessários outros passos. Para sua curiosidade, deixei as linhas de comando prontas ao final do *script* dessa lição. Rode-

as posteriormente para ver como ficaria um gráfico no ggplot2 (como sempre, mais bonito do que com a função da instalação base!).

Vejamos agora um modelo que inclui uma variável previsora com mais de dois níveis. Faça um modelo linear, chamado `mod3`, com a variável resposta F1.NORM e a variável previsora CONT.SEG, nos dados `VOGAL_e2`.

```
mod3 <- lm(F1.NORM ~ CONT.SEG, data = VOGAL_e2)
```

Veja o resultado de `mod3` com a função `summary()`.

```
summary(mod3)

##
## Call:
## lm(formula = F1.NORM ~ CONT.SEG, data = VOGAL_e2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -72.167 -17.788  -1.421  15.539  73.615
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 433.429      3.007 144.162  < 2e-16 ***
## CONT.SEGlabial              -11.106      3.943  -2.817 0.004997 **
## CONT.SEGpalatal.sibilante   -14.299      4.014  -3.562 0.000394 ***
## CONT.SEGvelar                -7.108      3.546  -2.004 0.045425 *
## CONT.SEGvibrante             -5.213      3.447  -1.512 0.130906
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.51 on 672 degrees of freedom
## Multiple R-squared:  0.02409,    Adjusted R-squared:  0.01828
## F-statistic: 4.147 on 4 and 672 DF,  p-value: 0.002515
```

Vamos direto aos coeficientes desta vez. Para bem entender o que significam as estimativas, é necessário saber quais são os níveis da variável CONT.SEG. Digite `levels(VOGAL_e2$CONT.SEG)`.

```
levels(VOGAL_e2$CONT.SEG)

## [1] “dental.alveolar”   “labial”           “palatal.sibilante”
## [4] “velar”             “vibrante”
```

A partir dos níveis de CONT.SEG, a que se refere a estimativa do Intercept?

- dental.alveolar
- labial
- vibrante

- velar
- palatal.sibilante

O valor de 433,4 Hz, portanto, é a estimativa da medida de F1.NORM quando a consoante seguinte é uma dental-alveolar (p.ex. pe*d*agogia). Os demais valores devem ser lidos em relação a esse nível, somando-se o valor das respectivas estimativas ao coeficiente linear (Intercept). Assim, a estimativa da medida de F1.NORM para quando a consoante seguinte é labial (p.ex., de*m*ora) é 433,4 - 11,1 = 422,3 Hz; para as consoantes palatais ou sibilantes (p.ex. pe*s*ado) é 433,4 - 14,3 = 419,1 Hz etc. (lembre-se que a soma de um valor negativo equivale à subtração!). Em outras palavras, a estimativa de cada termo previsor é computada pela função F1.NORM = 433,429 + (-11,106 * CONT.SEGlabial) + (-14,299 * CONT.SEGpalatal.sibilante) + (-7,108 * CONT.SEGvelar) + (-5,213 * CONT.SEGvibrante).

Em relação a consoantes dental-alveolares, quais níveis têm medidas de F1.NORM significativamente diferentes?

- segmentos palatais-sibilantes
- segmentos palatais e vibrantes
- segmentos labiais, palatais-sibilantes e velares

Em relação a consoantes dental-alveolares, quais níveis não têm medidas de F1.NORM significativamente diferentes?

- segmentos vibrantes
- segmentos palatais-sibilantes
- segmentos labiais, palatais-sibilantes e velares

Façamos agora um gráfico de efeitos para visualizar as diferenças entre as consoantes. A partir da linha de comando com `plot()` e `effect()` digitada acima, substitua o nome da variável para CONT.SEG e do modelo para `mod3` (Figura 12.3).

```
plot(effect("CONT.SEG", mod3), ylim = c(450, 400), grid = T)
```

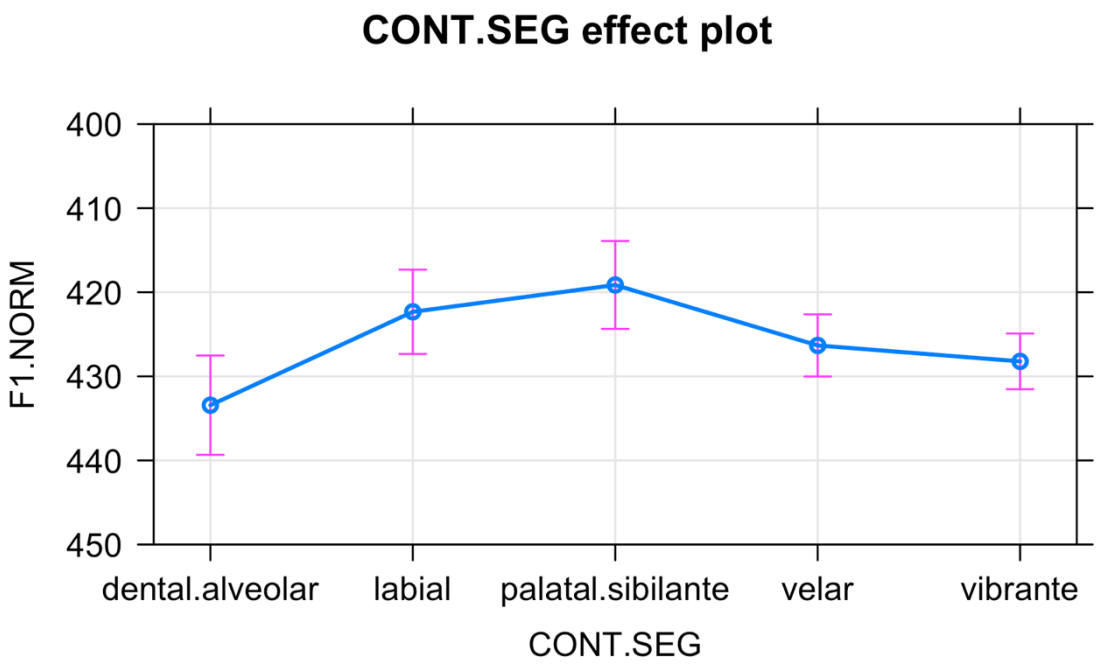

Figura 12.3: Gráfico de efeitos da variável Contexto Fônico Seguinte para a altura da vogal /e/ pretônica. Fonte: própria.

No gráfico de efeitos, vemos que as labiais, palatais-sibilantes e velares se distanciam mais da média de F1.NORM em relação às dental-alveolares. Como estas são o valor de referência de CONT.SEG, as estimativas de todos os níveis são calculados a partir desse nível. Mas e se quiséssemos saber se há diferenças significativas na medida de F1.NORM entre, por exemplo, vibrantes e palatais-sibilantes? ou entre labiais e velares? Seria necessário mudar o nível de referência a cada novo teste?

Para múltiplas comparações, pode-se usar o método de Tukey, por meio da função `TukeyHSD()`. Digite `TukeyHSD(aov(mod3))` para verificar o teste de significância de todos os pares possíveis dos níveis de CONT.SEG.

```
TukeyHSD(aov(mod3))

##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = mod3)
##
## $CONT.SEG
##                                          diff         lwr        upr
## labial-dental.alveolar            -11.105787 -21.8908147 -0.3207598
## palatal.sibilante-dental.alveolar -14.299162 -25.2787251 -3.3195998
```

```
## velar-dental.alveolar                -7.108380 -16.8082987  2.5915390
## vibrante-dental.alveolar             -5.213098 -14.6411626  4.2149660
## palatal.sibilante-labial             -3.193375 -13.2738266  6.8870762
## velar-labial                          3.997407  -4.6716831 12.6664979
## vibrante-labial                       5.892689  -2.4711082 14.2564860
## velar-palatal.sibilante               7.190783  -1.7191618 16.1007270
## vibrante-palatal.sibilante            9.086064   0.4728719 17.6992564
## vibrante-velar                        1.895282  -5.0130258  8.8035889
##                                         p adj
## labial-dental.alveolar              0.0399172
## palatal.sibilante-dental.alveolar 0.0036031
## velar-dental.alveolar               0.2649058
## vibrante-dental.alveolar            0.5547385
## palatal.sibilante-labial            0.9090578
## velar-labial                        0.7151076
## vibrante-labial                     0.3038915
## velar-palatal.sibilante             0.1781207
## vibrante-palatal.sibilante          0.0327983
## vibrante-velar                      0.9443750
```

O resultado do teste de Tukey é uma tabela com as estimativas de diferença entre cada par, o intervalo de confiança (os limites lwr e upr) e o valor-*p* ajustado (já que são múltiplas comparações). Veja que só são significativas as diferenças cujo intervalo de confiança não inclui zero. A partir dessas comparações, o pesquisador pode decidir juntar novos níveis em um mesmo fator, contanto que também haja justificativa teórica para tal.

Vamos agora fazer um último modelo univariado com uma variável previsora numérica. Crie um modelo chamado `mod4` que testa se há correlação entre F1.NORM e F1.SEG.NORM, a altura da vogal da sílaba seguinte, nos dados de `VOGAL_e2`.

```
mod4 <- lm(F1.NORM ~ F1.SEG.NORM, data = VOGAL_e2)
```

Veja o resultado de `mod4` com `summary()`.

```
summary(mod4)
```

```
##
## Call:
## lm(formula = F1.NORM ~ F1.SEG.NORM, data = VOGAL_e2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.456 -18.508  -0.944  15.235  74.681
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 379.50710    9.52279  39.852  < 2e-16 ***
## F1.SEG.NORM   0.12278    0.02494   4.924 1.07e-06 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.32 on 675 degrees of freedom
## Multiple R-squared:  0.03467,    Adjusted R-squared:  0.03324
## F-statistic: 24.24 on 1 and 675 DF,  p-value: 1.068e-06
```

Vamos novamente direto aos coeficientes, para treinar sua leitura. O coeficiente linear deste modelo é 379,5 Hz e o coeficiente angular é 0,12 Hz, e ambos diferem significativamente de zero. Novamente, o fato de o primeiro coeficiente diferir de zero não nos diz nada, pois não esperaríamos que a vogal /e/ tivesse 0 Hz, mas o segundo coeficiente nos traz uma informação relevante: existe uma correlação entre a altura da vogal seguinte e a altura da vogal /e/ pretônica. O coeficiente angular é positivo, o que indica que quanto maior o valor de F1.SEG.NORM, maior o valor de F1.NORM; em termos numéricos, a leitura desse resultado é que a cada unidade de F1.SEG.NORM (400 Hz, 401 Hz, 402 Hz...), estima-se que o valor de F1.NORM aumente em 0,12 Hz. Ou, ainda, a estimativa da variável resposta segue a função F1.NORM = 379,50710 + (0,12278 * F1.SEG.NORM). Digamos que uma vogal da sílaba seguinte à pretônica tenha F1.SEG.NORM de 400 Hz; a estimativa do valor de F1.NORM é 379,50710 + (0,12278 * 400) = 428,6191 Hz.

O gráfico de efeitos permite visualizar essa correlação de forma mais clara. A partir da linha de comando em que usamos a função `plot()` e `effect()`, substitua o nome da variável para F1.SEG.NORM e o modelo para `mod4` (Figura 12.4).

```
plot(effect("F1.SEG.NORM", mod4), ylim = c(450, 400), grid = T)
```

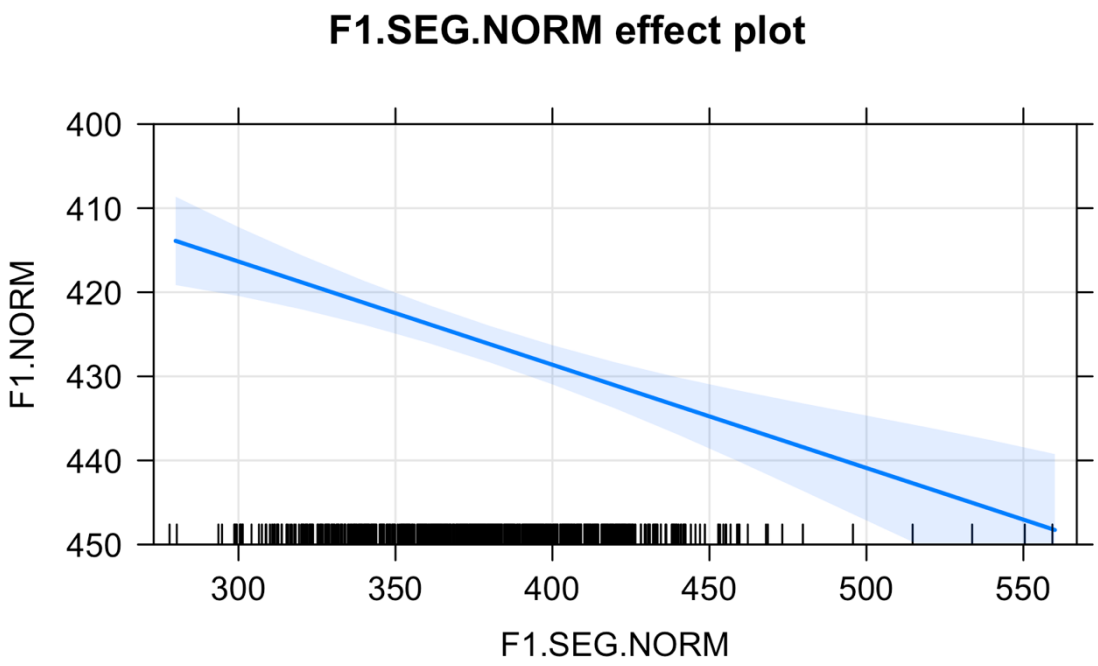

Figura 12.4: Gráfico de efeitos da variável F1 da vogal da sílaba seguinte para a altura da vogal /e/ pretônica. Fonte: própria.

Como F1.SEG.NORM é uma variável numérica, o tipo de gráfico mudou para uma linha de regressão. A figura mostra uma curva descendente porque invertemos o eixo y para adequá-lo à convenção de representar valores mais altos de F1 na parte de baixo. Aí vemos claramente que quanto maior o valor de F1.SEG.NORM, mais baixa tende a ser a vogal /e/ pretônica. (Eis aí o fenômeno de harmonia vocálica!) A mancha em volta da linha de regressão indica o intervalo de confiança das estimativas. Os pequenos traços verticais ao longo do eixo x mostram onde estão e onde se concentram as observações: há muito mais dados de F1.SEG.NORM entre cerca de 330 Hz e 420 Hz. Veja que o intervalo de confiança das estimativas se "alarga" nas partes em que há menor número de observações, justamente porque é mais difícil chegar a estimativas precisas quando não temos muitos dados.

Vimos então como ler os resultados de um modelo linear com uma variável previsora binária (`mod1` e `mod2`), uma variável previsora com mais de 2 fatores (`mod3`) e uma variável previsora numérica (`mod4`). Podemos agora partir para modelos um pouco mais complexos. Façamos então um modelo `mod5` com a inclusão de duas variáveis

previsoras, AMOSTRA + SEXO. Digite `mod5 <- lm(F1.NORM ~ AMOSTRA + SEXO, data = VOGAL_e2)`.

```
mod5 <- lm(F1.NORM ~ AMOSTRA + SEXO, data = VOGAL_e2)
```

E veja o resultado com `summary()`.

```
summary(mod5)
```

```
##
## Call:
## lm(formula = F1.NORM ~ AMOSTRA + SEXO, data = VOGAL_e2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -73.763 -16.827  -1.507  16.510  74.168
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    431.190      1.789 241.086  < 2e-16 ***
## AMOSTRASP2010   -8.392      1.955  -4.293 2.02e-05 ***
## SEXOmasculino   -1.378      1.975  -0.698    0.486
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.43 on 674 degrees of freedom
## Multiple R-squared:  0.02734,    Adjusted R-squared:  0.02446
## F-statistic: 9.474 on 2 and 674 DF,  p-value: 8.756e-05
```

O modelo agora fornece o valor do coeficiente linear, 431,190 Hz, e dois coeficientes angulares, um para AMOSTRASP2010 e outro para SEXOmasculino. Como o modelo inclui duas variáveis previsoras, o valor do coeficiente linear deve ser interpretado em relação a ambos os níveis de referência: AMOSTRA PBSP e SEXO feminino. O valor de 431,190 Hz, portanto, é a estimativa da altura da vogal /e/ pretônica na fala de *paraibanas*.

Se somarmos a estimativa de AMOSTRASP2010 ao coeficiente linear (431,190 - 8,392 = 422,798 Hz), teremos o valor estimado de F1.NORM para falantes da amostra SP2010 do sexo *feminino* – isso porque, em relação ao nível de referência, alteramos apenas AMOSTRA, não SEXO. Se somarmos a estimativa de SEXOmasculino ao coeficiente linear (431,190 - 1,378 = 429,82 Hz), teremos o valor estimado de F1.NORM para falantes do sexo masculino da AMOSTRA PBSP – isso porque, em relação ao nível de referência, neste caso alteramos apenas SEXO, não AMOSTRA. Para obter a estimativa de F1.NORM para

falantes do sexo masculino da amostra SP2010, é necessário somar ambos os coeficientes angulares ao coeficiente linear: 431,190 - 8,392 - 1,378 = 421,42 Hz (Tabela 12.1).

Tabela 12.1: Cálculo das probabilidades em logodds a partir das estimativas geradas pelo modelo de regressão linear.

	F	M
PBSP	431,190	429,812
SP2010	422,798	421,420

Fonte: própria.

O modelo nos informa que existe uma diferença significativa na altura da vogal /e/ pretônica na fala de paraibanos e paulistanos, mas não há diferença significativa entre os sexos. Isso pode ser mais bem visualizado por meio de um gráfico de efeitos! A partir da última linha de comando em que você empregou a função `plot()`, modifique o primeiro argumento – onde estava effect("F1.SEG.NORM", mod4) – para allEffects(mod5). Em modelos multivariados, em vez de `effect()`, usamos a função `allEffects()` com o modelo como único argumento. Certifique-se de que sua linha de comando é `plot(allEffects(mod5), ylim = c(450, 400), grid = T)` (Figura 12.5).

```
plot(allEffects(mod5), ylim = c(450, 400), grid = T)
```

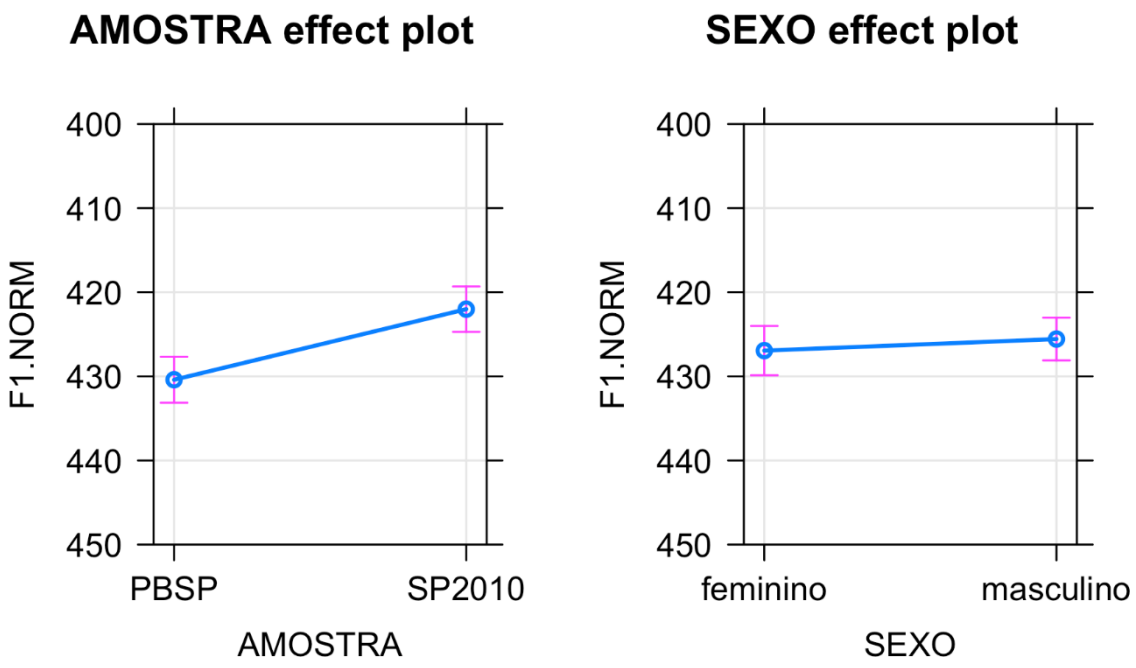

Figura 12.5: Gráfico de efeitos das variáveis Amostra e Sexo para a altura da vogal /e/ pretônica. Fonte: própria.

A Figura 12.5 mostra dois gráficos de efeitos, um para cada variável. Aí vemos que a diferença na altura da vogal /e/ pretônica é significativa para AMOSTRA (pois os intervalos de confiança não se sobrepõem), mas não é para a variável SEXO.

No modelo acima, incluímos duas variáveis previsoras por meio do operador de soma +. Essa notação não é por acaso: existe aí a pressuposição de que o efeito dos previsores é *aditivo*, de modo que, para chegar aos valores estimados de cada uma das combinações possíveis (mulheres paraibanas, homens paraibanos, mulheres paulistanas e homens paulistanos), somamos os coeficientes respectivos. Isso pressupõe que o efeito de cada uma das variáveis é independente. Contudo, isso nem sempre é o caso nos dados.

Vejamos um exemplo de Gries (2019, p.223). Imagine um estudo que compara a extensão de sujeitos e objetos em orações principais e em orações subordinadas, por meio do número de sílabas. O pesquisador coletou um *corpus*, separou sentenças com verbos transitivos diretos em que havia um SN sujeito e um SN objeto, contou o número de sílabas para cada SN, e codificou cada SN para as variáveis função sintática e tipo de oração.

A Figura 12.6 representa um resultado hipotético de um modelo linear que incluiu extensão em sílabas como variável resposta e função sintática e tipo de oração como variáveis previsoras. Os pontos representam as médias previstas do número de sílabas do SN para cada uma das condições.

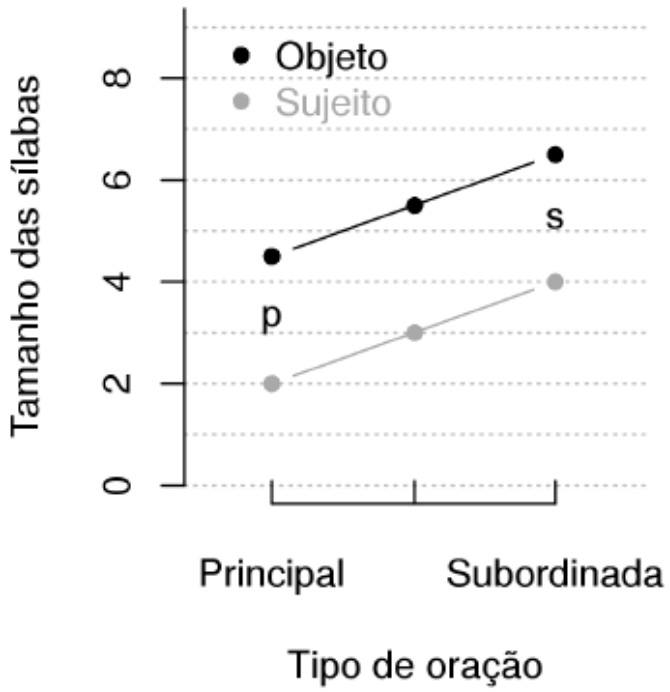

Figura 12.6: Exemplo de não interação entre VIs. Fonte: Gries (2019, p. 223).

A partir da figura, qual é a extensão prevista de sujeitos em orações principais?

- 2 sílabas
- 4 sílabas
- 4,5 sílabas
- 6,5 sílabas

A partir da figura, qual é a extensão prevista de objetos em orações subordinadas?

- 2 sílabas
- 4 sílabas
- 4,5 sílabas
- 6,5 sílabas

Nesse modelo, a extensão de SNs é em média maior em orações subordinadas do que em orações coordenadas, e em objetos em relação a sujeitos. A relação entre ambas as condições é constante: em média, as orações subordinadas têm SNs com 2 sílabas a mais do que em orações principais (tanto para sujeitos quanto para objetos), e objetos têm em média 2,5 sílabas a mais do que sujeitos (tanto em orações principais quanto em orações subordinadas). Os efeitos de ambas as variáveis são aditivos, de modo que se espera o menor número de sílabas em SNs sujeito em orações principais, e o maior número de sílabas em SNs objeto em orações subordinadas. A independência entre variáveis pode ser visualizada pelas linhas paralelas no gráfico.

A Figura 12.7 representa outro resultado hipotético a partir dos dados. O gráfico à esquerda mostra as estimativas de extensão média do SN em cada uma das quatro condições, e a figura à direita mostra, por meio da linha pontilhada, o que se esperaria caso a relação entre as variáveis fosse de independência.

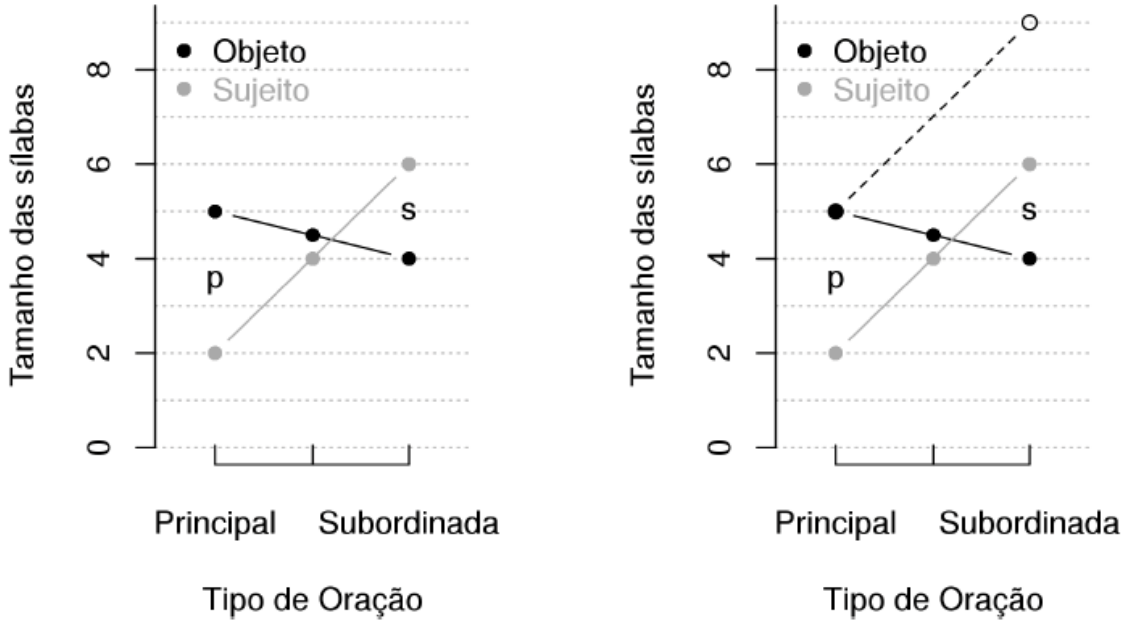

Figura 12.7: Exemplo de interação entre VIs (1). Fonte: Gries (2019, p. 223).

A partir da figura à esquerda, qual é a estimativa de extensão do SN sujeito em orações principais nesse modelo?

- 2 sílabas

- 4 sílabas
- 5 sílabas
- 6 sílabas

A partir da figura à esquerda, qual é a estimativa de extensão do SN objeto em orações subordinadas nesse modelo?

- 2 sílabas
- 4 sílabas
- 5 sílabas
- 6 sílabas

A figura à esquerda ilustra uma interação entre as variáveis função sintática e tipo de oração. A relação entre elas não é constante: enquanto para SNs sujeitos a diferença de extensão é de 4 sílabas entre orações principais e subordinadas, a diferença para SNs objetos não só é menor (1 sílaba), mas também vai na direção oposta (SNs maiores em orações principais do que em orações subordinadas). A interação é claramente visualizada pelo fato de que as linhas não são paralelas, mas se cruzam. Um modelo que previsse o efeito aditivo entre as variáveis previsoras – i.e. tamanho das sílabas ~ função sintática + tipo de oração – poderia chegar ao resultado do gráfico à direita, em que a estimativa da extensão de SNs objeto em orações subordinadas está bastante equivocada.

Veja agora a Figura 12.8, que ilustra outro caso hipotético de interação. De modo semelhante ao exemplo anterior, o gráfico à esquerda mostra as verdadeiras médias de extensão dos SNs e o gráfico à direita ilustra o resultado a que se chegaria caso não se previsse uma interação entre as variáveis previsoras.

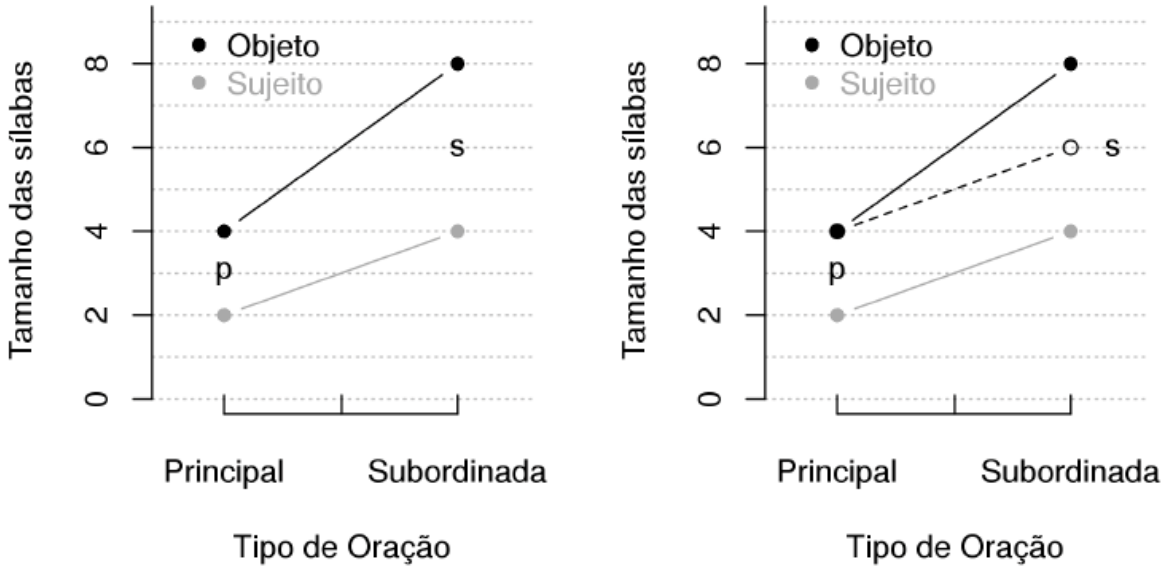

Figura 12.8: Exemplo de interação entre VIs (2). Fonte: Gries (2019, p. 225).

A partir da figura à esquerda, qual é a estimativa de extensão de SN sujeito em orações principais neste modelo?

- 2 sílabas
- 4 sílabas
- 6 sílabas
- 8 sílabas

A partir da figura à esquerda, qual é a estimativa de extensão de SN objeto em orações subordinadas neste modelo?

- 2 sílabas
- 4 sílabas
- 6 sílabas
- 8 sílabas

A partir da figura à direita, qual é a estimativa de extensão de SN objeto em orações subordinadas caso não se preveja a interação?

- 2 sílabas
- 4 sílabas

- 6 sílabas
- 8 sílabas

Aqui novamente se tem uma interação. Embora as linhas não se cruzem no gráfico, elas também não são paralelas. Isso significa que um modelo que não prevê a interação entre as variáveis função sintática e tipo de oração não seria capaz de estimar corretamente qual é a extensão dos SNs em cada condição. Para prever corretamente a extensão de SNs objeto em orações subordinadas, seria necessário ter mais um coeficiente que corrigisse a estimativa – neste caso, um coeficiente que informasse que ainda é necessário somar +2 para prever a extensão na condição objeto-oração subordinada.

Outro modo de entender o que é a interação entre duas variáveis previsoras é que o efeito de cada uma delas não pode ser determinado isoladamente, independentemente do efeito da outra. É necessário levar em consideração o efeito conjunto de ambas para bem prever a estimativa. Daí a importância de realizar análises multivariadas: seria impossível prever interações testando uma única variável a cada teste. Daí também deriva a inadequação de se usar o termo "variável independente" – em vez de "variável previsora" – quando se realiza uma análise multivariada, pois o efeito das variáveis não necessariamente é independente.

O ponto aqui é mostrar a importância de testar interações nos modelos lineares a fim de chegar a estimativas mais precisas para a variável resposta.

Você deve estar se perguntando como determinar se há interação entre duas variáveis. O R não tem um jeito automático de informar a você se duas variáveis previsoras são ou não são independentes entre si (nenhum software faz isso). Cabe ao pesquisador prever possíveis casos de interação, tentar visualizá-los em gráficos exploratórios e testá-los nos modelos estatísticos multivariados. A literatura sobre o assunto também é uma boa fonte para identificar casos de possível interação entre variáveis. Na Sociolinguística Variacionista, por exemplo, há muitos casos reportados de interação entre as variáveis classe social e sexo/gênero dos falantes (p.ex. Labov, 1990).

De posse dessa informação, um pesquisador que esteja trabalhando com essas duas variáveis previsoras já deve estar atento a um possível efeito interativo entre elas.

Nos modelos lineares, uma interação é incluída por meio do operador * em vez de +. No modelo `mod5`, vimos que há um efeito de AMOSTRA na estimativa de F1.NORM, mas não há um efeito de SEXO (não há diferença significativa entre homens e mulheres). A partir da linha de comando em que se criou `mod5`, crie um novo modelo `mod6` com a substituição de + por *. Nele, estamos testando não só o efeito das variáveis AMOSTRA e SEXO, como também se há uma interação entre elas.

```
mod6 <- lm(F1.NORM ~ AMOSTRA * SEXO, data = VOGAL_e2)
```

Visualize agora o resultado de `mod6` com `summary()`.

```
summary(mod6)

##
## Call:
## lm(formula = F1.NORM ~ AMOSTRA * SEXO, data = VOGAL_e2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -73.555 -17.028  -1.527  16.718  74.440
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 431.4651     2.1207 203.451  < 2e-16 **
*
## AMOSTRASP2010                -8.9387     2.9889  -2.991  0.00289 **
## SEXOmasculino                -1.8616     2.8118  -0.662  0.50816
## AMOSTRASP2010:SEXOmasculino   0.9562     3.9534   0.242  0.80896
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.45 on 673 degrees of freedom
## Multiple R-squared:  0.02743,    Adjusted R-squared:  0.02309
## F-statistic: 6.327 on 3 and 673 DF,  p-value: 0.0003107
```

Olhando os coeficientes, vemos que existe uma diferença significativa entre as amostras, e que não há diferença entre os sexos (como já visto acima). O modelo agora inclui um novo coeficiente, da interação entre AMOSTRASP2010:SEXOmasculino, que estima o valor de 0,9562 Hz. Isso significa que a estimativa de F1.NORM para falantes paulistanos do sexo masculino é 431,4651 - 8,9387 - 1,8616 + 0,9562 – ou seja, além dos coeficientes para AMOSTRASP2010 e para SEXOmasculino, há um novo coeficiente "de

ajuste". Contudo, neste modelo, o coeficiente da interação, 0,9562, não difere significativamente de zero, de modo que não faz diferença somar ou não esse novo coeficiente. Tal valor já estava previsto no intervalo de confiança do modelo sem interação. Pode-se concluir que as variáveis AMOSTRA e SEXO são independentes (e que SEXO não se correlaciona com a altura da vogal /e/ pretônica).

Podemos visualizar essa falta de interação por meio das funções `plot()` e `allEffects()`. A partir da linha de comando em que você usou essas duas funções, substitua o nome do modelo para `mod6`.

```
plot(allEffects(mod6), ylim = c(450, 400), grid = T)
```

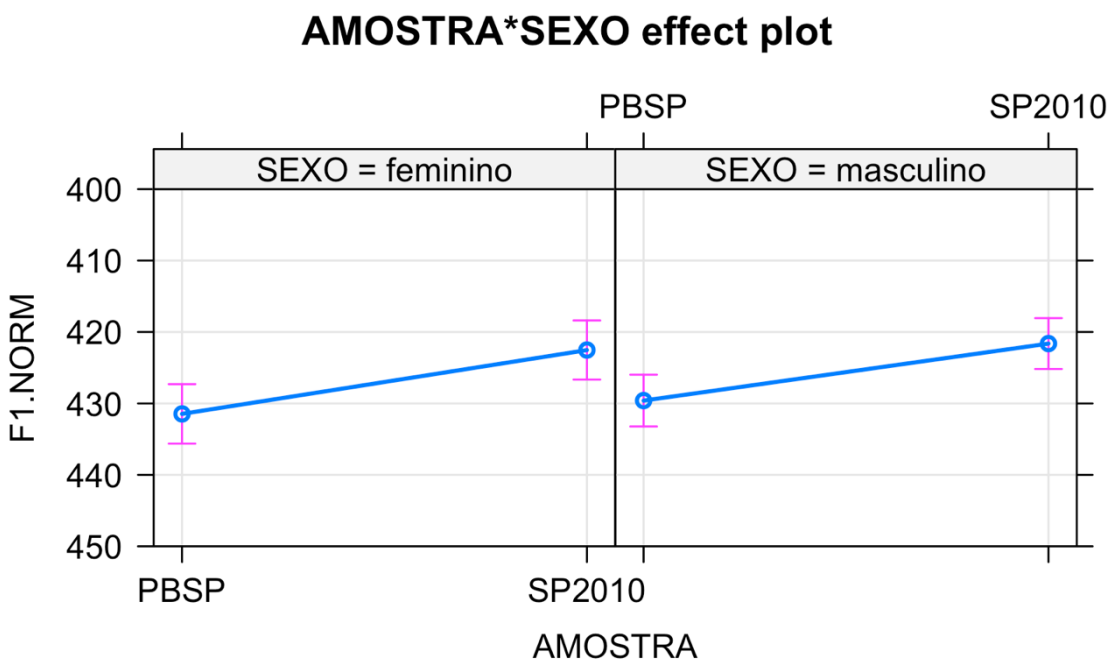

Figura 12.9: Gráfico de efeitos da interação entre as variáveis Amostra e Sexo para a altura da vogal /e/ pretônica. Fonte: própria.

Para o modelo `mod5`, em que havíamos incluído AMOSTRA + SEXO, foram plotados dois gráficos de efeitos, um para cada variável. Para o modelo `mod6`, em que incluímos AMOSTRA * SEXO, os gráficos incluem ambas as variáveis em cada painel. Essa figura é semelhante àquelas dos exemplos de Gries (basta imaginar ambos os painéis no mesmo plano). Aí vemos retas paralelas, que indicam independência entre as variáveis; a

diferença entre as amostras PBSP e SP2010 é significativa e constante tanto para falantes do sexo feminino quanto do sexo masculino.

Isso conclui a primeira parte sobre modelos de regressão linear, mas não conclui os passos da análise! Na próxima lição, veremos como avaliar a inclusão ou não de novas variáveis no modelo e como checar se nosso modelo não viola os pressupostos de um modelo linear.

Para saber mais

Recomendo fortemente a leitura dos capítulos 7 e 8 de Levshina (2015) para se aprofundar nos preceitos da análise de regressão linear. Esses capítulos apresentam os passos e os pressupostos de modelos lineares de modo bastante detalhado.

Exercícios

Nesta lista de exercícios, você vai desenvolver uma análise semelhante à que fizemos na Lição 12, mas agora sobre a vogal /o/ pretônica. Primeiro, carregue os dados da planilha Pretonicas.csv e crie um subconjunto de dados da vogal /o/ pretônica.

Nos dados de /e/ pretônica, a variável CONT.SEG foi recodificada de modo a juntar os segmentos consonantais de acordo com ponto/modo de articulação – ver *script* da Lição 12. Recodifique os dados das variáveis CONT.PREC e CONT.SEG do conjunto de dados da vogal /o/ pretônica com os mesmos critérios empregados para a vogal /e/ pretônica.

1. Há quantos dados de vogal /o/ pretônica na planilha Pretonicas.csv?
2. Há quantos dados de vibrantes no contexto seguinte à vogal pretônica /o/?
3. Há quantos dados de consoantes labiais no contexto precedente à vogal pretônica /o/?
4. Crie um modelo linear para testar se há correlação entre a altura da vogal pretônica (F1.NORM) e a origem do falante (AMOSTRA). Há diferença significativa entre paraibanos e paulistanos quanto à sua realização da vogal pretônica /o/? Justifique sua resposta.

5. Neste modelo, a distribuição dos resíduos segue a distribuição normal? Justifique sua resposta.
6. Faça um boxplot da distribuição de F1.NORM por AMOSTRA. Se existem valores atípicos, parece adequado excluir dados cujas medidas de F1.NORM estão acima de qual ponto?
    a. 600 Hz
    b. 550 Hz
    c. 450 Hz
    d. não há valores atípicos
7. Faça um novo subconjunto de dados incluindo apenas aqueles que estão abaixo do limite estipulado na questão anterior. Quantos dados foram efetivamente excluídos?
8. Refaça a análise para verificar se há correlação entre F1.NORM e AMOSTRA. A nova distribuição de resíduos segue a distribuição normal? Justifique sua resposta.
9. Faça um gráfico de efeitos do modelo acima. Há sobreposição entre os níveis dos intervalos de confiança da variável AMOSTRA? Explique sua resposta.
10. Crie um modelo que testa se há correlação entre a altura da vogal /o/ pretônica (F1.NORM) e o contexto precedente à vogal (CONT.PREC). Entre quais níveis da variável CONT.PREC há diferença significativa?
    a. entre labial e dental.alveolar
    b. entre velar e palatal.sibilante
    c. entre vibrante e labial
    d. entre palatal.sibilante e labial
11. Em quantos Hz a estimativa de F1.NORM para vibrante difere da estimativa do nível de referência?
12. Qual é a medida média de F1.NORM para a vogal /o/ quando é precedida por uma consoante vibrante?
13. O quanto da variação em F1.NORM é explicado pela variável CONT.PREC?

a. 0,8%
b. 1,5%
c. 3,4%
d. 3,9%
e. 27,5%

14. Teste se há interação entre as variáveis AMOSTRA e CONT.PREC. Neste modelo, há interação entre AMOSTRA e CONT.PREC? Justifique sua resposta.
15. Qual é a medida média de F1.NORM para a vogal /o/ quando precedida de consoante velar na fala de paraibanos?
16. Qual é a medida média de F1.NORM para a vogal /o/ quando precedida de consoante labial na fala de paulistanos?
17. Crie um modelo para testar se há correlação entre a altura da vogal /o/ pretônica (F1.NORM) e a altura da vogal da sílaba seguinte (F1.SEG.NORM). A cada unidade de F1.SEG.NORM, em quanto se modifica a estimativa de F1.NORM?
    a. 0,03006
    b. 0,0689
    c. 0,20859
    d. 6,938
18. Calcule a estimativa da medida de F1.NORM quando F1.SEG.NORM tem 450 Hz.
19. Vimos três testes estatísticos que podem ser aplicados a VDs numéricas: (i) teste-t, (ii) teste de correlação e (iii) regressão linear. Às vezes podemos aplicar mais de um deles, às vezes não. Se o pesquisador quer testar se há correlação entre uma VD numérica e uma VI numérica, qual(is) teste(s) pode(m) ser aplicado(s)?
    a. apenas (i)
    b. apenas (iii)
    c. (i) e (ii)
    d. (ii) e (iii)

20. Considere (i) teste-t, (ii) teste de correlação e (iii) regressão linear. Se um pesquisador quer testar se há correlação entre uma VD numérica e uma VI nominal binária, qual(is) teste(s) pode(m) ser aplicado(s)?
    a. apenas (i)
    b. apenas (iii)
    c. (i) e (iii)
    d. (ii) e (iii)
21. Considere (i) teste-t, (ii) teste de correlação e (iii) regressão linear. Se um pesquisador quer testar se há correlação entre uma VD numérica, uma VI nominal com 5 fatores e uma VI numérica, qual(is) teste(s) pode(m) ser aplicado(s)?
    a. apenas (ii)
    b. apenas (iii)
    c. (i) e (iii)
    d. (ii) e (iii)

Lição 13: Regressão Linear Parte 2

N.B.: Rode as linhas de comando a seguir antes de iniciar esta lição. Defina como diretório de trabalho aquele que contém o arquivo Pretonicas.csv.

```
# Definir diretório de trabalho

#setwd()

# Importar planilha de dados

pretonicas <- read_csv("Pretonicas.csv",
                       col_types = cols(.default = col_factor(),
                                        VOGAL = col_factor(levels = c(
"i", "e", "a", "o", "u")),
                                        F1 = col_double(),
                                        F2 = col_double(),
                                        F1.NORM = col_double(),
                                        F2.NORM = col_double(),
                                        F1.SIL.SEG = col_double(),
                                        F2.SIL.SEG = col_double(),
                                        F1.SEG.NORM = col_double(),
                                        F2.SEG.NORM = col_double(),
                                        DIST.TONICA = col_double(),
                                        Begin.Time.s = col_double(),
                                        End.Time.s = col_double(),
                                        Duration.ms = col_double(),
                                        IDADE = col_integer(),
                                        IDADE.CHEGADA = col_integer(),
                                        ANOS.SP = col_integer()
                                        )
                       )

pretonicas$CONT.PREC <-  fct_collapse(pretonicas$CONT.PREC,
                          dental.alveolar = c("t", "d", "n", "l"),
                          labial = c("p", "b", "m", "f", "v"),
                          palatal.sibilante = c("S", "Z", "L", "s", "z
"),
                          velar = c("k", "g"),
                          vibrante = c("h", "R")
                                    )

pretonicas$CONT.PREC <- fct_relevel(pretonicas$CONT.PREC, "dental.alve
olar", "labial", "palatal.sibilante", "velar", "vibrante")

pretonicas$CONT.SEG <- fct_collapse(pretonicas$CONT.SEG,
                     dental.alveolar = c("t", "d", "n", "l"),
                     labial = c("p", "b", "m", "f", "v"),
                     palatal.sibilante = c("S", "Z", "L", "N", "s", "z
```

```
"),
                    velar = c("k", "g"),
                    vibrante = c("r", "h", "R")
                                )

pretonicas$CONT.SEG <- fct_relevel(pretonicas$CONT.SEG, "dental.alveol
ar", "labial", "palatal.sibilante", "velar", "vibrante")

### Criar subconjunto de dados da vogal /e/ pretonica

VOGAL_e <- filter(pretonicas, VOGAL == "e") %>%
  droplevels()

### Retirar valores atípicos

VOGAL_e2 <- filter(VOGAL_e, F1.NORM < 500)
```

Esta lição dá continuidade ao tópico Regressão Linear, iniciado na lição anterior. Ali, criamos modelos relativamente simples de regressão linear com variáveis previsoras binária (AMOSTRA), com mais de 2 fatores (CONT.SEG) e numérica (F1.SEG.NORM), bem como modelos com duas variáveis sem e com interação, com vistas a treinar a leitura dos resultados. Entretanto, análises de regressão linear por meio da função `lm()` permitem incluir mais variáveis previsoras, e um modelo completo deve abarcar tantas variáveis pertinentes quanto possível – são esses modelos completos que, preferencialmente, você reportará em suas publicações.

Por outro lado, ao criar modelos multivariados, é importante ter em mente o princípio da Navalha de Occam, também conhecido como Princípio da Simplicidade ou Princípio da Parcimônia. Ele estabelece que teorias mais simples são preferíveis a teorias mais complexas, e que não se deve agregar hipóteses desnecessárias a uma teoria. Transpondo tal princípio a nossos modelos estatísticos, poderíamos imaginar que um modelo que inclui 20 variáveis previsoras se aproxima mais da realidade; contudo, se 15 dessas variáveis contribuem pouco para fazer previsões a respeito da variável resposta, um modelo com 5 variáveis é preferível por ser mais simples.

É daí também que surge o interesse em modelos multivariados: o efeito de uma variável previsora pode se mostrar pertinente em uma análise univariada, mas se revelar não tão influente quando considerada frente a outras variáveis. A análise multivariada

pode indiciar que o efeito de determinada variável é apenas superficial ou pequeno em vista de outros efeitos. Cabe então a pergunta: como decidir quais e quantas variáveis previsoras incluir num modelo linear?

A decisão sobre quais variáveis têm efeito de fato para descrever, explicar e prever o comportamento da variável resposta cabe inequivocadamente ao pesquisador. Cada variável previsora é uma hipótese a respeito da variável resposta, e as hipóteses, como já vimos, devem ter por base a teoria e a literatura sobre determinado assunto – em outras palavras, devem ser bem motivadas. Mas o pesquisador sempre pode propor novas correlações ou descobrir que um efeito que se mostrou significativo em um conjunto de dados não é em outro conjunto.

Para chegar a um modelo satisfatório dos dados, há duas abordagens possíveis: começar por um modelo estatístico simples e a ele acrescentar novas variáveis previsoras, uma a uma; ou começar com um modelo complexo, com todas as variáveis previsoras, e procurar eliminar aquelas que têm pouca ou nenhuma influência na variável resposta. O R tem algumas funções que facilitam a construção de tais modelos, dentre as quais estão as funções `step()` e `drop1()`.

Nesta lição, vamos precisar de quatro pacotes: tidyverse, car, lme4 e lmerTest. Carregue-os com as linhas de comando a seguir.

```
library(tidyverse)
library(car)
library(lme4)
library(lmerTest)
```

Nesta lição, vamos usar o dataframe `VOGAL_e2`, com que trabalhamos na lição passada. Veja sua estrutura com `str()`.

```
str(VOGAL_e2)

## spec_tbl_df [677 × 27] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ PALAVRA      : Factor w/ 259 levels “diferente”,”melhor”,..: 1 1
2 3 1 4 5 6 7 8 ...
##  $ Transc.Fon   : Factor w/ 259 levels “d<i>-f<e>-'RE-te”,..: 1 1 2
3 1 4 5 6 7 8 ...
##  $ VOGAL        : Factor w/ 1 level “e”: 1 1 1 1 1 1 1 1 1 1 ...
##  $ F1           : num [1:677] 613 656 573 735 567 ...
##  $ F2           : num [1:677] 2014 1848 2413 1656 1375 ...
##  $ F1.NORM      : num [1:677] 447 464 431 496 429 ...
```

```
##  $ F2.NORM      : num [1:677] 1698 1611 1905 1512 1366 ...
##  $ CONT.PREC    : Factor w/ 5 levels “dental.alveolar”,..: 2 2 2 5
2 4 2 3 2 1 ...
##  $ CONT.SEG     : Factor w/ 5 levels “dental.alveolar”,..: 5 5 3 2
5 5 1 5 3 2 ...
##  $ VOGAL.SIL.SEG: Factor w/ 11 levels “a”,”aw”,”A”,”\u0097”,..: 5 5
4 7 5 5 1 5 1 5 ...
##  $ F1.SIL.SEG   : num [1:677] 569 524 686 652 661 ...
##  $ F2.SIL.SEG   : num [1:677] 1674 2428 1497 2159 1865 ...
##  $ F1.SEG.NORM  : num [1:677] 350 336 385 375 378 ...
##  $ F2.SEG.NORM  : num [1:677] 1360 1724 1274 1594 1452 ...
##  $ VOGAL.TONICA : Factor w/ 14 levels “e”,”o”,”ow”,”a”,..: 9 9 2 1
9 9 4 9 4 9 ...
##  $ DIST.TONICA  : num [1:677] 1 1 1 1 1 1 1 1 1 2 ...
##  $ ESTR.SIL.PRET: Factor w/ 5 levels “CV”,”CVs”,”CCV”,..: 1 1 1 1 1
1 1 1 1 1 ...
##  $ Begin.Time.s : num [1:677] 219 226 576 584 614 ...
##  $ End.Time.s   : num [1:677] 219 226 576 584 614 ...
##  $ Duration.ms  : num [1:677] 10.4 11.6 30.3 17.5 17.1 ...
##  $ AMOSTRA      : Factor w/ 2 levels “PBSP”,”SP2010”: 1 1 1 1 1 1 1
1 1 1 ...
##  $ PARTICIPANTE : Factor w/ 14 levels “MartaS”,”JosaneV”,..: 1 1 1
1 1 1 1 1 1 1 ...
##  $ SEXO         : Factor w/ 2 levels “feminino”,”masculino”: 1 1 1
1 1 1 1 1 1 1 ...
##  $ IDADE        : int [1:677] 32 32 32 32 32 32 32 32 32 32 ...
##  $ IDADE.CHEGADA: int [1:677] 18 18 18 18 18 18 18 18 18 18 ...
##  $ ANOS.SP      : int [1:677] 14 14 14 14 14 14 14 14 14 14 ...
##  $ CONTEXTO     : Factor w/ 632 levels “diferente o clima de eu com
ele”,..: 1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, “spec”)=
##   .. cols(
##   ..   .default = col_factor(),
##   ..   PALAVRA = col_factor(levels = NULL, ordered = FALSE, include
_na = FALSE),
##   ..   Transc.Fon = col_factor(levels = NULL, ordered = FALSE, incl
ude_na = FALSE),
##   ..   VOGAL = col_factor(levels = c(“i”, “e”, “a”, “o”, “u”), orde
red = FALSE, include_na = FALSE),
##   ..   F1 = col_double(),
##   ..   F2 = col_double(),
##   ..   F1.NORM = col_double(),
##   ..   F2.NORM = col_double(),
##   ..   CONT.PREC = col_factor(levels = NULL, ordered = FALSE, inclu
de_na = FALSE),
##   ..   CONT.SEG = col_factor(levels = NULL, ordered = FALSE, includ
e_na = FALSE),
##   ..   VOGAL.SIL.SEG = col_factor(levels = NULL, ordered = FALSE, i
nclude_na = FALSE),
##   ..   F1.SIL.SEG = col_double(),
##   ..   F2.SIL.SEG = col_double(),
##   ..   F1.SEG.NORM = col_double(),
##   ..   F2.SEG.NORM = col_double(),
##   ..   VOGAL.TONICA = col_factor(levels = NULL, ordered = FALSE, in
```

```
clude_na = FALSE),
##   ..   DIST.TONICA = col_double(),
##   ..   ESTR.SIL.PRET = col_factor(levels = NULL, ordered = FALSE, i
nclude_na = FALSE),
##   ..   Begin.Time.s = col_double(),
##   ..   End.Time.s = col_double(),
##   ..   Duration.ms = col_double(),
##   ..   AMOSTRA = col_factor(levels = NULL, ordered = FALSE, include
_na = FALSE),
##   ..   PARTICIPANTE = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   SEXO = col_factor(levels = NULL, ordered = FALSE, include_na
= FALSE),
##   ..   IDADE = col_integer(),
##   ..   IDADE.CHEGADA = col_integer(),
##   ..   ANOS.SP = col_integer(),
##   ..   CONTEXTO = col_factor(levels = NULL, ordered = FALSE, includ
e_na = FALSE)
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Imagine que um pesquisador tenha levantado a hipótese de que altura da vogal /e/ pretônica depende das variáveis AMOSTRA, SEXO, F1.SEG.NORM, CONT.PREC e CONT.SEG. A partir de `VOGAL_e2`, crie primeiro um modelo linear chamado `mod`, com F1.NORM como variável resposta e as variáveis AMOSTRA + SEXO + F1.SEG.NORM + CONT.PREC + CONT.SEG como variáveis previsoras.

```
mod <- lm(F1.NORM ~
            AMOSTRA +
            SEXO +
            F1.SEG.NORM +
            CONT.PREC +
            CONT.SEG,
          data = VOGAL_e2)
```

Veja o resultado de `mod` com `summary()`.

```
summary(mod)
```

```
##
## Call:
## lm(formula = F1.NORM ~ AMOSTRA + SEXO + F1.SEG.NORM + CONT.PREC +
##     CONT.SEG, data = VOGAL_e2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.457 -16.874  -0.574  14.869  70.772
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               397.14855   10.42949  38.079  < 2e-16 **
```

```
*
## AMOSTRASP2010               -8.46124    1.93283  -4.378 1.39e-05 **
*
## SEXOmasculino               -2.80518    1.95100  -1.438 0.150958
## F1.SEG.NORM                  0.11337    0.02459   4.611 4.80e-06 **
*
## CONT.PREClabial             -0.56978    2.88994  -0.197 0.843762
## CONT.PRECpalatal.sibilante  -8.27567    2.96849  -2.788 0.005458 **
## CONT.PRECvelar              -1.31555    7.46441  -0.176 0.860157
## CONT.PRECvibrante            3.17324    3.38029   0.939 0.348201
## CONT.SEGlabial             -11.78880    4.14795  -2.842 0.004619 **
## CONT.SEGpalatal.sibilante  -14.49915    3.96319  -3.658 0.000274 **
*
## CONT.SEGvelar               -6.72795    3.56755  -1.886 0.059748 .
## CONT.SEGvibrante            -3.72273    3.60498  -1.033 0.302137
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.61 on 665 degrees of freedom
## Multiple R-squared:  0.101,  Adjusted R-squared:  0.08617
## F-statistic: 6.795 on 11 and 665 DF,  p-value: 6.897e-11
```

O resultado do modelo inclui todas as variáveis previsoras (e respectivos fatores, em caso de variáveis nominais), suas estimativas e valores de significância em relação ao coeficiente linear (Intercept). Como visto na lição passada, sua leitura depende do conhecimento de quais são os níveis de referência.

Para a variável AMOSTRA, qual é o nível de referência? Veja o resultado de `str()`, se necessário.

- PBSP
- SP2010

Para a variável SEXO, qual é o nível de referência?

- feminino
- masculino

Para a variável F1.SEG.NORM, qual é o nível de referência?

- F1.SEG.NORM = 0 Hz
- F1.SEG.NORM = 400 Hz
- F1.SEG.NORM = 600 Hz

Para a variável CONT.PREC, qual é o nível de referência?

- dental.alveolar
- labial
- palatal.sibilante
- velar
- vibrante

Para a variável CONT.SEG, qual é o nível de referência?

- dental.alveolar
- labial
- palatal.sibilante
- velar
- vibrante

Os respectivos valores de referência são o zero ou o primeiro nível de acordo com a ordem em que aparecem na planilha ou de acordo com o especificado pelo usuário no momento da importação dos dados. Em `mod`, o coeficiente linear 397,14855 se refere, portanto, à estimativa do valor de F1.NORM para falantes paraibanos do sexo feminino, quando F1.SEG.NORM é zero e quando o contexto precedente e o contexto seguinte são consoantes dental-alveolares. As estimativas para todos os outros cenários possíveis podem ser deduzidos a partir da soma dos valores dos coeficientes angulares.

Vemos que a estimativa de F1.NORM para AMOSTRASP2010 difere significativamente em relação a PBSP; a cada unidade de F1.NORM; quando CONT.PREC é palatal-sibilante (em relação à dental-alveolar); e quando CONT.SEG é labial ou palatal-sibilante (em relação à dental-alveolar). Não há diferença significativa para as estimativas de homens e mulheres, e em relação aos demais contextos precedentes ou seguintes em relação à dental-alveolar. Sobre a variável SEXO, um pesquisador pode se perguntar se é pertinente manter essa variável no modelo, uma vez que parece não estar contribuindo muito para a estimativa da variável resposta. Sobre as variáveis CONT.PREC e CONT.SEG, o pesquisador pode se perguntar se as variáveis são relevantes de modo global, já que

apenas um ou dois fatores de cada variável mostram diferenças significativas em relação ao nível de referência.

A função `step()` compara diferentes modelos com e sem a inclusão de diferentes variáveis e reporta, ao final, quais variáveis devem ser mantidas. Para isso, baseia-se no AIC (Akaike Information Criterion), que penaliza o modelo se tem muitas variáveis – de modo semelhante ao R^2 ajustado. A função pode ser aplicada em três direções: (i) forward; (ii) backward; e (iii) both. Comecemos com a opção "forward".

Para isso, primeiro precisamos criar um modelo que não inclui qualquer variável previsora. Crie então um modelo chamado `m0`, com a seguinte linha de comando: `m0 <- lm(F1.NORM ~ 1, data = VOGAL_e2)`.

```
m0 <- lm(F1.NORM ~ 1, data = VOGAL_e2)
```

Vamos agora aplicar a função `step()`, com os seguintes argumentos: (i) `m0`; (ii) direction = "forward"; (iii) scope = ~ AMOSTRA + SEXO + F1.SEG.NORM + CONT.PREC + CONT.SEG. Guarde o resultado num objeto chamado `m.fw`.

```
m.fw <- step(m0, direction = "forward", scope = ~ AMOSTRA + SEXO + F1.
SEG.NORM + CONT.PREC + CONT.SEG)

## Start:  AIC=4399.27
## F1.NORM ~ 1
##
##                Df Sum of Sq    RSS    AIC
## + F1.SEG.NORM  1   15538.4 432617 4377.4
## + AMOSTRA      1   11939.5 436216 4383.0
## + CONT.SEG     4   10796.0 437359 4390.8
## + CONT.PREC    4    6421.6 441733 4397.5
## <none>                     448155 4399.3
## + SEXO         1     336.6 447818 4400.8
##
## Step:  AIC=4377.38
## F1.NORM ~ F1.SEG.NORM
##
##              Df Sum of Sq    RSS    AIC
## + AMOSTRA     1   10793.0 421824 4362.3
## + CONT.SEG    4    9127.0 423490 4370.9
## + CONT.PREC   4    6876.3 425740 4374.5
## <none>                    432617 4377.4
## + SEXO        1     422.5 432194 4378.7
##
## Step:  AIC=4362.27
## F1.NORM ~ F1.SEG.NORM + AMOSTRA
##
##              Df Sum of Sq    RSS    AIC
```

```
## + CONT.SEG   4    9685.2 412138 4354.5
## + CONT.PREC  4    7346.8 414477 4358.4
## <none>                   421824 4362.3
## + SEXO       1     395.9 421428 4363.6
##
## Step:  AIC=4354.55
## F1.NORM ~ F1.SEG.NORM + AMOSTRA + CONT.SEG
##
##             Df Sum of Sq    RSS    AIC
## + CONT.PREC  4    8010.5 404128 4349.3
## <none>                   412138 4354.5
## + SEXO       1     903.8 411235 4355.1
##
## Step:  AIC=4349.26
## F1.NORM ~ F1.SEG.NORM + AMOSTRA + CONT.SEG + CONT.PREC
##
##        Df Sum of Sq    RSS    AIC
## + SEXO  1    1252.4 402875 4349.2
## <none>              404128 4349.3
##
## Step:  AIC=4349.16
## F1.NORM ~ F1.SEG.NORM + AMOSTRA + CONT.SEG + CONT.PREC + SEXO
```

Acima, especificamos que queremos partir do modelo `m0`, sem variáveis previsoras, e avaliar se a adição de novas variáveis melhora seu poder explanatório. O R então calculou o AIC para diferentes modelos e incluiu as variáveis F1.NORM, AMOSTRA, CONT.SEG, CONT.PREC e SEXO – nessa ordem. Essa ordem de seleção indica a importância relativa de cada variável para explicar a variação em F1.NORM. Digite agora `m.fw` para ver o cálculo de coeficientes de cada previsor.

```
m.fw

##
## Call:
## lm(formula = F1.NORM ~ F1.SEG.NORM + AMOSTRA + CONT.SEG + CONT.PREC
+
##     SEXO, data = VOGAL_e2)
##
## Coefficients:
##                (Intercept)                F1.SEG.NORM
##                   397.1486                     0.1134
##              AMOSTRASP2010             CONT.SEGlabial
##                   -8.4612                   -11.7888
##  CONT.SEGpalatal.sibilante              CONT.SEGvelar
##                  -14.4992                    -6.7280
##           CONT.SEGvibrante            CONT.PREClabial
##                   -3.7227                    -0.5698
## CONT.PRECpalatal.sibilante             CONT.PRECvelar
##                   -8.2757                    -1.3156
```

```
##          CONT.PRECvibrante                SEXOmasculino
##                     3.1732                      -2.8052
```

Vamos agora fazer um modelo "de trás para frente". A função `step()` toma novamente como argumentos o modelo de qual se quer partir – aqui usaremos `mod`, o modelo completo feito acima – e a direção – direction = "backward". Neste caso, não é necessário especificar o argumento scope. Digite então `m.bw <- step(mod, direction = "backward")`.

```
m.bw <- step(mod, direction = "backward")

## Start:  AIC=4349.16
## F1.NORM ~ AMOSTRA + SEXO + F1.SEG.NORM + CONT.PREC + CONT.SEG
##
##                Df Sum of Sq    RSS    AIC
## <none>                      402875 4349.2
## - SEXO          1    1252.4 404128 4349.3
## - CONT.PREC     4    8359.2 411235 4355.1
## - CONT.SEG      4   11162.6 414038 4359.7
## - AMOSTRA       1   11609.9 414485 4366.4
## - F1.SEG.NORM   1   12881.8 415757 4368.5
```

No modelo "de trás para frente", o R começa com o modelo completo e tenta excluir variáveis uma a uma. Nenhuma variável foi excluída, e o R as manteve na ordem em que foram especificadas dentro do modelo inicial – AMOSTRA, SEXO, F1.SEG.NORM, CONT.PREC e CONT.SEG. O importante aqui é checar se as mesmas variáveis que foram incluídas no modelo "para frente" também são incluídas no modelo "de trás para frente". Caso isso não ocorra, é um sinal de que há interação entre variáveis do modelo. Cabe ao pesquisador encontrá-la(s) e incluir a interação num novo modelo.

Veja também o resultado guardado em `m.bw`.

```
m.bw

##
## Call:
## lm(formula = F1.NORM ~ AMOSTRA + SEXO + F1.SEG.NORM + CONT.PREC +
##     CONT.SEG, data = VOGAL_e2)
##
## Coefficients:
##                (Intercept)                AMOSTRASP2010
##                   397.1486                      -8.4612
##              SEXOmasculino                  F1.SEG.NORM
##                   -2.8052                       0.1134
##           CONT.PREClabial  CONT.PRECpalatal.sibilante
##                   -0.5698                      -8.2757
```

```
##             CONT.PRECvelar           CONT.PRECvibrante
##                    -1.3156                      3.1732
##             CONT.SEGlabial   CONT.SEGpalatal.sibilante
##                   -11.7888                    -14.4992
##              CONT.SEGvelar            CONT.SEGvibrante
##                    -6.7280                    -3.7227
```

Note que os coeficientes angulares são os mesmos calculados para o modelo forward. Por fim, apliquemos a direção "both". Neste caso, o programa começa executando o mesmo que a direção "forward" mas, toda vez que inclui uma nova variável, ele tenta excluir alguma variável que possa não mais estar contribuindo para o modelo. A partir da linha de comando em que você criou `m.fw`, mude o nome do objeto para `m.both` e apague o argumento `direction`; não é necessário especificá-lo pois este é o valor *default* da função.

```
m.both <- step(m0, scope = ~ AMOSTRA + SEXO + F1.SEG.NORM + CONT.PREC
+ CONT.SEG)

## Start:  AIC=4399.27
## F1.NORM ~ 1
##
##               Df Sum of Sq    RSS    AIC
## + F1.SEG.NORM  1   15538.4 432617 4377.4
## + AMOSTRA      1   11939.5 436216 4383.0
## + CONT.SEG     4   10796.0 437359 4390.8
## + CONT.PREC    4    6421.6 441733 4397.5
## <none>                     448155 4399.3
## + SEXO         1     336.6 447818 4400.8
##
## Step:  AIC=4377.38
## F1.NORM ~ F1.SEG.NORM
##
##               Df Sum of Sq    RSS    AIC
## + AMOSTRA      1   10793.0 421824 4362.3
## + CONT.SEG     4    9127.0 423490 4370.9
## + CONT.PREC    4    6876.3 425740 4374.5
## <none>                     432617 4377.4
## + SEXO         1     422.5 432194 4378.7
## - F1.SEG.NORM  1   15538.4 448155 4399.3
##
## Step:  AIC=4362.27
## F1.NORM ~ F1.SEG.NORM + AMOSTRA
##
##               Df Sum of Sq    RSS    AIC
## + CONT.SEG     4    9685.2 412138 4354.5
## + CONT.PREC    4    7346.8 414477 4358.4
## <none>                     421824 4362.3
## + SEXO         1     395.9 421428 4363.6
## - AMOSTRA      1   10793.0 432617 4377.4
```

```
## - F1.SEG.NORM  1   14391.9 436216 4383.0
##
## Step:  AIC=4354.55
## F1.NORM ~ F1.SEG.NORM + AMOSTRA + CONT.SEG
##
##               Df Sum of Sq    RSS    AIC
## + CONT.PREC    4    8010.5 404128 4349.3
## <none>                     412138 4354.5
## + SEXO         1     903.8 411235 4355.1
## - CONT.SEG     4    9685.2 421824 4362.3
## - AMOSTRA      1   11351.2 423490 4370.9
## - F1.SEG.NORM  1   12965.0 425103 4373.5
##
## Step:  AIC=4349.26
## F1.NORM ~ F1.SEG.NORM + AMOSTRA + CONT.SEG + CONT.PREC
##
##               Df Sum of Sq    RSS    AIC
## + SEXO         1    1252.4 402875 4349.2
## <none>                     404128 4349.3
## - CONT.PREC    4    8010.5 412138 4354.5
## - CONT.SEG     4   10348.9 414477 4358.4
## - AMOSTRA      1   11627.3 415755 4366.5
## - F1.SEG.NORM  1   12739.3 416867 4368.3
##
## Step:  AIC=4349.16
## F1.NORM ~ F1.SEG.NORM + AMOSTRA + CONT.SEG + CONT.PREC + SEXO
##
##               Df Sum of Sq    RSS    AIC
## <none>                     402875 4349.2
## - SEXO         1    1252.4 404128 4349.3
## - CONT.PREC    4    8359.2 411235 4355.1
## - CONT.SEG     4   11162.6 414038 4359.7
## - AMOSTRA      1   11609.9 414485 4366.4
## - F1.SEG.NORM  1   12881.8 415757 4368.5
```

E veja os valores de coeficiente angular no resultado guardado em `m.both`.

```
m.both

##
## Call:
## lm(formula = F1.NORM ~ F1.SEG.NORM + AMOSTRA + CONT.SEG + CONT.PREC
+
##     SEXO, data = VOGAL_e2)
##
## Coefficients:
##                (Intercept)                 F1.SEG.NORM
##                   397.1486                      0.1134
##              AMOSTRASP2010              CONT.SEGlabial
##                   -8.4612                   -11.7888
##  CONT.SEGpalatal.sibilante               CONT.SEGvelar
##                   -14.4992                    -6.7280
##           CONT.SEGvibrante             CONT.PREClabial
##                   -3.7227                    -0.5698
```

```
## CONT.PRECpalatal.sibilante              CONT.PRECvelar
##                    -8.2757                     -1.3156
##          CONT.PRECvibrante               SEXOmasculino
##                     3.1732                     -2.8052
```

Novamente, o que se espera é que as variáveis selecionadas sejam as mesmas e que os coeficientes angulares também tenham os mesmos valores. Caso isso não ocorra, há uma forte evidência de que as variáveis não são independentes entre si e deve-se verificar se há interações entre as variáveis do modelo.

Outra função que permite avaliar se vale a pena manter determinada variável no modelo estatístico é `drop1()`. A função requer um modelo com inclusão de todas as variáveis previsoras que se quer testar – no nosso caso, `mod` – e o tipo de teste a se aplicar – aqui vamos usar "F", que toma por base a estatística-F para comparar modelos (aquela que aparece ao pé do resultado de `summary()`; outra opção seria o teste "Chisq"). Digite então `drop1(mod, test = "F")`.

```
drop1(mod, test = "F")
```

```
## Single term deletions
##
## Model:
## F1.NORM ~ AMOSTRA + SEXO + F1.SEG.NORM + CONT.PREC + CONT.SEG
##             Df Sum of Sq    RSS    AIC F value    Pr(>F)
## <none>                   402875 4349.2
## AMOSTRA      1   11609.9 414485 4366.4 19.1637 1.393e-05 ***
## SEXO         1    1252.4 404128 4349.3  2.0673  0.150958
## F1.SEG.NORM  1   12881.8 415757 4368.5 21.2632 4.802e-06 ***
## CONT.PREC    4    8359.2 411235 4355.1  3.4495  0.008400 **
## CONT.SEG     4   11162.6 414038 4359.7  4.6063  0.001127 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

O resultado de `drop1()` apresenta um valor de significância para cada variável previsora incluída no modelo (assim como outras medidas estatísticas). Aqui vemos que SEXO não se correlaciona significativamente com F1.NORM – diferentemente do resultado da função `step()`. Cabe ao pesquisador decidir se ele seguirá o resultado da função `step()` ou `drop1()`. Os resultados podem diferir porque cada teste se baseia em um critério diferente (AIC para `step()`; estatística-F ou qui-quadrado para `drop1()`). Por mais frustrante que isso possa ser, não há aqui fórmula mágica para lhe dizer qual decisão

tomar. Mas esse fato é importante para nos lembrar de que a análise estatística não dá todas as respostas, pois está sempre sujeita à interpretação do pesquisador.

Com `drop1()`, o pesquisador pode ainda atualizar o modelo e continuar aplicando a função para tentar excluir mais uma variável, pois a função exclui as variáveis previsoras uma de cada vez. Aqui, não a aplicaremos mais pois todas as variáveis remanescentes são significativas.

Temos então boas evidências de que as variáveis F1.SEG.NORM, AMOSTRA, CONT.SEG e CONT.PREC se correlacionam com a altura da vogal /e/ pretônica, mas evidências não tão fortes quanto ao papel da variável SEXO. Vamos criar um modelo linear chamado `modelo` com a inclusão apenas das variáveis selecionadas tanto em `step()` quanto em `drop1()` – ou seja, excluindo a variável SEXO –, nos dados de `VOGAL_e2`.

```
modelo <- lm(F1.NORM ~ F1.SEG.NORM + AMOSTRA + CONT.SEG + CONT.PREC, d
ata = VOGAL_e2)
```

Após chegar a um modelo de seus dados, ainda é necessário fazer alguns testes para checar se o modelo não viola os pressupostos básicos de modelos lineares. A checagem desses pressupostos é importante para se certificar de que seus resultados são confiáveis.

O primeiro deles é fácil: a variável resposta deve ser numérica e contínua. Numa variável contínua, a relação entre os valores das observações é quantitativa; por exemplo, é possível dizer que uma vogal com 300 Hz de F1 tem a metade do valor de uma vogal com 600 Hz. Para variáveis nominais, aplicamos outros tipos de teste (regressão logística – que veremos nas Lições 14 e 15 –, multinomial ou de Poisson – que não serão vistas neste curso).

Uma segunda checagem a fazer é verificar se a relação entre a variável resposta e uma variável previsora numérica é de fato linear. Isso pode ser feito por meio da função `plot()` – como fizemos na Lição 11 – ou ainda por meio da função `crPlot()` do pacote car.

Em nosso modelo, apenas F1.SEG.NORM é uma variável numérica. Digite então `crPlot(modelo, var = "F1.SEG.NORM")` para checar se a relação entre essa variável e a variável resposta é linear.

```
crPlot(modelo, var = "F1.SEG.NORM")
```

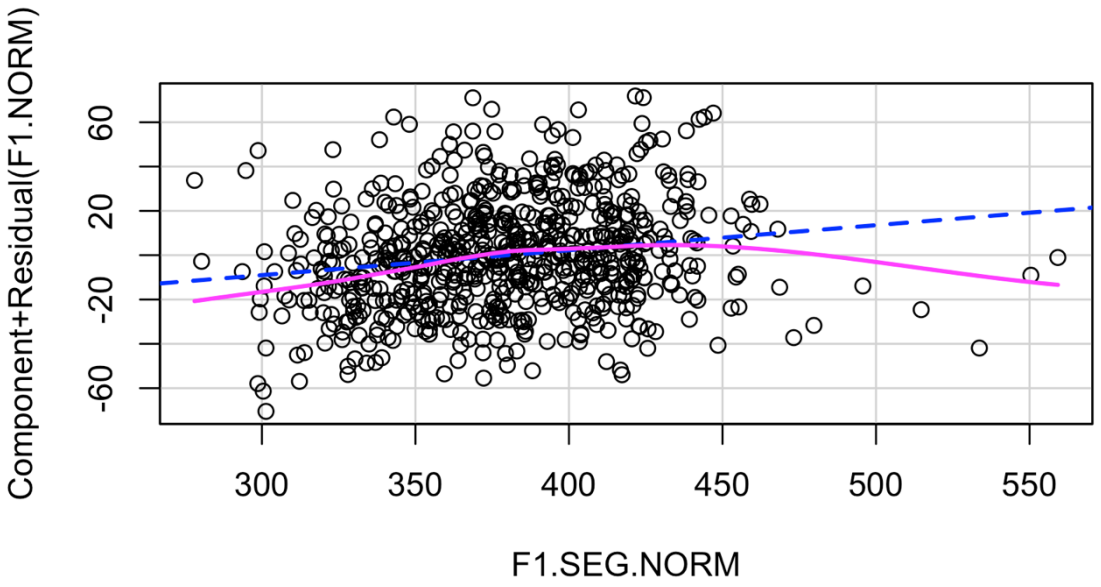

Figura 13.1: Plot dos valores previstos e observados de F1.SEG.NORM no modelo. Fonte: própria.

Na figura plotada, a linha pontilhada corresponde aos valores previstos pelo modelo, e a linha contínua corresponde a uma linha de regressão suavizada que segue mais fielmente a distribuição observada. Se a linha contínua não segue a linha pontilhada, isso é um sinal de que o pressuposto de relação de linearidade entre as variáveis não é cumprido. Na Figura 13.1, vemos que a linha contínua mais se distancia da linha pontilhada nos maiores valores de F1.SEG.NORM. No início da análise, na lição anterior, retiramos dados acima de 500 Hz apenas para a variável resposta (F1.NORM), mas não para a vogal da sílaba seguinte. Esses correspondem apenas a quatro dados, de modo que retirá-los não causará grande impacto na amostra, ao mesmo tempo que melhorará o modelo. Crie então um novo subconjunto de dados, chamado `VOGAL_e3`, com os dados de F1.SEG.NORM abaixo de 500 Hz.

```
VOGAL_e3 <- filter(VOGAL_e2, F1.SEG.NORM < 500)
```

Crie agora um novo modelo linear, chamado `modelo2`, com a mesma fórmula F1.NORM ~ AMOSTRA + F1.SEG.NORM + CONT.PREC + CONT.SEG, no conjunto `VOGAL_e3`.

```
modelo2 <- lm(F1.NORM ~ AMOSTRA + F1.SEG.NORM + CONT.PREC + CONT.SEG,
data = VOGAL_e3)
```

E aplique a função `crPlot()` a `modelo2`, com var = "F1.SEG.NORM", para checar se a relação entre as variáveis se aproxima mais da linearidade.

```
crPlot(modelo2, var = "F1.SEG.NORM")
```

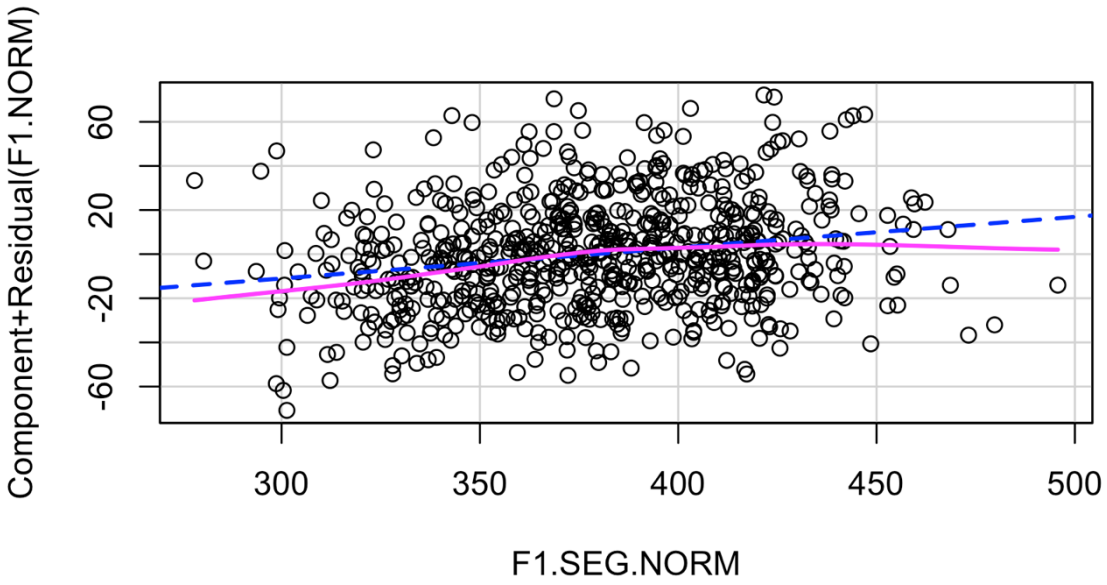

Figura 13.2: Plot dos valores previstos e observados de F1.SEG.NORM no modelo2. Fonte: própria.

Em relação à Figura 13.1, a linha contínua na Figura 13.2 se aproxima muito mais da linha pontilhada.

Outro pressuposto fundamental num modelo linear é que as variáveis previsoras não sejam dependentes entre si. Tal dependência é chamada de multicolinearidade, que ocorre quando algumas variáveis previsoras do modelo se referem a um mesmo efeito. No conjunto original de pretônicas, por exemplo, F1.SEG.NORM se refere à altura da vogal da sílaba seguinte; há ainda outra variável na planilha, chamada VOGAL.SIL.SEG, que

codifica a vogal em vez de seu valor de F1. Trata-se de uma mesma variável, vista de duas maneiras (a primeira, contínua; a segunda, nominal), de modo que não devem ser incluídas em um mesmo modelo. Em geral, o próprio pesquisador é capaz de prever que duas variáveis não são independentes entre si e já não as incluir num mesmo teste estatístico.

Mas há uma função do pacote car, `vif()`, que permite checar se variáveis incluídas num modelo são colineares. Aplique-a a `modelo2`.

```
vif(modelo2)
```

```
##                 GVIF Df GVIF^(1/(2*Df))
## AMOSTRA     1.046156  1        1.022818
## F1.SEG.NORM 1.038512  1        1.019074
## CONT.PREC   1.973176  4        1.088669
## CONT.SEG    1.978711  4        1.089050
```

Quando duas variáveis são colineares, os valores de GVIF (da primeira coluna) e GVIF-ajustado (da terceira coluna) são altos, acima de 5. Nosso modelo está ok quanto a esse critério, já que todos os valores são abaixo de 2. Caso verifique valores acima de 5, você deve considerar não incluir as variáveis colineares no mesmo modelo.

Outro pressuposto já foi mencionado na lição anterior: a distribuição dos resíduos é normal, com valores simétricos e mediana zero. Digite `summary(modelo2)` para visualizar o resultado de nosso último modelo.

```
summary(modelo2)
```

```
##
## Call:
## lm(formula = F1.NORM ~ AMOSTRA + F1.SEG.NORM + CONT.PREC + CONT.SEG
,
##     data = VOGAL_e3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -59.972 -16.741  -1.065  14.618  71.855
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                384.43394   10.83542  35.479  < 2e-16 **
*
## AMOSTRASP2010               -8.26394    1.93126  -4.279 2.15e-05 **
*
## F1.SEG.NORM                  0.13958    0.02592   5.385 1.01e-07 **
*
```

```
## CONT.PREClabial             -0.07849    2.87425  -0.027 0.978223
## CONT.PRECpalatal.sibilante  -8.02946    2.95385  -2.718 0.006734 **
## CONT.PRECvelar              -1.14571    7.42957  -0.154 0.877492
## CONT.PRECvibrante            3.65342    3.38749   1.079 0.281201
## CONT.SEGlabial             -10.12030    4.10273  -2.467 0.013888 *
## CONT.SEGpalatal.sibilante  -13.55855    3.92429  -3.455 0.000585 **
*
## CONT.SEGvelar               -6.02371    3.53098  -1.706 0.088484 .
## CONT.SEGvibrante            -2.44226    3.59664  -0.679 0.497350
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.49 on 662 degrees of freedom
## Multiple R-squared:  0.1099, Adjusted R-squared:  0.09646
## F-statistic: 8.174 on 10 and 662 DF,  p-value: 1.524e-12
```

Vemos nos resíduos que a distribuição não parece ser normal, principalmente pelos valores Min (-59,972) e Max (71,855). Já vimos, em lições anteriores, uma função para verificar numericamente se uma distribuição é normal. Qual é ela?

- `bonferroni.test()`
- `chisq.test()`
- `shapiro.test()`
- `t.test()`

Aplique então o teste de Shapiro a `modelo2$residuals`.

```
shapiro.test(modelo2$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  modelo2$residuals
## W = 0.99349, p-value = 0.005183
```

O que informa o teste acima?

- a distribuição dos resíduos de `modelo2` é normal
- a distribuição dos resíduos de `modelo2` não é normal

A assunção de normalidade dos resíduos torna-se menos importante quando se trabalha com uma amostra grande. Neste caso, em que temos mais de 600 dados, podemos ficar razoavelmente seguros de que a não normalidade dos resíduos não afeta as estimativas de maneira danosa. Em amostras menores, a não normalidade dos resíduos

pode ser mais problemática, visto que cada observação tem maior peso quanto menor for a amostra (Lição 6).

Por fim, é importante avaliar se cada observação é independente uma das outras. Idealmente, cada dado coletado da população deveria ter a mesma chance de entrar na amostra. Em estudos linguísticos, isso raramente é o caso. Nos dados de vogais pretônicas, em que estamos trabalhando, as mais de 674 ocorrências de /e/ pretônico advêm de 14 falantes (7 paraibanos e 7 paulistanos), o que significa que vieram de um conjunto pequeno da população. De cada falante, foram extraídos cerca de 50 dados da vogal /e/, o que significa que os dados em cada subconjunto não são independentes uns dos outros.

Grande parte da variabilidade nos dados linguísticos se deve ao próprio falante. Na estatística, este tipo de variável é chamada de *efeito aleatório*, pois normalmente muda a cada amostra. Efeitos aleatórios se contrapõem a efeitos *fixos*, que podem ser facilmente reproduzidos em outros estudos. Por exemplo, a variável SEXO é um efeito fixo, pois seus níveis – feminino e masculino – podem ser facilmente reproduzidos em nova amostra de falantes paraibanos e paulistanos. Se escolhêssemos novos falantes aleatoriamente, provavelmente teríamos homens e mulheres na nova amostra. Por outro lado, essa mesma nova amostragem dificilmente conteria os exatos 14 falantes da primeira amostra. Considerando-se que muito da variabilidade nos dados vem dos próprios indivíduos, é importante levar em conta sua contribuição para o resultado final dos modelos estatísticos.

Outro efeito aleatório comum em estudos linguísticos é o *item lexical*. Certas palavras podem ter comportamento idiossincrático, independentemente de condicionamentos mais gerais como classe morfológica, contexto fonológico precedente a um segmento alvo, contexto fonológico seguinte, função sintática etc. Seguindo o mesmo raciocínio acima, uma nova amostra aleatória de dados linguísticos (coletados da mesma maneira) muito provavelmente conterá substantivos, verbos, advérbios..., mas dificilmente conterá exatamente os mesmos itens lexicais que compõem a amostra original.

Efeitos aleatórios, quando existirem, *sempre* devem ser incluídos nos modelos estatísticos. Modelos que incluem tanto efeitos fixos quanto efeitos aleatórios são chamados de *modelos de efeitos mistos*. O motivo de termos deixado os efeitos aleatórios para o final é bastante prático: você verá daqui a pouco que esses modelos demoram um bocado para rodar no R. Minha recomendação, portanto, é que você chegue a um modelo satisfatório com efeitos fixos, cheque se esse modelo não viola os pressupostos de modelos lineares (fazendo novos ajustes, se necessários) e, apenas como última etapa, inclua os efeitos aleatórios. Este é um excelente exemplar da expressão "por último, mas não menos importante": qualquer estudo que tenha efeitos aleatórios deve incluí-los na modelagem estatística.

Os modelos de efeitos mistos são implementados no R por meio do pacote lme4, que carregamos no início desta lição. O pacote lmerTest, também já carregado, fornece valores-*p* para as estimativas.

A função `lmer()` toma os mesmos argumentos de `lm()`, com a diferença de que requer a inclusão de efeitos aleatórios. Estas são indicadas dentro da fórmula com a notação (1|varaleatoria). Vamos então criar um modelo de efeitos mistos a partir do último modelo criado (modelo2). A partir daquela linha de comando, modifique (i) o nome do objeto para `mod1.lmer`; (ii) a função para lmer; e (iii) inclua duas novas variáveis na fórmula: + (1|PARTICIPANTE) + (1|PALAVRA).

```
mod1.lmer <- lmer(F1.NORM ~
                    AMOSTRA +
                    F1.SEG.NORM +
                    CONT.PREC +
                    CONT.SEG +
                    (1|PARTICIPANTE) +
                    (1|PALAVRA),
                  data = VOGAL_e3)
```

Aplique a função `summary()` a `mod1.lmer`.

```
summary(mod1.lmer)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method
[
## lmerModLmerTest]
## Formula:
## F1.NORM ~ AMOSTRA + F1.SEG.NORM + CONT.PREC + CONT.SEG + (1 |
```

```
##     PARTICIPANTE) + (1 | PALAVRA)
##    Data: VOGAL_e3
##
## REML criterion at convergence: 6166
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.37177 -0.67890 -0.03253  0.58279  2.88123
##
## Random effects:
##  Groups       Name        Variance Std.Dev.
##  PALAVRA      (Intercept)  29.805   5.459
##  PARTICIPANTE (Intercept)   9.184   3.030
##  Residual                 565.113  23.772
## Number of obs: 673, groups:  PALAVRA, 255; PARTICIPANTE, 14
##
## Fixed effects:
##                             Estimate Std. Error        df t value
## (Intercept)                379.20012   11.37177 414.46165  33.346
## AMOSTRASP2010               -8.06663    2.52093  12.06047  -3.200
## F1.SEG.NORM                  0.14891    0.02704 463.76920   5.508
## CONT.PREClabial              0.54952    3.22933 134.99159   0.170
## CONT.PRECpalatal.sibilante  -5.90882    3.32399 128.45128  -1.778
## CONT.PRECvelar               0.69095    7.79936 262.83540   0.089
## CONT.PRECvibrante            4.90485    3.62336 232.44699   1.354
## CONT.SEGlabial              -9.49738    4.39146 230.83428  -2.163
## CONT.SEGpalatal.sibilante  -12.53629    4.11419 302.56280  -3.047
## CONT.SEGvelar               -5.57322    3.86767 170.43261  -1.441
## CONT.SEGvibrante            -2.27535    3.83996 242.30288  -0.593
##                            Pr(>|t|)
## (Intercept)                 < 2e-16 ***
## AMOSTRASP2010               0.00759 **
## F1.SEG.NORM                6.04e-08 ***
## CONT.PREClabial             0.86514
## CONT.PRECpalatal.sibilante  0.07783 .
## CONT.PRECvelar              0.92947
## CONT.PRECvibrante           0.17716
## CONT.SEGlabial              0.03159 *
## CONT.SEGpalatal.sibilante   0.00251 **
## CONT.SEGvelar               0.15143
## CONT.SEGvibrante            0.55404
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) AMOSTR F1.SEG CONT.PRECl CONT.PREC. CONT.PRECvl
## AMOSTRASP20 -0.170
## F1.SEG.NORM -0.924  0.040
## CONT.PREClb -0.159  0.066 -0.033
## CONT.PRECp. -0.203  0.086  0.013  0.523
## CONT.PRECvl -0.094  0.064  0.034  0.220      0.183
## CONT.PRECvb -0.229  0.111  0.034  0.452      0.440      0.193
## CONT.SEGlbl -0.329  0.048  0.059  0.331      0.280      0.035
## CONT.SEGpl. -0.298 -0.011  0.095  0.055      0.132     -0.061
```

```
## CONT.SEGvlr -0.255  0.027 -0.004  0.150      0.106      0.045
## CONT.SEGvbr -0.316 -0.001  0.084 -0.091      0.101     -0.032
##             CONT.PRECvb CONT.SEGl CONT.SEG. CONT.SEGvl
## AMOSTRASP20
## F1.SEG.NORM
## CONT.PRECl
## CONT.PRECp.
## CONT.PRECvl
## CONT.PRECvb
## CONT.SEGlbl  0.286
## CONT.SEGpl. -0.018       0.545
## CONT.SEGvlr  0.199       0.621     0.582
## CONT.SEGvbr  0.229       0.581     0.590     0.660
```

O principal resultado a checar num modelo de efeitos mistos é se os mesmos efeitos fixos continuam a ser correlacionados após a inclusão dos efeitos aleatórios. Ao ver os coeficientes de `mod1.lmer`, percebemos que a variável CONT.PREC deixa de ser significativamente correlacionada com F1.NORM. Isso é sinal de que o efeito anteriormente observado se deve a alguns falantes ou a alguns itens lexicais específicos (mais provavelmente este último caso, já que se trata de uma variável linguística) e, tendo-os em conta como efeito aleatório, pode-se chegar à conclusão de que CONT.PREC não tem um efeito verdadeiro sobre a variável resposta.

Podemos também aplicar a função `step()`, na direção "de trás para frente", para verificar se a variável CONT.PREC é excluída desse novo modelo (as direções "forward" e "both" não funcionarão aqui, já que a função `lmer()` não permite criar um modelo sem efeitos aleatórios). Digite então `m.bw.lmer <- step(mod1.lmer, direction = "backward")`.

```
m.bw.lmer <- step(mod1.lmer, direction = "backward")
```

Veja o resultado de `m.bw.lmer`.

```
m.bw.lmer

## Backward reduced random-effect table:
##
##                    Eliminated npar  logLik    AIC    LRT Df
## <none>                             14 -3083.0 6194.0
## (1 | PARTICIPANTE)          1   13 -3084.0 6194.1 2.0447  1
## (1 | PALAVRA)               0   12 -3085.8 6195.6 3.4823  1
##                    Pr(>Chisq)
## <none>
## (1 | PARTICIPANTE)    0.15273
## (1 | PALAVRA)         0.06203 .
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Backward reduced fixed-effect table:
## Degrees of freedom method: Satterthwaite
##
##             Eliminated  Sum Sq Mean Sq NumDF  DenDF F value    Pr(>
F)
## CONT.PREC            1  5456.1  1364.0     4 180.78  2.3711   0.054
14
## AMOSTRA              0  9896.0  9896.0     1 649.74 17.2840 3.650e-
05
## F1.SEG.NORM          0 16752.3 16752.3     1 504.77 29.2589 9.794e-
08
## CONT.SEG             0  6040.5  1510.1     4 175.21  2.6375   0.035
63
##
## CONT.PREC   .
## AMOSTRA     ***
## F1.SEG.NORM ***
## CONT.SEG    *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Model found:
## F1.NORM ~ AMOSTRA + F1.SEG.NORM + CONT.SEG + (1 | PALAVRA)
```

As primeiras linhas do resultado mostram as variáveis eliminadas e mantidas na coluna Eliminated. Dos efeitos fixos, CONT.PREC de fato se mostra não correlacionada com F1.NORM, de modo que podemos retirá-la do modelo final. Dos efeitos aleatórios, PARTICIPANTE foi eliminado; isso se deve ao fato de que a normalização de Lobanov tem justamente o objetivo de minimizar diferenças que se devem a características individuais. Em qualquer outra análise que não envolva normalização, a inclusão da variável falante/participante é fundamental.

Cabe comentar, por fim, como apresentar os resultados de uma análise multivariada de regressão linear. Na Tabela 13.1 se apresentam duas tabelas de Gries (2019, p. 229) como exemplos: é importante reportar todos os valores da tabela de coeficientes, bem como as medidas estatísticas que aparecem ao pé do resultado de `summary()`.

Tabela 13.1: Exemplo de tabela de resultados (1).

	Soma de Quadrados	Estimativa	Erro Padrão	t	p
Intercepto	23,61	2,75	1,52	1,8	0,08
GERMAN	2931,69	1,75	0,09	20,1	<0,001
CLASS	3010,30	-8,72	0,43	-20,37	<0,001
Resíduo	558,68				
R^2	R^2 Múltiplo = 0,974			$F_{2,77} = 1416$	$p < 0,001$
	R^2 Ajustado = 0,973				

Fonte: Gries (2019, p.229).

Eis outro exemplo na Tabela 13.2: este é do artigo de Walker et al (2014), publicado na revista *Language Variation and Change*. Ao reportar modelos de efeitos mistos, é relevante indicar os efeitos aleatórios incluídos no modelo.

Tabela 13.2: Exemplo de tabela de resultados (2).

	Estimate	*SE*	*t* value	*p* value
Intercept	-.09334	.1316	-.709	.478
Speaker = Puerto Rican	.16994	.16247	1.046	.296
Variant = [s]	.32958	.05556	5.932	<.001
Participant = Puerto Rican	-.20599	.06993	-2.946	.003
Speaker = Puerto Rican: Variant = [s]	-.23736	.07228	-3.284	.001

Note: random effects = (1 + speaker nationality * variant | participant) + (1 + variant | speaker)

Fonte: Walker et al (2014, p.179).

E na Tabela 13.3 está um modelo elaborado por Elisa Battisti – um modelo que eu mesma passei a adotar em meus trabalhos por ser particularmente informativo e elegante. Para variáveis previsoras fatoriais (todas as variáveis, neste caso), além das estatísticas geradas pelo modelo, inclui-se uma coluna com o número de dados do valor de aplicação e do total para a respectiva variante da linha. Ao pé da tabela, apresenta-se o modelo completo que foi testado.

Tabela 13.3: Exempo de tabela de resultados (3).

Análise de regressão de efeitos mistos de haplologia com Renda Domiciliar (N = 864)
Intercepto = -1.087

Variável	Estimativa	Erro padrão	Valor-*z*	*p*	Apl./N
Zona					
Centro (v. referência)					65/264 (25%)
Leste	-0,093	0,488	-0,191	0,848	25/163 (15%)
Norte	2,134	0,900	2,370	0,018 *	39/200 (19%)
Sul	1,006	0,358	2,811	0,005 **	106/247 (43%)
Renda Domiciliar					
a (v. referência)					110/347 (31%)
b1	-0,024	0,352	-0,071	0,943	83/251 (33%)
b2	-3,680	0,998	-3,681	<0,001 ***	6/70 (8%)
c1	-2,281	0,771	-2,975	0,003 **	36/196 (18%)
Sílaba um					
CCV (v. referência)					20/43 (46%)
CV	-0,344	0,632	-0,545	0,586	215/821 (26%)
Sílaba dois					
CV (v. referência)					209/728 (29%)
CVC	-0,654	0,326	-2,006	0,045 *	26/136 (19%)
Tonicidade					
Átona-átona (v. referência)					183/617 (30%)
Átona-tônica	-0,297	0,274	-1,087	0,277	52/247 (21%)

Modelo 2. HAPLOLOGIA ~ ZONA + RENDA.DOMIC + SILABAUM + SILABADOIS2 + TONICIDADE + (1|PALAVDIR) + (1|PALAVESQ) + (1|INDIVIDUO)

Fonte: Battisti (c.p.)

Cabe ainda ressaltar que o melhor modo de reportar os seus resultados deve seguir modelos de sua área de pesquisa específica. O melhor jeito de saber isso é *lendo artigos de periódicos renomados da sua área.*

Nesta e na última lição, vimos como realizar e ler os resultados de modelos lineares, as funções `step()` e `drop1()` que auxiliam o pesquisador a escolher as variáveis a serem incluídas ou retiradas do modelo, e uma lista de pressupostos que devem ser checados a fim de avaliar quão confiáveis são os resultados. Em especial, enfatiza-se a importância de se realizar uma análise de efeitos mistos, com a inclusão de efeitos

aleatórios, para que se possa confirmar se os efeitos fixos de fato têm influência na variável resposta. O resultado final a ser reportado deve ser, idealmente, aquele do modelo de efeitos mistos.

Deixei disponível no Anexo B um *script* com uma sugestão de roteiro de análise para um conjunto de dados de uma variável numérica, como é o caso das vogais pretônicas. Trata-se efetivamente apenas de uma sugestão – cabe a você decidir as análises que efetivamente acabará fazendo. A ideia desse roteiro é juntar o conteúdo que foi aqui apresentado ao longo de diversas lições, mas que, na prática, serão realizados em sequência por você.

Para saber mais

Recomendo fortemente a leitura dos capítulos 7 e 8 de Levshina (2015) para se aprofundar nos preceitos da análise de regressão linear. Esses capítulos apresentam os passos e os pressupostos de modelos lineares de modo bastante detalhado.

Exercícios

Nesta lista de exercícios, você vai desenvolver uma análise semelhante à que fizemos na Lição 13, mas agora sobre a vogal /o/ pretônica. Primeiro, carregue os dados da planilha Pretonicas.csv e crie um subconjunto de dados da vogal /o/ pretônica. Recodifique as variáveis CONT.PREC e CONT.SEG com os mesmos critérios empregados na recodificação dessas variáveis para a vogal /e/ pretônica.

Imagine que você tenha levantado a hipótese de que a altura da vogal /o/ pretônica (F1.NORM) depende das variáveis AMOSTRA, SEXO, CONT.PREC, CONT.SEG, F1.SEG.NORM, ESTR.SIL.PRET. A última variável ainda não foi apresentada: ESTR.SIL.PRET codifica a estrutura da sílaba em que se encontra a vogal pretônica (CV; CCV; CVr; CVs – em que C = consoante, V = vogal, r = /r/ em coda, s = /s/ em coda).

1. Entre quais pares de variáveis há colinearidade? Considere: (i) AMOSTRA; (ii) SEXO; (iii) CONT.PREC; (iv) CONT.SEG; (v) F1.SEG.NORM; (vi) ESTR.SIL.PRET.

a. entre (iii)-(vi) e entre (iv)-(vi)
b. entre (i)-(v) e entre (v)-(vi)
c. entre (iii)-(iv) e entre (iii)-(vi)
d. entre (iv)-(v) e entre (ii)-(v)

2. Entre qual par de variáveis há interação? Considere: (i) AMOSTRA; (ii) SEXO; (iii) CONT.PREC; (iv) CONT.SEG; (v) F1.SEG.NORM; (vi) ESTR.SIL.PRET.
    a. entre (iii) e (iv)
    b. entre (i) e (ii)
    c. entre (i) e (v)
    d. entre (ii) e (vi)
    e. entre (iv) e (v)
3. A partir dos dados da vogal /o/ pretônica, crie um modelo com F1.NORM como variável resposta e com todas as variáveis previsoras acima, exceto aquela que é colinear a duas outras variáveis. Inclua a interação identificada na questão 2. Quanto da variação em F1.NORM é explicada por esse modelo?
    a. 8,6%
    b. 22,9%
    c. 24,7%
4. Qual variável não apresenta correlação significativa com F1.NORM?
    a. AMOSTRA
    b. CONT.PREC
    c. CONT.SEG
    d. ESTR.SIL.PRET
    e. F1.SEG.NORM
    f. SEXO
    g. nenhuma
    h. todas

5. Calcule a estimativa de F1.NORM na fala de paraibanas quando a vogal /o/ pretônica é precedida de consoante velar e seguida de consoante palatal-sibilante (p.ex., no item lexical "gostaria").
6. Use as funções `step()` e `drop1()` para checar se as variáveis do modelo criado acima devem ser mantidas. Os testes com `step()` e `drop1()` concordam quanto às variáveis que devem ser mantidas? Justifique sua resposta.
7. Os testes "forward", "backward" e "both" de `step()` concordam quanto às variáveis que devem ser mantidas? Justifique sua resposta.
8. A função `crPlot()`, do pacote car, infelizmente não se aplica a modelos com interação. Crie um novo modelo com a inclusão das mesmas variáveis do último teste, mas sem a inclusão de interação entre variáveis. Escolha se você vai manter a variável excluída por `drop1()` ou não. Em seguida, aplique a função `crPlot()` para testar se há linearidade entre a variável resposta e variável previsora numérica do modelo.
9. Em qual intervalo da variável previsora há menor concordância entre os valores previstos e os valores observados?
    a. entre 300 Hz e 350 Hz
    b. entre 350 Hz e 400 Hz
    c. entre 400 hz e 450 Hz
    d. entre 450 Hz e 500 Hz
10. Crie um modelo de efeitos mistos, com as variáveis aleatórias PARTICIPANTE e PALAVRA, e com a inclusão das mesmas variáveis do penúltimo teste – ou seja, com a interação entre as duas variáveis identificada na questão 2. Não se assuste pois o R mostrará alguns avisos de que certas combinações entre fatores de variáveis não existem.
11. Qual variável a seguir não apresenta correlação significativa com F1.NORM no modelo de efeitos mistos?

a. AMOSTRA
b. CONT.PREC
c. CONT.SEG
d. F1.SEG.NORM
e. interação CONT.PREC:CONT.SEG
f. todas as variáveis são significativamente correlacionadas

12. A partir do modelo de efeitos mistos, calcule a estimativa da altura da vogal /o/ pretônica (F1.NORM) quando o F1 da vogal da sílaba seguinte é 450 Hz.

Lição 14: Regressão Logística Parte 1

N.B.: Rode as linhas de comando a seguir antes de iniciar esta lição. Defina como diretório de trabalho aquele que contém o arquivo DadosRT.csv.

```
# Definir diretório de trabalho

#setwd()

# Importar dados da planilha

dados <- read_csv("DadosRT.csv",
                  col_types = cols(.default = col_factor(),
                                   VD = col_factor(levels = c("tepe",
"retroflexo")),
                                   FAIXA.ETARIA = col_factor(levels =
c("1a", "2a", "3a")),
                                   ESCOLARIDADE = col_factor(levels =
c("fundamental", "medio", "superior")),
                                   REGIAO = col_factor(levels = c("cen
tral", "periferica")),
                                   CONT.FON.PREC = col_factor(levels =
c("i", "e", "3", "a", "0", "o", "u")),
                                   TONICIDADE = col_factor(levels = c(
"atona", "tonica")),
                                   POSICAO.R = col_factor(levels = c("
final", "medial")),
                                   CLASSE.MORFOLOGICA = col_factor(lev
els = c("adjetivo", "adverbio", "conj.prep", "morf.inf", "substantivo"
, "verbo")),
                                   IDADE = col_integer(),
                                   INDICE.SOCIO = col_double(),
                                   FREQUENCIA = col_double()
                                   )
                  )

dados$CONT.FON.SEG <- fct_collapse(dados$CONT.FON.SEG,
                                   pausa = "#",
                                   coronal = c("t", "d", "s", "z", "x"
, "j", "ts", "dz", "l", "n"),
                                   labial = c("p", "b", "f", "v", "m")
,
                                   dorsal = c("k", "g", "h")
                                   )

dados$CONT.FON.SEG <- fct_relevel(dados$CONT.FON.SEG, "pausa", "corona
l", "dorsal", "labial")

###Funções úteis (Gries 2019)
```

```
logit <- function(x) {
  log(x/(1-x))
  }

ilogit <- function(x) {
  1/(1+exp(-x))
  }
```

Esta e a próxima lição são dedicadas a análises de regressão logística, que se aplicam a variáveis dependentes/resposta binárias. A regressão logística permite a inclusão de múltiplas variáveis previsoras, assim como a análise de regressão linear. Cabe lembrar que, antes de chegar à análise multivariada de regressão logística, o pesquisador idealmente já terá feito gráficos e análises exploratórias (como o qui-quadrado) a fim de saber como se distribuem seus dados. O interesse nas análises de regressão logística é verificar o efeito simultâneo de múltiplas variáveis previsoras, a fim de chegar a um modelo para descrever, explicar e prever o comportamento da variável resposta.

A regressão logística também gera um coeficiente linear (Intercept) e coeficientes angulares para cada variável/termo previsor do modelo, e avalia se a estimativa difere significativamente de zero. Contudo, enquanto a regressão linear gera coeficientes na mesma unidade que da variável resposta – nos exemplos das Lições 12 e 13, medidas de F1.NORM em Hz –, a regressão logística gera coeficientes em *logodds* (também chamado de log-odds-ratio), sobre o qual falaremos mais adiante. Os valores de logodds entram na função y = a + bx1, cx2, dx3... para permitir a estimativa de probabilidade de ocorrência dos níveis da variável resposta.

Carregue inicialmente os pacotes necessários para esta lição: tidyverse, car, effects e rms.

```
library(tidyverse)
library(car)
library(effects)
library(rms)
```

Deixei disponível para você o conjunto de dados da pronúncia do /r/ em coda silábica como tepe ou como retroflexo na fala de paulistanos, num dataframe chamado `dados`. Aplique a função `str()` para se (re)familiarizar com ele.

```
str(dados)

## spec_tbl_df [9,226 × 20] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ VD                : Factor w/ 2 levels “tepe”,”retroflexo”: 2 2
2 1 2 2 2 2 1 2 ...
##  $ PARTICIPANTE      : Factor w/ 118 levels “IvanaB”,”HeloisaS”,..:
1 1 1 1 1 1 1 1 1 1 ...
##  $ SEXO.GENERO       : Factor w/ 2 levels “feminino”,”masculino”: 1
1 1 1 1 1 1 1 1 1 ...
##  $ IDADE             : int [1:9226] 30 30 30 30 30 30 30 30 30 30 .
..
##  $ FAIXA.ETARIA      : Factor w/ 3 levels “1a”,”2a”,”3a”: 1 1 1 1 1
1 1 1 1 1 ...
##  $ ESCOLARIDADE      : Factor w/ 3 levels “fundamental”,..: 1 1 1 1
1 1 1 1 1 1 ...
##  $ REGIAO            : Factor w/ 2 levels “central”,”periferica”: 2
2 2 2 2 2 2 2 2 ...
##  $ INDICE.SOCIO      : num [1:9226] 2 2 2 2 2 2 2 2 2 2 ...
##  $ ORIGEM.PAIS       : Factor w/ 5 levels “mista”,”SPcapital”,..: 1
1 1 1 1 1 1 1 1 ...
##  $ CONT.FON.PREC     : Factor w/ 7 levels “i”,”e”,”3”,”a”,..: 4 6 2
2 4 4 5 4 5 3 ...
##  $ CONT.FON.SEG      : Factor w/ 4 levels “pausa”,”coronal”,..: 2 2
3 4 3 2 2 2 3 2 ...
##  $ TONICIDADE        : Factor w/ 2 levels “atona”,”tonica”: 2 1 1 1
2 2 2 2 1 2 ...
##  $ POSICAO.R         : Factor w/ 2 levels “final”,”medial”: 2 2 2 2
1 1 2 2 2 2 ...
##  $ CLASSE.MORFOLOGICA: Factor w/ 6 levels “adjetivo”,”adverbio”,..:
5 5 5 5 5 5 5 5 3 5 ...
##  $ FREQUENCIA        : num [1:9226] 1.34 0.16 0.22 0.44 1.94 1.94 0
.35 0.03 5.98 0.16 ...
##  $ ESTILO            : Factor w/ 4 levels “conversacao”,..: 1 1 1 1
1 1 1 1 1 1 ...
##  $ ITEM.LEXICAL      : Factor w/ 1151 levels “parte”,”jornal”,..: 1
2 3 4 5 5 6 7 8 9 ...
##  $ cont.precedente   : Factor w/ 6836 levels “do CEU é daquela”,..:
1 2 3 4 5 6 7 8 9 10 ...
##  $ ocorrencia        : Factor w/ 1760 levels “parte <R>“,”jornal <R
>“,..: 1 2 3 4 5 5 6 7 8 9 ...
##  $ cont.seguinte     : Factor w/ 6813 levels “que as perua(s) ia”,.
.: 1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, “spec”)=
##   .. cols(
##   ..   .default = col_factor(),
##   ..   VD = col_factor(levels = c(“tepe”, “retroflexo”), ordered =
FALSE, include_na = FALSE),
##   ..   PARTICIPANTE = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   SEXO.GENERO = col_factor(levels = NULL, ordered = FALSE, inc
lude_na = FALSE),
##   ..   IDADE = col_integer(),
##   ..   FAIXA.ETARIA = col_factor(levels = c(“1a”, “2a”, “3a”), orde
red = FALSE, include_na = FALSE),
```

```
##   ..   ESCOLARIDADE = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   REGIAO = col_factor(levels = c("central", "periferica"), ord
ered = FALSE, include_na = FALSE),
##   ..   INDICE.SOCIO = col_double(),
##   ..   ORIGEM.PAIS = col_factor(levels = NULL, ordered = FALSE, inc
lude_na = FALSE),
##   ..   CONT.FON.PREC = col_factor(levels = c("i", "e", "3", "a", "0
", "o", "u"), ordered = FALSE, include_na = FALSE),
##   ..   CONT.FON.SEG = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   TONICIDADE = col_factor(levels = c("atona", "tonica"), order
ed = FALSE, include_na = FALSE),
##   ..   POSICAO.R = col_factor(levels = c("final", "medial"), ordere
d = FALSE, include_na = FALSE),
##   ..   CLASSE.MORFOLOGICA = col_factor(levels = c("adjetivo", "adve
rbio", "conj.prep", "morf.inf", "substantivo",
##   ..     "verbo"), ordered = FALSE, include_na = FALSE),
##   ..   FREQUENCIA = col_double(),
##   ..   ESTILO = col_factor(levels = NULL, ordered = FALSE, include_
na = FALSE),
##   ..   ITEM.LEXICAL = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   cont.precedente = col_factor(levels = NULL, ordered = FALSE,
include_na = FALSE),
##   ..   ocorrencia = col_factor(levels = NULL, ordered = FALSE, incl
ude_na = FALSE),
##   ..   cont.seguinte = col_factor(levels = NULL, ordered = FALSE, i
nclude_na = FALSE)
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Aplique também a função `View()` para visualizar a planilha. Em especial, veja as variáveis SEXO.GENERO, FAIXA.ETARIA, INDICE.SOCIO e REGIAO, com que trabalharemos nesta lição.

```
View(dados)
```

N.B.: Resultado aqui omitido.

A variável SEXO.GENERO indica se o dado veio de um falante do sexo feminino ou masculino. FAIXA.ETARIA está dividida em três níveis – 1a: de 20 a 34 anos; 2a: de 35 a 59 anos; e 3a: 60 anos ou mais. REGIAO indica a região de residência atual do falante, central ou periférica. Por fim, INDICE.SOCIO é um índice contínuo que vai de 1 a 5, que leva em conta a escolaridade, ocupação e renda do falante, bem como escolaridade e ocupação de seus pais (ver Oushiro, 2015, cap.3 para detalhes); quanto maior o índice, maior foi a pontuação média do falante para os critérios acima.

De modo semelhante ao que fizemos na análise de regressão linear, vamos começar com modelos simples, com o intuito de treinar a leitura dos resultados. Na próxima lição, veremos como criar e avaliar modelos mais complexos, que são aqueles que você efetivamente vai querer reportar nas publicações.

A função para criar um modelo de regressão logística é `glm()` – do inglês, "generalized linear model". Trata-se efetivamente de uma generalização do modelo de regressão linear para variáveis não numéricas. Você verá que muito da implementação da regressão logística é paralela à regressão linear. Mas é importante já mencionar uma diferença importante: as estimativas da regressão logística são fornecidas em relação ao *segundo* nível da variável dependente. Aplique a função `levels()` ao vetor `dados$VD` para ver os níveis da variável dependente.

```
levels(dados$VD)
## [1] "tepe"       "retroflexo"
```

O R organizou os níveis na ordem tepe e retroflexo a partir da especificação de levels no momento da importação dos dados (ver *script*). Isso significa que os resultados dos modelos deverão ser lidos em termos do que aumenta ou diminui a probabilidade de ocorrência do *retroflexo*. Essa variante foi escolhida para leitura dos resultados pois temos mais interesse em vê-los da perspectiva da variante não prototípica da comunidade paulistana. Será mais interessante descobrir quais fatores mais favorecem o emprego de retroflexo do que do tepe.

Podemos então seguir para a regressão logística. Assim como a função `lm()`, o primeiro argumento de `glm()` é uma fórmula no formato VR ~ VP..., e o segundo argumento é o conjunto de dados. Aqui, entretanto, há mais um argumento: family = binomial, que indica que a variável resposta é binária. Crie então um primeiro modelo chamado `mod1`, que testa se há correlação entre a variação na pronúncia de /r/ em coda (VD) e o SEXO.GENERO do falante. Digite `mod1 <- glm(VD ~ SEXO.GENERO, data = dados, family = binomial)`.

```
mod1 <- glm(VD ~ SEXO.GENERO, data = dados, family = binomial)
```

Veja o resultado com a aplicação de `summary()` a `mod1`.

```
summary(mod1)
```

```
##
## Call:
## glm(formula = VD ~ SEXO.GENERO, family = binomial, data = dados)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8899  -0.8899  -0.7375   1.4952   1.6941
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -1.16304    0.03475 -33.470   <2e-16 ***
## SEXO.GENEROmasculino  0.44111    0.04672   9.442   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10993  on 9225  degrees of freedom
## Residual deviance: 10903  on 9224  degrees of freedom
## AIC: 10907
##
## Number of Fisher Scoring iterations: 4
```

Você já está bem familiarizado com o *output* de um teste estatístico no R. Primeiro, ele informa a fórmula empregada e os resíduos. Em seguida, mostra a tabela de coeficientes, com o erro padrão e um valor de significância para cada um. Por fim, dá informações gerais sobre o modelo como um todo. Mas há aqui algumas pequenas diferenças.

A principal delas está nas estimativas dos coeficientes, que são dadas em logodds. Quando fizemos um modelo de regressão linear, era fácil interpretar as estimativas, pois elas eram dadas na mesma unidade que a variável resposta. Mas, aqui, o que significam -1,16304 logodds para o coeficiente linear e 0,44111 logodds para o coeficiente angular? Na sequência, vamos ver como esse valor é calculado para você mais bem entendê-lo.

Visualize as frequências dos dados de SEXO.GENERO pela VD, substituindo no *script* os termos necessários.

```
dados %>%
  count(SEXO.GENERO, VD)
```

```
## # A tibble: 4 × 3
##   SEXO.GENERO VD            n
##   <fct>       <fct>     <int>
## 1 feminino    tepe       3478
```

```
## 2 feminino    retroflexo  1087
## 3 masculino   tepe        3137
## 4 masculino   retroflexo  1524
```

As chances (= odds) de algo ocorrer se dão pela divisão simples entre o número de resultados favoráveis e o número de resultados desfavoráveis. Embora relacionadas com a probabilidade, as chances são algo diferente. As chances normalmente são expressas por uma razão, do tipo 3:1. Assim, as chances de ocorrer *tepe* na fala de mulheres são 3478 / 1087, de acordo com a tabela acima. Guarde esse resultado num objeto chamado `odds_F`.

```
odds_F <- 3478 / 1087
```

Faça o mesmo cálculo para os homens, e guarde-o num objeto chamado `odds_M`.

```
odds_M <- 3137 / 1524
```

Faça agora o cálculo do odds-ratio, a razão (= divisão) entre as chances de ocorrer tepe na fala das mulheres (odds_F) e na fala dos homens (odds_M). Guarde-o num objeto chamado `odds_ratio`.

```
odds_ratio <- odds_F / odds_M
```

Agora aplique a função `log()` a `odds_ratio`.

```
log(odds_ratio)
```

```
## [1] 0.4411073
```

Veja que o resultado do log de odds-ratio é justamente a estimativa calculada para os homens na tabela de coeficientes de nosso modelo – então agora você sabe de onde saiu essa estimativa. Por que usar o log do odds-ratio, e não simplesmente as chances (= odds) ou a probabilidade (= proporção de ocorrências da variante de interesse em relação ao total)?

A Figura 14.1 (Gries, 2019, p. 265) mostra a relação entre essas três medidas de probabilidade. Odds é uma escala de vai de zero até +∞. Isso não é difícil de entender. As chances de algo ocorrer (=resultado favorável – Fav) são maiores, menores ou iguais às chances de não ocorrer (=resultado desfavorável – Des). Se o resultado favorável é mais frequente do que o resultado desfavorável, a divisão Fav/Des dará um número maior do que 1, e não tem limite máximo. Se o resultado favorável é menos frequente do que o

resultado desfavorável, a divisão Fav/Des dará um número menor do que 1, mas nunca negativo. O ponto neutro da escala é 1, pois ele equivale ao cenário em que Fav = Des, portanto Fav/Des = 1.

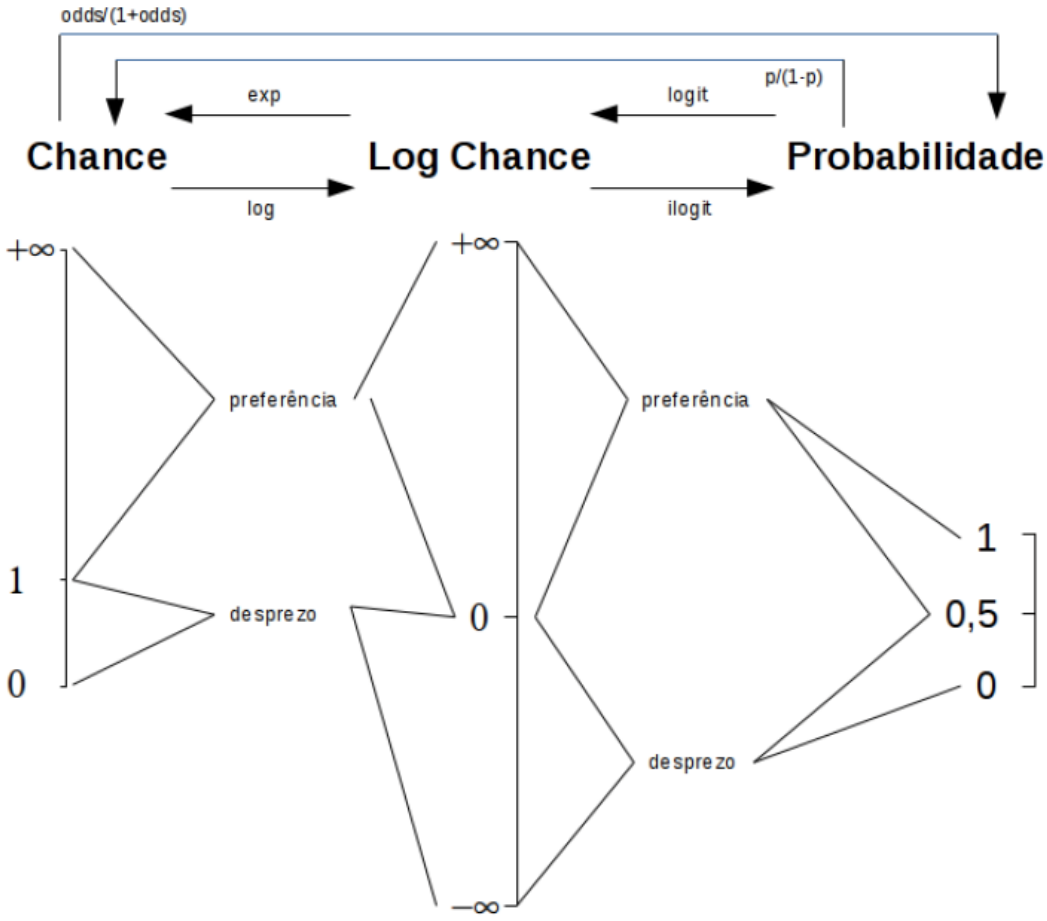

Figura 14.1: Relações entre as medidas de odds, logodds e probabilidade. Fonte: Gries (2019, p. 265).

Por outro lado, a probabilidade é uma medida mais conhecida. Ela é uma escala de 0 a 1, em que 0 representa uma chance nula de algo ocorrer (0%) e 1 representa certeza de que vai ocorrer (100%). De modo simples, ela é calculada pelo número de resultados favoráveis pelo total de observações (Fav/T). O ponto neutro aí é 50%; um número abaixo disso indica maior probabilidade de que o evento não vai ocorrer e um número acima disso indica maior probabilidade de que o evento vai ocorrer.

Por fim, a escala de logodds vai de menos infinito a mais infinito, com ponto neutro em zero. Em relação a odds, ela tem a vantagem de ser uma escala simétrica, com

um mesmo intervalo entre o ponto neutro e suas extremidades. A operação log tem justamente o papel de transformar valores entre 0 e 1 em um valor negativo (experimente depois no Console do R!). Desse modo, a interpretação de valores é muito mais intuitiva do que numa escala assimétrica, pois os intervalos em que há favorecimento ou desfavorecimento de um evento são diretamente comparáveis.

Em relação à escala de probabilidade, ter o ponto neutro em zero – em vez de 0,5 – também traz vantagens. Vimos, no modelo de regressão linear, que o resultado em termos de *diferença* em relação ao intercept permite avaliar mais prontamente se ela é zero ou não. Um logodds de zero (ou próximo a ele) indica prontamente se há diferenças significativas; valores positivos indicam tendência a favorecimento (em relação a outro nível da mesma variável previsora); e valores negativos indicam tendência a desfavorecimento (em relação a outro nível da mesma variável previsora). No resultado de nosso primeiro modelo, portanto, um logodds de 0,44 para homens indica que, *em relação às mulheres*, os homens favorecem o retroflexo.

Coloquei a comparação entre homens e mulheres acima em destaque porque uma estimativa de logodds positiva não significa que os homens usam o retroflexo mais frequentemente do que o tepe. Essa é uma diferença relativa. Lembre-se que para chegar à estimativa, é necessário somar o valor do intercept ao valor do coeficiente angular. Os homens paulistanos, portanto, têm -1.16304 + 0.44111 = -0.7219327 logodds de emprego de retroflexo. A Figura 14.1 mostra qual operação se aplica para transformar uma medida em outra – exp, log, logit ou ilogit. Aplique a função `ilogit()` a -0.7219327 para transformar a medida de logodds em probabilidade.

```
ilogit(-0.7219327)
```

```
## [1] 0.3269675
```

O cálculo acima indica que a probabilidade de os homens empregarem retroflexo é 32,7%. Aplique também a função `ilogit()` ao valor de -1.16304, a estimativa em logodds para as mulheres.

```
ilogit(-1.16304)
```

```
## [1] 0.2381153
```

O cálculo para as mulheres indica uma probabilidade de 23,8% de elas empregarem o retroflexo. Para comparar, faça agora uma tabela de proporções, substituindo os termos df e VAR na estrutura pré-montada do *script* – reveja a Lição 4, se necessário.

```
dados %>%
  count(SEXO.GENERO, VD) %>%
  group_by(SEXO.GENERO) %>%
  mutate(prop = prop.table(n))
```

```
## # A tibble: 4 × 4
## # Groups:   SEXO.GENERO [2]
##   SEXO.GENERO VD             n  prop
##   <fct>       <fct>      <int> <dbl>
## 1 feminino    tepe        3478 0.762
## 2 feminino    retroflexo  1087 0.238
## 3 masculino   tepe        3137 0.673
## 4 masculino   retroflexo  1524 0.327
```

O cálculo de probabilidade, nesse modelo univariado, corresponde exatamente ao valor de logodds, pois não há outras variáveis previsoras que possam ajustar a estimativa de logodds. Legal, não?

Não é necessário memorizar toda essa demonstração, nem a realizar todas as vezes que criar um modelo de regressão logística. Ela serve aqui para que você perceba que as medidas de logodds se relacionam diretamente a outras medidas estatísticas mais conhecidas (chances e probabilidades), mas têm a vantagem de ser uma escala centrada em zero, o que facilita a interpretação dos resultados.

As medidas em logodds podem ser estranhas no começo, pela falta de contato que se tem com elas. No entanto, aos poucos você vai se acostumar a ver uma medida em logodds e saber se se trata de um efeito forte ou fraco. A Tabela 14.1 (adaptada de Levshina, 2015, p. 265) mostra as equivalências entre probabilidades, *odds* e *logodds*.

Tabela 14.1: Tabela de conversão de valores entre medidas de probabilidades, odds e logodds.

Probabilidade	*Odds*	Logit (*log odds*)
0,001	0,001	-6,91
0,01	0,01	-4,60
0,05	0,05	-2,94

0,10	0,11	-2,20
0,25	0,33	-1,10
0,50	1	0
0,75	3	1,10
0,90	9	2,20
0,95	19	2,94
0,99	99	4,60
0,999	999	6,91

Fonte: Levshina (2015, p. 265).

A essa altura, o resultado de `mod1` ficou lá atrás, mas ainda falta comentar as demais medidas estatísticas geradas no modelo. Digite então `summary(mod1)` no Console.

```
summary(mod1)

##
## Call:
## glm(formula = VD ~ SEXO.GENERO, family = binomial, data = dados)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8899  -0.8899  -0.7375   1.4952   1.6941
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -1.16304    0.03475 -33.470   <2e-16 ***
## SEXO.GENEROmasculino  0.44111    0.04672   9.442   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10993  on 9225  degrees of freedom
## Residual deviance: 10903  on 9224  degrees of freedom
## AIC: 10907
##
## Number of Fisher Scoring iterations: 4
```

De modo semelhante ao modelo linear, a regressão logística apresenta o erro padrão junto à estimativa. Se se faz a divisão entre Estimativa / Erro Padrão, chega-se ao valor da terceira coluna. Aqui, em vez de um valor-*t*, temos um valor-*z*, que é consultado numa tabela de distribuição normal padrão (como fizemos na tabela de qui-quadrado, na Lição 9). Dessa tabela se obtém o valor de significância.

Ao pé da tabela, diferentemente do modelo linear, o R não reporta diretamente medidas que permitem avaliar o poder explanatório do modelo, como R^2 e a estatística-F. No entanto, os valores de desvio nulo e desvio residual permitem fazer esse cálculo. O desvio nulo se refere a quanto há de variabilidade total nos dados, sem a inclusão de qualquer variável previsora. O desvio residual se refere a quanto há de variabilidade nos dados depois da inclusão da(s) variável(is) previsora(s). Portanto, a diferença entre o desvio residual e o desvio nulo é o quanto o nosso modelo é capaz de prever da variabililidade dos dados.

No Anexo C, deixei disponível um *script* que permite fazer o cálculo de significância do modelo como um todo "à mão", de posse dos dados de desvio nulo, desvio residual e graus de liberdade. Depois dessa lição, dedique um tempo para entendê-lo. Ele não será explicado aqui porque há uma maneira mais simples de se obter essa medida estatística, que veremos logo adiante.

O AIC (Akaike Information Criterion), como vimos na Lição 13, é uma medida que permite comparar modelos e é usada na função `step()` para selecionar ou excluir variáveis. E a última medida estatística reportada para o modelo logístico é o Fisher Scoring iterations. Ela se refere ao número de iterações do modelo até que os resultados convergiram. Aqui, o modelo convergiu após 4 tentativas. Quando este número é grande (digamos, acima de 20), é sinal de que foram incluídas mais variáveis previsoras do que é possível explicar com a quantidade de dados de que se dispõe. A solução, neste caso, é diminuir o número de variáveis incluídas no modelo.

De modo semelhante ao modelo linear, podemos plotar um gráfico de efeitos para mais bem visualizar os resultados numéricos. Para isso, vamos usar uma função do pacote effects, já carregado acima.

Para modelos logísticos, usamos a função `allEffects()` – mesmo se o modelo contém apenas uma variável previsora – e o argumento type = "response". Faça o gráfico de efeitos com `plot(allEffects(mod1), type = "response")` (Figura 14.2).

```
plot(allEffects(mod1), type = "response")
```

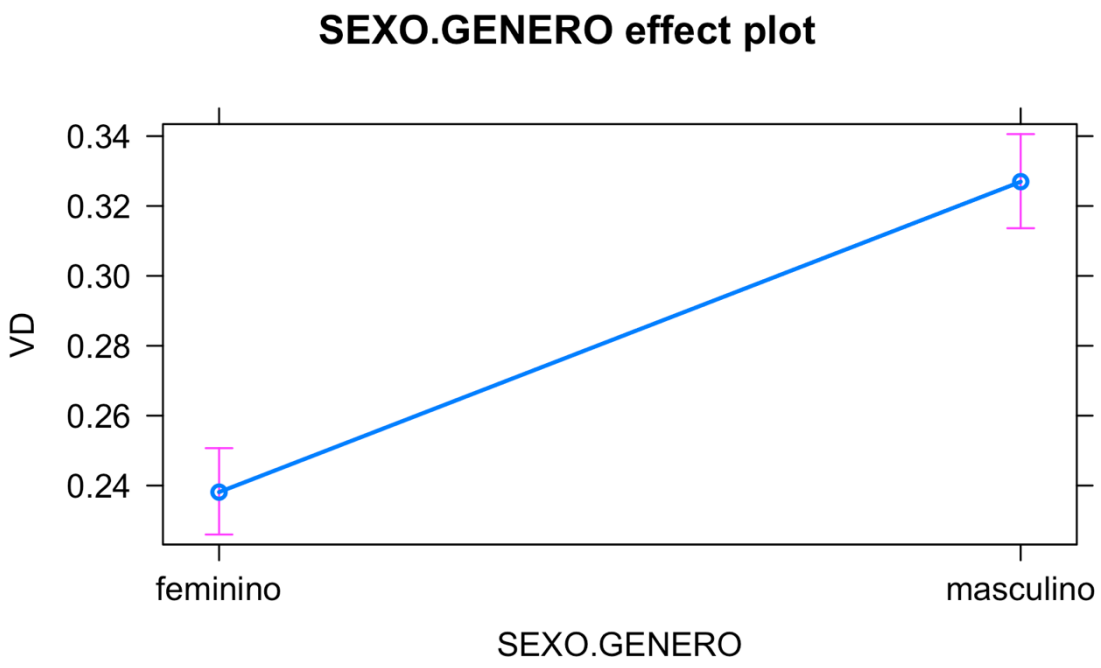

Figura 14.2: Gráfico de efeitos da variável Sexo para o uso de /r/ retroflexo. Fonte: própria.

O gráfico de efeitos mostra os resultados com as medidas de probabilidade. Vemos que a estimativa de emprego de retroflexo é cerca de 23% para as mulheres e 32% para os homens, e que os intervalos de confiança não se sobrepõem, o que permite inferir a diferença significativa entre os níveis.

Há outra função que faz regressões logísticas, chamada `lrm()` – "logistic regression model". Esta função pertence ao pacote rms, também carregado no início desta lição. A função `lrm()` toma como argumentos apenas a fórmula (VD ~ SEXO.GENERO) e o conjunto de dados (data = dados). Aplique-a então com `lrm(VD ~ SEXO.GENERO, data = dados)`.

```
lrm(VD ~ SEXO.GENERO, data = dados)

## Logistic Regression Model
##
##  lrm(formula = VD ~ SEXO.GENERO, data = dados)
##
##                  Model Likelihood    Discrimination    Rank Discrim
.
##                        Ratio Test           Indexes          Indexe
s
##Obs    9226    LR chi2      90.07    R2       0.014      C        0.55
```

```
5
##tepe  6615    d.f.             1    g        0.221      Dxy     0.10
9
##retr  2611    Pr(> chi2) <0.0001    gr       1.247      gamma   0.21
7
##max |deriv| 1e-11                   gp       0.044      tau-a   0.04
4
##                                    Brier    0.201
##
##                         Coef    S.E.   Wald Z Pr(>|Z|)
##  Intercept              -1.1630 0.0347 -33.47 <0.0001
##  SEXO.GENERO=masculino   0.4411 0.0467   9.44 <0.0001
##
```

Trata-se de outro método para o mesmo fim. Veja que os coeficientes gerados correspondem exatamente àqueles do modelo prévio com a função `glm()`. Mas, além de não ser necessário usar a função `summary()` para visualizar o resultado, a função `lrm()` apresenta algumas medidas estatísticas relevantes no topo do *output*. A primeira coluna fornece o total de observações e de cada variante da variável resposta.

A segunda coluna, Model Likelihood Ratio Test, informa se o modelo como um todo é significativo. O teste de Razão de Verossimilhança compara dois modelos (com e sem variáveis previsoras) e com isso gera um valor-*p* para o modelo – como é feito no *script* do Anexo C. As duas colunas à direita mostram várias medidas estatísticas de qualidade do ajuste, ou seja, de quão bem o modelo é capaz de explicar a variação encontrada nos dados (como o R^2). Para modelos de regressão logística, a estatística mais reportada é o índice de Concordância C, a primeira medida da quarta coluna. De Hosmer & Lemeshow 2000 (*apud* Levshina, 2015, p. 259), temos C = 0,5: pouco poder de discriminação de resultado; 0,7 < C < 0,8: poder aceitável de discriminação de resultado; 0,8 < C < 0,9: poder excelente de discriminação de resultado; e C > 0,9: poder notório de discriminação de resultado. Em nosso modelo, C = 0,55 é um sinal de que ele ainda pode ser melhorado.

Vamos seguir agora para um modelo com uma variável previsora nominal com mais de dois níveis. Com a função `glm()`, crie o modelo `mod2` que testa se há correlação entre VD e a FAIXA.ETARIA do falante. Não se esqueça de incluir o conjunto de dados e family = binomial.

```
mod2 <- glm(VD ~ FAIXA.ETARIA, data = dados, family = binomial)
```

E veja o resultado com `summary()`.

```
summary(mod2)

##
## Call:
## glm(formula = VD ~ FAIXA.ETARIA, family = binomial, data = dados)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9570  -0.7757  -0.7061   1.4151   1.7386
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -0.54330    0.03755 -14.468   <2e-16 ***
## FAIXA.ETARIA2a -0.50356    0.05484  -9.183   <2e-16 ***
## FAIXA.ETARIA3a -0.71874    0.05833 -12.322   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10993  on 9225  degrees of freedom
## Residual deviance: 10824  on 9223  degrees of freedom
## AIC: 10830
##
## Number of Fisher Scoring iterations: 4
```

Vamos direto aos coeficientes. O resultado mostra que, em relação à primeira faixa etária (nível de referência), os falantes tanto de segunda quanto de terceira faixa etária tendem a *des*favorecer o retroflexo – que se infere pelas estimativas negativas em logodds. Em outras palavras, os falantes mais jovens tendem a empregar mais retroflexos do que os mais velhos, o que pode ser indício de uma mudança em progresso na comunidade paulistana.

A partir da última linha de comando em que você usou `plot()` e `allEffects()`, mude o nome do modelo para `mod2` (Figura 14.3).

```
plot(allEffects(mod2), type = “response”)
```

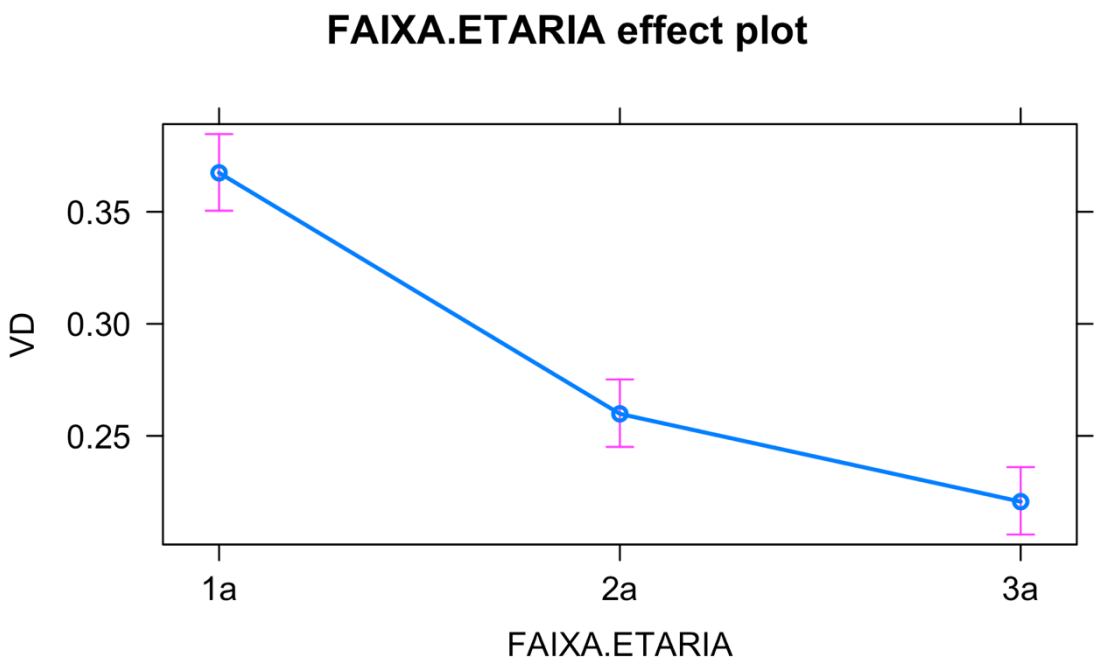

Figura 14.3: Gráfico de efeitos da variável Faixa Etária para o uso de /r/ retroflexo. Fonte: própria.

O gráfico de efeitos mostra claramente que, quanto mais velho o falante, menor a tendência a empregar retroflexo. Pela figura, também parece haver uma diferença significativa entre falantes de 2ª e 3ª faixas etárias.

Aplique a função `lrm()` ao mesmo modelo para obter outras medidas estatísticas relevantes.

```
lrm(VD ~ FAIXA.ETARIA, data = dados)

## Logistic Regression Model
##
##  lrm(formula = VD ~ FAIXA.ETARIA, data = dados)
##
##                       Model Likelihood     Discrimination    Rank Discri
m.
##                             Ratio Test            Indexes          Index
es
##Obs    9226      LR chi2     169.38    R2       0.026    C       0.5
80
##tepe  6615       d.f.             2    g        0.316    Dxy     0.1
59
##retr  2611       Pr(> chi2) <0.0001    gr       1.372    gamma   0.2
37
##max |deriv| 3e-11                      gp       0.065    tau-a   0.0
65
##                                       Brier    0.199
```

```
##
##                  Coef    S.E.   Wald Z Pr(>|Z|)
##  Intercept       -0.5433 0.0376 -14.47 <0.0001
##  FAIXA.ETARIA=2a -0.5036 0.0548  -9.18 <0.0001
##  FAIXA.ETARIA=3a -0.7187 0.0583 -12.32 <0.0001
##
```

Façamos agora um modelo com uma variável previsora numérica, INDICE.SOCIO. Guarde o resultado num objeto chamado `mod3`.

```
mod3 <- glm(VD ~ INDICE.SOCIO, data = dados, family = binomial)
```

Veja o resultado de `mod3` com `summary()`.

```
summary(mod3)

##
## Call:
## glm(formula = VD ~ INDICE.SOCIO, family = binomial, data = dados)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2758  -0.8498  -0.6874   1.2902   2.0168
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.53407    0.12684   12.10   <2e-16 ***
## INDICE.SOCIO -0.81609    0.04201  -19.43   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10993  on 9225  degrees of freedom
## Residual deviance: 10596  on 9224  degrees of freedom
## AIC: 10600
##
## Number of Fisher Scoring iterations: 4
```

Como se trata de uma variável numérica contínua, a estimativa em logodds representa em quanto aumenta ou diminui a probabilidade de emprego de retroflexo a cada unidade de INDICE.SOCIO (ou seja, quanto mais alto o nível socioeconômico do falante). O coeficiente negativo -0,81609 indica que quanto mais alto o índice socioeconômico, menor a probabilidade de que o falante empregue o retroflexo.

Visualize esse resultado por meio de um gráfico de efeitos. A partir da última linha de comando em que se usou `plot()` e `allEffects()`, mude o nome do modelo para `mod3` (Figura 14.4).

```
plot(allEffects(mod3), type = “response”)
```

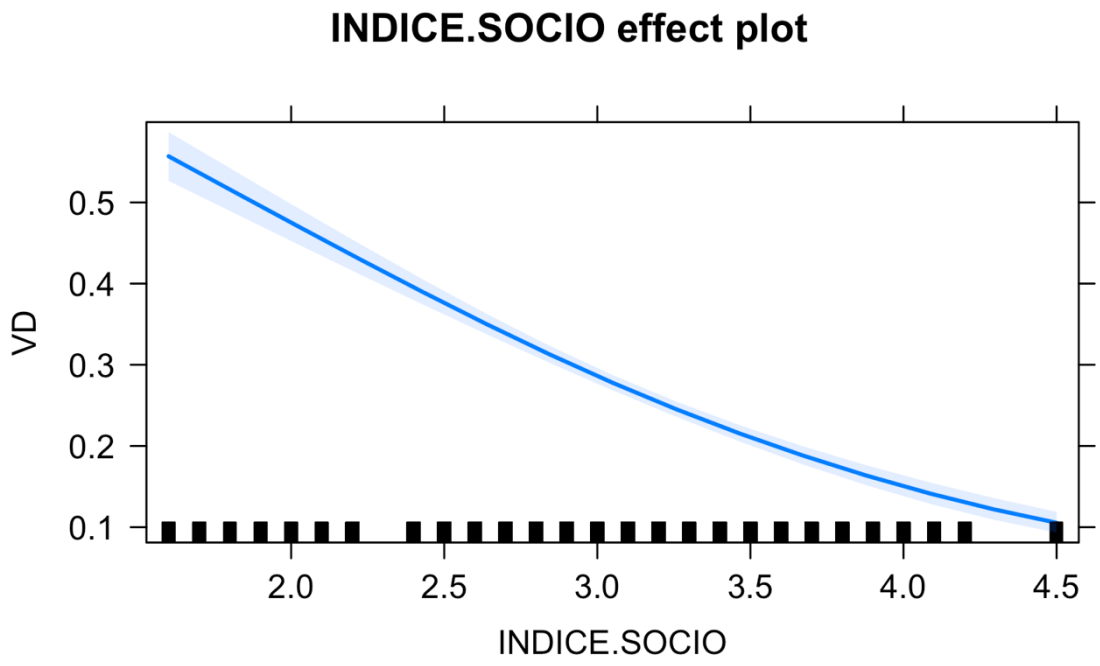

Figura 14.4: Gráfico de efeitos da variável Índice Socioeconômico para o uso de /r/ retroflexo. Fonte: própria.

Como já visto nos modelos lineares, a variável previsora numérica é representada pela linha de regressão. Vemos aí a drástica queda em probabilidade de emprego de retroflexo, de 50% para 10%, quanto mais alto o nível socioeconômico do falante.

Calcule a probabilidade, em logodds, de um falante com índice socioeconômico 4.2 empregar o retroflexo. Para isso, empregue os coeficientes gerados pelo modelo dentro da função y = 1.53407 + (-0.81609 * INDICE.SOCIO).

```
1.53407 + (-0.81609 * 4.2)
```

```
## [1] -1.893508
```

Transforme o valor em logodds em probabilidade. Para isso, aplique a função `ilogit()` à estimativa gerada acima.

```
ilogit(-1.893508)
```

```
## [1] 0.130845
```

Confira no gráfico de efeitos que a probabilidade calculada acima corresponde à medida do eixo y (VD) quando x = 4.2.

Agora obtenha as demais medidas estatísticas com uso da função `lrm()`.

```
lrm(VD ~ INDICE.SOCIO, data = dados)
```

```
## Logistic Regression Model
##
##  lrm(formula = VD ~ INDICE.SOCIO, data = dados)
##
##                    Model Likelihood    Discrimination    Rank Discri
m.
##                          Ratio Test           Indexes          Index
es
##Obs       9226    LR chi2     397.05    R2       0.061    C       0.
629
##tepe      6615    d.f.             1    g        0.535    Dxy     0.
259
##retr      2611    Pr(> chi2) <0.0001    gr       1.707    gamma   0.
273
##max |deriv| 3e-11                       gp       0.105    tau-a   0.
105
##                                        Brier    0.195
##
##               Coef    S.E.   Wald Z Pr(>|Z|)
##  Intercept     1.5341 0.1268  12.09 <0.0001
##  INDICE.SOCIO -0.8161 0.0420 -19.43 <0.0001
##
```

Vamos fazer agora um modelo com mais de uma variável previsora: VD ~ FAIXA.ETARIA + REGIAO. Chame esse modelo de `mod4`.

```
mod4 <- glm(VD ~ FAIXA.ETARIA + REGIAO, data = dados, family = binomia
l)
```

Visualize o resultado de `mod4` com `summary()`.

```
summary(mod4)
```

```
##
## Call:
## glm(formula = VD ~ FAIXA.ETARIA + REGIAO, family = binomial,
##     data = dados)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1156  -0.8393  -0.6064   1.2405   1.9638
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -1.05605    0.04841 -21.816   <2e-16 ***
## FAIXA.ETARIA2a   -0.54427    0.05606  -9.708   <2e-16 ***
## FAIXA.ETARIA3a   -0.71516    0.05949 -12.021   <2e-16 ***
## REGIAOperiferica  0.90882    0.04930  18.435   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10993  on 9225  degrees of freedom
## Residual deviance: 10467  on 9222  degrees of freedom
## AIC: 10475
##
## Number of Fisher Scoring iterations: 4
```

Vemos na tabela de coeficientes o mesmo resultado que havíamos visto para FAIXA.ETARIA (quanto mais velho, menor a tendência a usar retroflexo) e a correlação significativa com REGIAO (maior tendência ao retroflexo para os habitantes de bairros periféricos). Esse resultado não é surpreendente se levarmos em conta o resultado anterior para INDICE.SOCIO, uma vez que região de residência em São Paulo (e em muitas cidades) é um indicativo da classe social do falante.

Visualize os resultados com um gráfico de efeitos (Figura 14.5).

```
plot(allEffects(mod4), type = "response")
```

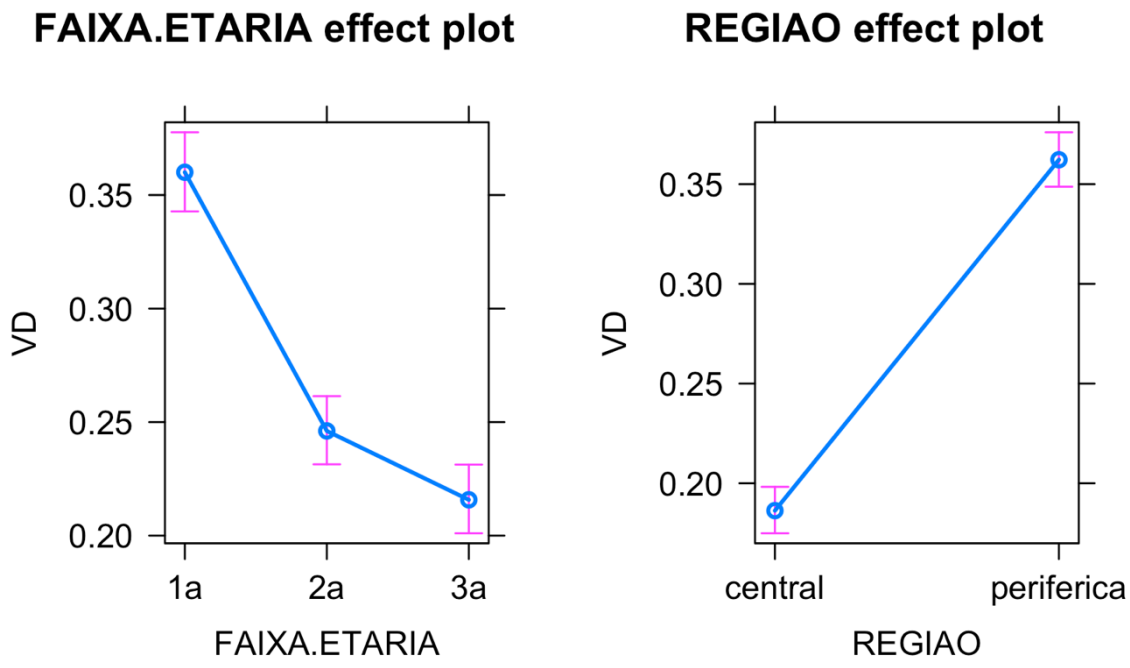

Figura 14.5: Gráfico de efeitos das variáveis Faixa Etária e Região de Residência para o uso de /r/ retroflexo. Fonte: própria.

E obtenha as demais estatísticas com a função `lrm()`.

```
lrm(VD ~ FAIXA.ETARIA + REGIAO, data = dados)
```

```
## Logistic Regression Model
##
##  lrm(formula = VD ~ FAIXA.ETARIA + REGIAO, data = dados)
##
##                    Model Likelihood    Discrimination    Rank Discri
m.
##                          Ratio Test           Indexes          Index
es
##Obs       9226    LR chi2     526.35    R2       0.080    C       0.
648
##tepe      6615    d.f.             3    g        0.616    Dxy     0.
296
##retr      2611    Pr(> chi2) <0.0001    gr       1.851    gamma   0.
352
##max |deriv| 9e-10                       gp       0.119    tau-a   0.
120
##                                        Brier    0.191
##
##                     Coef    S.E.   Wald Z Pr(>|Z|)
##  Intercept          -1.0560 0.0484 -21.82 <0.0001
##  FAIXA.ETARIA=2a    -0.5443 0.0561  -9.71 <0.0001
##  FAIXA.ETARIA=3a    -0.7152 0.0595 -12.02 <0.0001
##  REGIAO=periferica   0.9088 0.0493  18.44 <0.0001
##
```

Perdoe-me a repetição, mas é justamente isso que vai torná-lo proficiente em R! Façamos um último modelo simples, agora com interação.

A partir do último modelo criado com `glm()` com as variáveis FAIXA.ETARIA e REGIAO, mude o operador de soma para a multiplicação para testar se essas variáveis interagem entre si. Guarde o resultado num objeto chamado `mod5`.

```
mod5 <- glm(VD ~ FAIXA.ETARIA * REGIAO, data = dados, family = binomia
l)
```

Visualize o resultado de `mod5`.

```
summary(mod5)
```

```
##
## Call:
## glm(formula = VD ~ FAIXA.ETARIA * REGIAO, family = binomial,
##     data = dados)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1847  -0.8046  -0.6561   1.1701   1.8970
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|
)
## (Intercept)                     -1.31754    0.06490 -20.301  < 2e-1
```

```
6
## FAIXA.ETARIA2a                   -0.10887    0.09287  -1.172  0.2411
2
## FAIXA.ETARIA3a                   -0.30091    0.09562  -3.147  0.0016
5
## REGIAOperiferica                  1.33476    0.08168  16.341  < 2e-1
6
## FAIXA.ETARIA2a:REGIAOperiferica -0.69440    0.11688  -5.941 2.83e-0
9
## FAIXA.ETARIA3a:REGIAOperiferica -0.67794    0.12268  -5.526 3.27e-0
8
##
## (Intercept)                      ***
## FAIXA.ETARIA2a
## FAIXA.ETARIA3a                   **
## REGIAOperiferica                 ***
## FAIXA.ETARIA2a:REGIAOperiferica ***
## FAIXA.ETARIA3a:REGIAOperiferica ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10993  on 9225  degrees of freedom
## Residual deviance: 10421  on 9220  degrees of freedom
## AIC: 10433
##
## Number of Fisher Scoring iterations: 4
```

Nesse modelo com interação, a diferença entre a 1ª e a 2ª faixas etárias deixou de ser significativa (mas mantém-se a diferença entre a 1ª e a 3ª). A região periférica tem um coeficiente ainda mais alto do que em `mod4`, em favorecimento do retroflexo. Além disso, a interação entre periferia e as duas faixas etárias mais velhas é significativa; tais interações indicam que é necessário diminuir a estimativa para falantes de 2ª e 3ª faixas etárias da periferia para corrigi-lo em relação a um modelo sem interação.

Calcule a estimativa de emprego de retroflexo para um paulistano de 1a faixa etária que mora na periferia.

```
-1.31754 + 1.33476
```

```
## [1] 0.01722
```

Calcule a estimativa de emprego de retroflexo de um paulistano da 2a faixa etária que mora na região periférica. Note que, além das estimativas desses níveis, é necessário somar a estimativa do termo de interação.

```
-1.31754 -0.10887 + 1.33476 -0.69440
```

```
## [1] -0.78605
```

Com uma interação, fica ainda mais difícil interpretar os resultados numéricos, não? O melhor é *ver* o resultado. Faça um gráfico de efeitos de `mod5` para visualizar essas estimativas na escala de probabilidade (Figura 14.6).

```
plot(allEffects(mod5), type = "response")
```

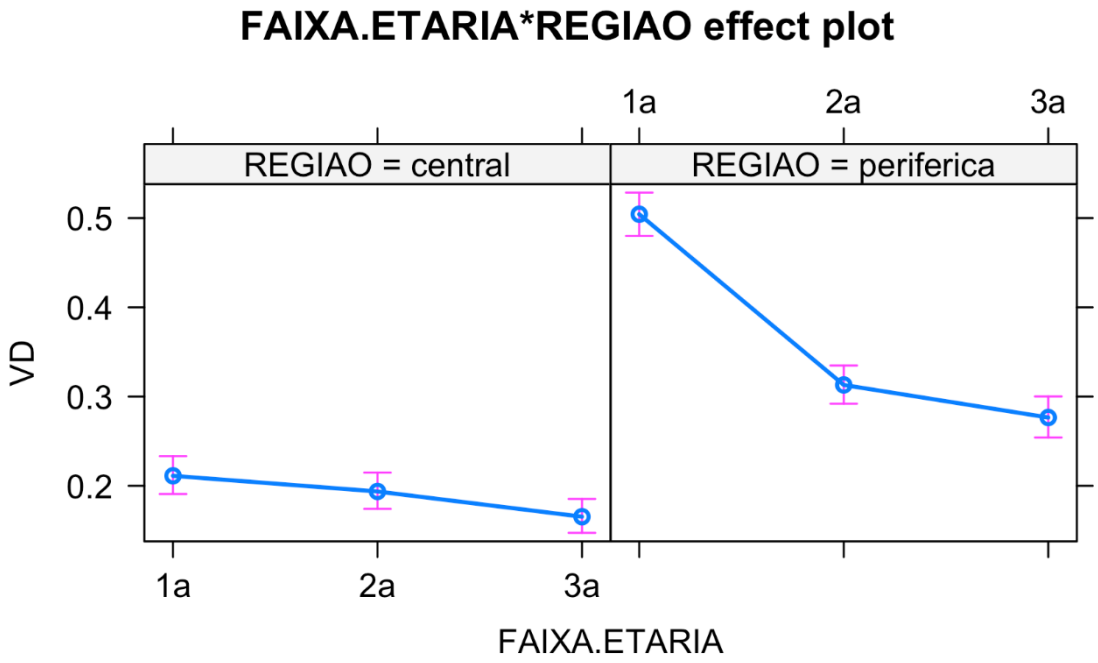

Figura 14.6: Gráfico de efeitos da interação entre as variáveis Faixa Etária e Região de Residência para o uso de /r/ retroflexo. Fonte: própria.

O gráfico de efeitos da interação mostra mais claramente que uma mudança na direção do retroflexo está ocorrendo na região periférica de São Paulo. Os falantes da região central, diferentemente, apresentam uma relativa estabilidade, com uma tendência bem mais tímida na direção do retroflexo. São os jovens de periferia que têm liderado essa possível mudança na comunidade. O fato de que os jovens da região central e da região periférica se comportam diferentemente é justamente o que caracteriza a interação: não é possível afirmar que há uma mudança na direção do retroflexo simplesmente: isso depende da região de residência do falante. Em outras palavras: FAIXA.ETARIA e REGIAO não são independentes, mas interagem entre si.

Aplique por fim a função `lrm()` ao modelo com interação para obter outras medidas estatísticas.

```
lrm(VD ~ FAIXA.ETARIA * REGIAO, data = dados)

## Logistic Regression Model
##
##  lrm(formula = VD ~ FAIXA.ETARIA * REGIAO, data = dados)
##
##                    Model Likelihood    Discrimination    Rank Discri
m.
##                          Ratio Test           Indexes          Index
es
##Obs       9226    LR chi2     571.97    R2       0.086    C       0.
648
##tepe      6615    d.f.             5    g        0.591    Dxy     0.
296
##retr      2611    Pr(> chi2) <0.0001    gr       1.806    gamma   0.
352
##max |deriv| 2e-09                       gp       0.120    tau-a   0.
120
##                                        Brier    0.190
##
##                                     Coef    S.E.   Wald Z Pr(>|Z|)
##  Intercept                          -1.3175 0.0649 -20.30 <0.0001
##  FAIXA.ETARIA=2a                    -0.1089 0.0929  -1.17 0.2411
##  FAIXA.ETARIA=3a                    -0.3009 0.0956  -3.15 0.0016
##  REGIAO=periferica                   1.3348 0.0817  16.34 <0.0001
##  FAIXA.ETARIA=2a * REGIAO=periferica -0.6944 0.1169  -5.94 <0.0001
##  FAIXA.ETARIA=3a * REGIAO=periferica -0.6779 0.1227  -5.53 <0.0001
##
```

Ao final do *script*, deixei os comandos para criar gráficos de efeitos de nossos modelos com funções do ggplot2, assim como as funções `logit()` e `ilogit()` que usamos para converter entre as medidas de probabilidade em diferentes escalas.

Na próxima lição, veremos modelos de regressão logística mais complexos e a aplicação de modelos de efeitos mistos a variáveis nominais binárias.

Para saber mais

Para saber mais sobre regressão logística e outros modelos aplicáveis a variáveis nominais, veja os capítulos 12 e 13 de Levshina (2015) e o capítulo 5 de Gries (2019).

Exercícios

Nesta lista de exercícios, você vai desenvolver uma análise semelhante à que fizemos na Lição 14, mas agora com outras variáveis previsoras. Primeiro, carregue os dados da planilha DadosRT.csv. Após carregar o dataframe, cheque sua estrutura, como de praxe. Certifique-se de que "retroflexo" é o segundo nível da VD, e organize variáveis ordinais em ordem lógica, de menor para maior. Para a variável REGIAO, defina a região central como o primeiro nível.

1. Faça um modelo de regressão logística para testar se há correlação entre a pronúncia de /r/ em coda (VD) e a posição da sílaba com /r/ na palavra (POSICAO.R). Há correlação significativa? Justifique sua resposta.
2. A qual nível de POSICAO.R se refere o coeficiente linear (Intercept)?
    a. posição medial
    b. posição final
3. A probabilidade de empregar retroflexo quando está no meio da palavra é...
    a. menor do que quando está no final da palavra
    b. maior do que quando está no final da palavra
4. Qual é o valor de índice C do modelo VD ~ POSICAO.R?
5. De acordo com a classificação de Hosmer & Lemeshow (2000, apud Levshina, 2015, p. 259), tal índice C tem...
    a. pouco poder de discriminação de resultado
    b. poder aceitável de discriminação de resultado
    c. poder excelente de discriminação de resultado
    d. poder notório de discriminação de resultado
6. Faça um modelo para testar se há correlação entre a pronúncia variável de /r/ em coda (VD) e a ESCOLARIDADE do falante. De acordo com o resultado deste modelo, não há diferença significativa entre...
    a. falantes com Ens. Fundamental e Ens.Médio
    b. falantes com Ens. Fundamental e Ens.Superior
    c. falantes com Ens. Médio e Ens. Superior

7. Em relação aos falantes menos escolarizados, os falantes com nível superior de escolaridade...
    a. tendem a empregar menos o retroflexo
    b. tendem a empregar mais o retroflexo
8. Transforme a medida de logodds da estimativa de emprego de retroflexo para falantes com nível superior para a medida de probabilidade (= proporção de 0% a 100%). Qual é a probabilidade de emprego de retroflexo para esses falantes?
    a. 21,9%
    b. 22,3%
    c. 33,9%
    d. 34,1%
    e. 50%
9. Faça um modelo para testar se há correlação entre a pronúncia variável de /r/ em coda (VD) e a IDADE do falante. A partir dele, calcule a estimativa, em logodds, de um falante com 50 anos de idade empregar o retroflexo.
10. Transforme o valor de logodds calculado na questão 9 para um valor de probabilidade.
11. Nesta lição, verificamos a interação entre FAIXA.ETARIA e REGIAO de residência do falante. A variável IDADE nada mais é do que a variável FAIXA.ETARIA vista de modo contínuo. Faça um modelo para testar a interação entre IDADE e REGIAO quanto ao uso variável de /r/ em coda. Há interação entre essas variáveis? Justifique sua resposta.
12. Calcule a probabilidade, em logodds, de um falante de 30 anos que mora na região central empregar o retroflexo.
13. Calcule a probabilidade, em logodds, de um falante de 30 anos que mora na região periférica empregar o retroflexo.
14. Transforme esta última medida de logodds em probabilidade.

Lição 15: Regressão Logística Parte 2

N.B.: Rode as linhas de comando a seguir antes de fazer esta lição. Defina como diretório de trabalho aquele que contém o arquivo DadosRT.csv.

```
# Definir diretório de trabalho

#setwd()

# Importar dados da planilha

dados <- read_csv("DadosRT.csv",
                  col_types = cols(.default = col_factor(),
                                   VD = col_factor(levels = c("tepe",
"retroflexo")),
                                   FAIXA.ETARIA = col_factor(levels =
c("1a", "2a", "3a")),
                                   ESCOLARIDADE = col_factor(levels =
c("fundamental", "medio", "superior")),
                                   REGIAO = col_factor(levels = c("cen
tral", "periferica")),
                                   CONT.FON.PREC = col_factor(levels =
c("i", "e", "3", "a", "0", "o", "u")),
                                   TONICIDADE = col_factor(levels = c(
"atona", "tonica")),
                                   POSICAO.R = col_factor(levels = c("
final", "medial")),
                                   CLASSE.MORFOLOGICA = col_factor(lev
els = c("adjetivo", "adverbio", "conj.prep", "morf.inf", "substantivo"
, "verbo")),
                                   IDADE = col_integer(),
                                   INDICE.SOCIO = col_double(),
                                   FREQUENCIA = col_double()
                                   )
                  )

dados$CONT.FON.SEG <- fct_collapse(dados$CONT.FON.SEG,
                                   pausa = "#",
                                   coronal = c("t", "d", "s", "z", "x"
, "j", "ts", "dz", "l", "n"),
                                   labial = c("p", "b", "f", "v", "m")
,
                                   dorsal = c("k", "g", "h")
                                   )

dados$CONT.FON.SEG <- fct_relevel(dados$CONT.FON.SEG, "pausa", "corona
l", "dorsal", "labial")

###Funções úteis (Gries 2019)
```

```
logit <- function(x) {
  log(x/(1-x))
  }

ilogit <- function(x) {
  1/(1+exp(-x))
  }
```

Nesta lição, vamos continuar trabalhando sobre o conjunto de dados da pronúncia variável de /r/ em coda silábica na fala de paulistanos. Como você já sabe, é uma boa ideia carregar os pacotes necessários para a sessão. Carregue os pacotes rms, effects, car, lme4 e lmerTest.

```
library(rms)
library(effects)
library(car)
library(lme4)
library(lmerTest)
```

No dataframe `dados`, o nível de referência de VD já foi redefinido, de modo que *tepe* é o primeiro nível e *retroflexo* é o segundo – e é aquele de acordo com o qual os resultados devem ser lidos (ver Lição 14).

Imagine que um pesquisador tenha levantado a hipótese de que a pronúncia de /r/ em coda (VD) se correlaciona com as variáveis SEXO.GENERO, FAIXA.ETARIA, REGIAO, INDICE.SOCIO, CONT.FON.PREC, CONT.FON.SEG, TONICIDADE, POSICAO.R e CLASSE.MORFOLOGICA. Aplique a função `str()` a `dados` para checar os níveis dessas variáveis previsoras.

```
str(dados)
```

```
## spec_tbl_df [9,226 × 20] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ VD               : Factor w/ 2 levels “tepe”,”retroflexo”: 2 2
2 1 2 2 2 2 1 2 ...
##  $ PARTICIPANTE     : Factor w/ 118 levels “IvanaB”,”HeloisaS”,..:
1 1 1 1 1 1 1 1 1 1 ...
##  $ SEXO.GENERO      : Factor w/ 2 levels “feminino”,”masculino”: 1
1 1 1 1 1 1 1 1 1 ...
##  $ IDADE            : int [1:9226] 30 30 30 30 30 30 30 30 30 30 .
..
##  $ FAIXA.ETARIA     : Factor w/ 3 levels “1a”,”2a”,”3a”: 1 1 1 1 1
1 1 1 1 1 ...
##  $ ESCOLARIDADE     : Factor w/ 3 levels “fundamental”,..: 1 1 1 1
1 1 1 1 1 1 ...
##  $ REGIAO           : Factor w/ 2 levels “central”,”periferica”: 2
2 2 2 2 2 2 2 2 ...
```

```
##  $ INDICE.SOCIO      : num [1:9226] 2 2 2 2 2 2 2 2 2 2 ...
##  $ ORIGEM.PAIS       : Factor w/ 5 levels “mista”,”SPcapital”,..: 1
1 1 1 1 1 1 1 1 1 ...
##  $ CONT.FON.PREC     : Factor w/ 7 levels “i”,”e”,”3”,”a”,..: 4 6 2
2 4 4 5 4 5 3 ...
##  $ CONT.FON.SEG      : Factor w/ 4 levels “pausa”,”coronal”,..: 2 2
3 4 3 2 2 2 3 2 ...
##  $ TONICIDADE        : Factor w/ 2 levels “atona”,”tonica”: 2 1 1 1
2 2 2 2 1 2 ...
##  $ POSICAO.R         : Factor w/ 2 levels “final”,”medial”: 2 2 2 2
1 1 2 2 2 2 ...
##  $ CLASSE.MORFOLOGICA: Factor w/ 6 levels “adjetivo”,”adverbio”,..:
5 5 5 5 5 5 5 5 3 5 ...
##  $ FREQUENCIA        : num [1:9226] 1.34 0.16 0.22 0.44 1.94 1.94 0
.35 0.03 5.98 0.16 ...
##  $ ESTILO            : Factor w/ 4 levels “conversacao”,..: 1 1 1 1
1 1 1 1 1 1 ...
##  $ ITEM.LEXICAL      : Factor w/ 1151 levels “parte”,”jornal”,..: 1
2 3 4 5 5 6 7 8 9 ...
##  $ cont.precedente   : Factor w/ 6836 levels “do CEU é daquela”,..:
1 2 3 4 5 6 7 8 9 10 ...
##  $ ocorrencia        : Factor w/ 1760 levels “parte <R>“,”jornal <R
>“,..: 1 2 3 4 5 5 6 7 8 9 ...
##  $ cont.seguinte     : Factor w/ 6813 levels “que as perua(s) ia”,.
.: 1 2 3 4 5 6 7 8 9 10 ...
##  - attr(*, “spec”)=
##   .. cols(
##   ..   .default = col_factor(),
##   ..   VD = col_factor(levels = c(“tepe”, “retroflexo”), ordered =
FALSE, include_na = FALSE),
##   ..   PARTICIPANTE = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   SEXO.GENERO = col_factor(levels = NULL, ordered = FALSE, inc
lude_na = FALSE),
##   ..   IDADE = col_integer(),
##   ..   FAIXA.ETARIA = col_factor(levels = c(“1a”, “2a”, “3a”), orde
red = FALSE, include_na = FALSE),
##   ..   ESCOLARIDADE = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   REGIAO = col_factor(levels = c(“central”, “periferica”), ord
ered = FALSE, include_na = FALSE),
##   ..   INDICE.SOCIO = col_double(),
##   ..   ORIGEM.PAIS = col_factor(levels = NULL, ordered = FALSE, inc
lude_na = FALSE),
##   ..   CONT.FON.PREC = col_factor(levels = c(“i”, “e”, “3”, “a”, “0
”, “o”, “u”), ordered = FALSE, include_na = FALSE),
##   ..   CONT.FON.SEG = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   TONICIDADE = col_factor(levels = c(“atona”, “tonica”), order
ed = FALSE, include_na = FALSE),
##   ..   POSICAO.R = col_factor(levels = c(“final”, “medial”), ordere
d = FALSE, include_na = FALSE),
##   ..   CLASSE.MORFOLOGICA = col_factor(levels = c(“adjetivo”, “adve
rbio”, “conj.prep”, “morf.inf”, “substantivo”,
```

```
##   ..     "verbo"), ordered = FALSE, include_na = FALSE),
##   ..   FREQUENCIA = col_double(),
##   ..   ESTILO = col_factor(levels = NULL, ordered = FALSE, include_
na = FALSE),
##   ..   ITEM.LEXICAL = col_factor(levels = NULL, ordered = FALSE, in
clude_na = FALSE),
##   ..   cont.precedente = col_factor(levels = NULL, ordered = FALSE,
include_na = FALSE),
##   ..   ocorrencia = col_factor(levels = NULL, ordered = FALSE, incl
ude_na = FALSE),
##   ..   cont.seguinte = col_factor(levels = NULL, ordered = FALSE, i
nclude_na = FALSE)
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Algumas dessas variáveis não são totalmente ortogonais entre si e apresentam o risco de multicolinearidade (ver Lição 13). Duas variáveis são ortogonais quando todos os níveis de uma coocorrem com todos os níveis da outra. De modo simples, todas as combinações entre os níveis são possíveis. No entanto, este não é o caso para CONT.FON.SEG e POSICAO.R; essas variáveis não são totalmente ortogonais pois o contexto seguinte "pausa" só ocorre em final de palavra, nunca no meio. Faça uma tabela de frequências entre essas duas variáveis para visualizar isso. Dessa vez, vamos usar as funções do pacote base, `with()` e `table()`.

```
with(dados, table(CONT.FON.SEG, POSICAO.R))
```

```
##             POSICAO.R
## CONT.FON.SEG final medial
##      pausa     725      0
##      coronal   742   4090
##      dorsal    134   1908
##      labial    243   1384
```

As variáveis CLASSE.MORFOLOGICA e POSICAO.R também não são ortogonais entre si. Faça uma tabela de frequências entre elas para verificar a combinação que não ocorre.

```
with(dados, table(CLASSE.MORFOLOGICA, POSICAO.R))
```

```
##                   POSICAO.R
## CLASSE.MORFOLOGICA final medial
##        adjetivo      204   1326
##        adverbio       14     23
##        conj.prep      44    925
##        morf.inf      683      0
##        substantivo   766   4060
##        verbo         133   1048
```

Na codificação de CLASSE.MORFOLOGICA, foi feita uma separação entre o /r/ de infinitivo dos verbos (p.ex., “amar”) e o /r/ que ocorre na raiz da palavra (p.ex., “ergue”). O /r/ infinitivo no português só ocorre em final de palavra, nunca no meio. Ao mesmo tempo, o /r/ infinitivo é sempre tônico, de modo que CLASSE.MORFOLOGICA também não é totalmente ortogonal a TONICIDADE. Faça uma tabela de frequências entre CLASSE.MORFOLOGICA e TONICIDADE para visualizar isso.

```
with(dados, table(CLASSE.MORFOLOGICA, TONICIDADE))
```

```
##                   TONICIDADE
## CLASSE.MORFOLOGICA atona tonica
##        adjetivo      579    951
##        adverbio       21     16
##        conj.prep     868    101
##        morf.inf        0    683
##        substantivo  2302   2524
##        verbo         845    336
```

Tabelas de frequências como essas devem ser feitas sempre que se suspeita ou se prevê que duas variáveis não são ortogonais entre si. A presença de células vazias ou com poucos dados é o principal causador de multicolinearidade, o que, como visto na Lição 13, viola os pressupostos de modelos de regressão. Contudo, a depender do grau de colinearidade e do número de dados de que se dispõe, as (poucas) células vazias podem não ser um problema.

Faça um modelo de regressão logística `mod` que inclui as variáveis previsoras TONICIDADE, POSICAO.R, CLASSE.MORFOLOGICA e CONT.FON.SEG, em relação a VD.

```
mod <- glm(VD ~ TONICIDADE + POSICAO.R + CLASSE.MORFOLOGICA + CONT.FON
.SEG, data = dados, family = binomial)
```

Na Lição 13, vimos uma função que permite avaliar se há colinearidade entre variáveis. Qual é ela?

- `crPlot()`, do pacote car
- `effect()`, do pacote effects
- `lmer()`, do pacote lme4
- `lrm()`, do pacote rms
- `vif()`, do pacote car

Aplique então a função `vif()` a `mod`. Digite `car::vif(mod)`. (A notação pacote::funcao() é outro modo de disponibilizar uma biblioteca no R.)

```
car::vif(mod)
```

```
##                         GVIF Df GVIF^(1/(2*Df))
## TONICIDADE          1.356133  1        1.164531
## POSICAO.R           1.880253  1        1.371223
## CLASSE.MORFOLOGICA 1.658880  5        1.051917
## CONT.FON.SEG        1.640220  3        1.085968
```

O que informa o resultado de `vif()` acima?

- há colinearidade entre as variáveis
- não há colinearidade entre as variáveis

Apesar das células vazias, o conjunto de dados é robusto o suficiente para superar a potencial multicolinearidade entre essas variáveis. Podemos então seguir com a criação de um modelo mais complexo, com a inclusão de todas as variáveis mencionadas acima.

Na Lição 13, vimos que há duas abordagens principais para decidir a manutenção ou não de variáveis previsoras num modelo: uma "de baixo para cima", que começa com um modelo sem variáveis previsoras e tenta adicioná-las uma a uma, e outra "de cima para baixo", que começa com um modelo completo e tenta eliminar variáveis uma a uma. Ambas se implementam com a função `step()`; a primeira com direction = "forward", e a segunda com direction = "backward".

No *script* temos um primeiro modelo completo com todas as variáveis que queremos testar. Ele já está pronto, mas você deve se atentar à sua estrutura, para reproduzir a aplicação com seus dados.

```
modelo <- glm(VD ~
                SEXO.GENERO +
                FAIXA.ETARIA * REGIAO +
                INDICE.SOCIO +
                CONT.FON.PREC +
                CONT.FON.SEG +
                TONICIDADE +
                POSICAO.R +
                CLASSE.MORFOLOGICA,
              data = dados, family = binomial)
```

Veja o resultado com `summary()`.

```
summary(modelo)

##
## Call:
## glm(formula = VD ~ SEXO.GENERO + FAIXA.ETARIA * REGIAO + INDICE.SOC
IO +
##     CONT.FON.PREC + CONT.FON.SEG + TONICIDADE + POSICAO.R + CLASSE.
MORFOLOGICA,
##     family = binomial, data = dados)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1420  -0.7736  -0.4927   0.8614   2.9249
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|
)
## (Intercept)                     1.09412    0.23689   4.619 3.86e-0
6
## SEXO.GENEROmasculino            0.54806    0.05273  10.393  < 2e-1
6
## FAIXA.ETARIA2a                 -0.14534    0.09925  -1.464 0.14311
1
## FAIXA.ETARIA3a                 -0.45935    0.10264  -4.475 7.62e-0
6
## REGIAOperiferica                1.28609    0.08849  14.534  < 2e-1
6
## INDICE.SOCIO                   -0.90138    0.04828 -18.671  < 2e-1
6
## CONT.FON.PRECe                  0.31751    0.13068   2.430 0.01510
8
## CONT.FON.PREC3                  0.52761    0.15575   3.388 0.00070
5
## CONT.FON.PRECa                  0.50403    0.13190   3.821 0.00013
3
## CONT.FON.PREC0                  0.47885    0.15456   3.098 0.00194
8
## CONT.FON.PRECo                 -0.31598    0.13825  -2.286 0.02227
7
## CONT.FON.PRECu                 -0.95710    0.17397  -5.502 3.76e-0
8
## CONT.FON.SEGcoronal             0.67861    0.11960   5.674 1.40e-0
8
## CONT.FON.SEGdorsal             -0.21684    0.13535  -1.602 0.10914
4
## CONT.FON.SEGlabial              0.23678    0.13011   1.820 0.06878
0
## TONICIDADEtonica                0.29545    0.07009   4.215 2.49e-0
5
## POSICAO.Rmedial                -0.82823    0.08958  -9.246  < 2e-1
6
## CLASSE.MORFOLOGICAadverbio      1.04562    0.37562   2.784 0.00537
4
## CLASSE.MORFOLOGICAconj.prep     0.47925    0.15094   3.175 0.00149
```

```
8
## CLASSE.MORFOLOGICAmorf.inf      -1.10812    0.14356  -7.719 1.17e-1
4
## CLASSE.MORFOLOGICAsubstantivo    0.12488    0.07681   1.626 0.10397
0
## CLASSE.MORFOLOGICAverbo          0.47789    0.09904   4.825 1.40e-0
6
## FAIXA.ETARIA2a:REGIAOperiferica -0.71666    0.12544  -5.713 1.11e-0
8
## FAIXA.ETARIA3a:REGIAOperiferica -0.76762    0.13267  -5.786 7.21e-0
9
##
## (Intercept)                     ***
## SEXO.GENEROmasculino            ***
## FAIXA.ETARIA2a
## FAIXA.ETARIA3a                  ***
## REGIAOperiferica                ***
## INDICE.SOCIO                    ***
## CONT.FON.PRECe                  *
## CONT.FON.PREC3                  ***
## CONT.FON.PRECa                  ***
## CONT.FON.PREC0                  **
## CONT.FON.PRECo                  *
## CONT.FON.PRECu                  ***
## CONT.FON.SEGcoronal             ***
## CONT.FON.SEGdorsal
## CONT.FON.SEGlabial              .
## TONICIDADEtonica                ***
## POSICAO.Rmedial                 ***
## CLASSE.MORFOLOGICAadverbio      **
## CLASSE.MORFOLOGICAconj.prep     **
## CLASSE.MORFOLOGICAmorf.inf      ***
## CLASSE.MORFOLOGICAsubstantivo
## CLASSE.MORFOLOGICAverbo         ***
## FAIXA.ETARIA2a:REGIAOperiferica ***
## FAIXA.ETARIA3a:REGIAOperiferica ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10993.1  on 9225  degrees of freedom
## Residual deviance:  9137.3  on 9202  degrees of freedom
## AIC: 9185.3
##
## Number of Fisher Scoring iterations: 5
```

Para interpretar os resultados, é necessário saber o nível de referência de cada variável. Qual é o nível de referência da variável SEXO?

- feminino
- masculino

Qual é o nível de referência da variável FAIXA.ETARIA?

- 1a
- 2a
- 3a

Qual é o nível de referência da variável REGIAO?

- central
- periferica

Qual é o nível de referência da variável INDICE.SOCIO?

- INDICE.SOCIO = 0
- INDICE.SOCIO = 5

Qual é o nível de referência da variável CONT.FON.PREC?

- a
- e
- i
- o
- u
- 3
- 0

Para a variável CONT.FON.PREC, os níveis já foram reorganizados no conjunto `dados`. Em vez da ordem alfabética, as vogais que precedem /r/ em coda foram ordenadas de [+anterior] para [-anterior], e [+alta] para [-alta]. Lembre-se que os níveis '3' e '0' correspondem às vogais média baixas /ɛ/ e /ɔ/ respectivamente.

Qual é o nível de referência da variável CONT.FON.SEG?

- coronal
- dorsal
- labial
- pausa

Assim como a variável CONT.FON.PREC, CONT.FON.SEG foi reorganizada previamente, no momento de importação dos dados. A variável foi originalmente codificada de modo detalhado, de acordo com cada segmento consonantal (/b/, /d/, /g/ etc.) ou existência de pausa, mas os segmentos foram recodificados aqui pelo Ponto de C (ver Clements & Hume, 1995). A pausa foi mantida e definida como nível de referência.

Qual é o nível de referência da variável TONICIDADE?

- atona
- tonica

Qual é o nível de referência da variável POSICAO.R?

- final
- medial

Qual é o nível de referência da variável CLASSE.MORFOLOGICA?

- adjetivo
- adverbio
- conj.prep
- morf.inf
- substantivo
- verbo

Trata-se de um modelo bastante complexo! O resultado para o coeficiente linear (Intercept), então, refere-se à pronúncia como *retroflexo* – lembre-se que os resultados são lidos em relação ao segundo nível da VD, que foi redefinida –, na fala de mulheres de 1ª faixa etária residentes da região central e de nível socioeconômico zero, em ocorrências de /r/ em coda precedidas da vogal /i/, seguidas de pausa, em sílabas átonas, em posição final de adjetivos! (respira... respira...) Veja que a leitura e interpretação de resultados fica mais difícil à medida que se incluem novas variáveis previsoras, de modo que deve haver bons motivos para incluí-las no modelo.

Aplique a função `lrm()` para obter as demais medidas estatísticas do modelo. Lembre-se que esta função toma como argumentos a fórmula e o conjunto de dados (sem family = binomial).

```
lrm(VD ~
      SEXO.GENERO +
      FAIXA.ETARIA * REGIAO +
      INDICE.SOCIO +
      CONT.FON.PREC +
      CONT.FON.SEG +
      TONICIDADE +
      POSICAO.R +
      CLASSE.MORFOLOGICA,
    data = dados)
```

```
## Logistic Regression Model
##
##  lrm(formula = VD ~ SEXO.GENERO + FAIXA.ETARIA * REGIAO + INDICE.SO
CIO
##      + CONT.FON.PREC + CONT.FON.SEG + TONICIDADE + POSICAO.R + CLAS
SE.  MORFOLOGICA,
##      data = dados)
##
##                    Model Likelihood    Discrimination    Rank Discri
m.
##                          Ratio Test           Indexes          Index
es
##Obs      9226    LR chi2    1855.86    R2       0.262    C       0.7
74
##tepe     6615    d.f.            23    g        1.333    Dxy     0.5
49
##retrofl  2611    Pr(> chi2) <0.0001    gr       3.793    gamma   0.5
49
##max |deriv| 2e-08                      gp       0.221    tau-a   0.2
23
##                                       Brier    0.163
##
##                                      Coef    S.E.   Wald Z Pr(>|Z|)
##  Intercept                            1.0941 0.2369   4.62 <0.0001
##  SEXO.GENERO=masculino                0.5481 0.0527  10.39 <0.0001
##  FAIXA.ETARIA=2a                     -0.1453 0.0993  -1.46 0.1431
##  FAIXA.ETARIA=3a                     -0.4594 0.1026  -4.48 <0.0001
##  REGIAO=periferica                    1.2861 0.0885  14.53 <0.0001
##  INDICE.SOCIO                        -0.9014 0.0483 -18.67 <0.0001
##  CONT.FON.PREC=e                      0.3175 0.1307   2.43 0.0151
##  CONT.FON.PREC=3                      0.5276 0.1557   3.39 0.0007
##  CONT.FON.PREC=a                      0.5040 0.1319   3.82 0.0001
##  CONT.FON.PREC=0                      0.4788 0.1546   3.10 0.0019
##  CONT.FON.PREC=o                     -0.3160 0.1382  -2.29 0.0223
##  CONT.FON.PREC=u                     -0.9571 0.1740  -5.50 <0.0001
##  CONT.FON.SEG=coronal                 0.6786 0.1196   5.67 <0.0001
##  CONT.FON.SEG=dorsal                 -0.2168 0.1354  -1.60 0.1091
```

```
##   CONT.FON.SEG=labial                     0.2368 0.1301   1.82 0.0688
##   TONICIDADE=tonica                       0.2954 0.0701   4.22 <0.0001
##   POSICAO.R=medial                       -0.8282 0.0896  -9.25 <0.0001
##   CLASSE.MORFOLOGICA=adverbio             1.0456 0.3756   2.78 0.0054
##   CLASSE.MORFOLOGICA=conj.prep            0.4792 0.1509   3.18 0.0015
##   CLASSE.MORFOLOGICA=morf.inf            -1.1081 0.1436  -7.72 <0.0001
##   CLASSE.MORFOLOGICA=substantivo          0.1249 0.0768   1.63 0.1040
##   CLASSE.MORFOLOGICA=verbo                0.4779 0.0990   4.83 <0.0001
##   FAIXA.ETARIA=2a * REGIAO=periferica -0.7167 0.1254  -5.71 <0.0001
##   FAIXA.ETARIA=3a * REGIAO=periferica -0.7676 0.1327  -5.79 <0.0001
##
```

Lembra que os modelos da lição anterior, com apenas uma ou duas variáveis previsoras, tinham índice C baixos? Nosso modelo agora está num nível aceitável, com C = 0,774. Mas será que todas as variáveis são relevantes?

Crie agora um modelo sem qualquer variável previsora, a partir do qual faremos o step forward. Digite `m0 <- glm(VD ~ 1, data = dados, family = binomial)`.

```
m0 <- glm(VD ~ 1, data = dados, family = binomial)
```

Na Lição 13, vimos que a função `step()`, na direção "forward", toma como argumentos: (i) um modelo sem variáveis previsoras – aqui, `m0`; (ii) direction = "forward"; e scope com ~ (sem a Variável Resposta) e a especificação de variáveis.

```
m.fw <- step(m0,
             direction = "forward",
             scope = ~
               SEXO.GENERO +
               FAIXA.ETARIA * REGIAO +
               INDICE.SOCIO +
               CONT.FON.PREC +
               CONT.FON.SEG +
               TONICIDADE +
               POSICAO.R +
               CLASSE.MORFOLOGICA)

## Start:  AIC=10995.12
## VD ~ 1
##
##                       Df Deviance   AIC
## + CONT.FON.PREC        6    10532 10546
## + INDICE.SOCIO         1    10596 10600
## + REGIAO               1    10635 10639
## + CONT.FON.SEG         3    10724 10732
## + CLASSE.MORFOLOGICA   5    10792 10804
## + FAIXA.ETARIA         2    10824 10830
## + TONICIDADE           1    10847 10851
## + SEXO.GENERO          1    10903 10907
## + POSICAO.R            1    10920 10924
```

```
## <none>                     10993 10995
##
## Step:  AIC=10546.06
## VD ~ CONT.FON.PREC
##
##                       Df Deviance   AIC
## + INDICE.SOCIO         1    10126 10142
## + REGIAO               1    10161 10177
## + FAIXA.ETARIA         2    10364 10382
## + CONT.FON.SEG         3    10390 10410
## + SEXO.GENERO          1    10451 10467
## + CLASSE.MORFOLOGICA   5    10476 10500
## + POSICAO.R            1    10497 10513
## + TONICIDADE           1    10510 10526
## <none>                      10532 10546
##
## Step:  AIC=10141.82
## VD ~ CONT.FON.PREC + INDICE.SOCIO
##
##                       Df Deviance     AIC
## + REGIAO               1   9864.8  9882.8
## + FAIXA.ETARIA         2   9891.3  9911.3
## + CONT.FON.SEG         3   9988.5 10010.5
## + SEXO.GENERO          1  10008.9 10026.9
## + CLASSE.MORFOLOGICA   5  10074.1 10100.1
## + POSICAO.R            1  10091.1 10109.1
## + TONICIDADE           1  10098.4 10116.4
## <none>                    10125.8 10141.8
##
## Step:  AIC=9882.75
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO
##
##                       Df Deviance    AIC
## + FAIXA.ETARIA         2   9646.5 9668.5
## + CONT.FON.SEG         3   9720.5 9744.5
## + SEXO.GENERO          1   9752.4 9772.4
## + CLASSE.MORFOLOGICA   5   9814.5 9842.5
## + POSICAO.R            1   9827.8 9847.8
## + TONICIDADE           1   9840.2 9860.2
## <none>                     9864.8 9882.8
##
## Step:  AIC=9668.49
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + FAIXA.ETARIA
##
##                        Df Deviance    AIC
## + CONT.FON.SEG          3   9504.4 9532.4
## + SEXO.GENERO           1   9539.4 9563.4
## + CLASSE.MORFOLOGICA    5   9596.4 9628.4
## + FAIXA.ETARIA:REGIAO   2   9603.1 9629.1
## + POSICAO.R             1   9606.8 9630.8
## + TONICIDADE            1   9621.0 9645.0
## <none>                      9646.5 9668.5
##
## Step:  AIC=9532.4
```

```
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + FAIXA.ETARIA + CONT.FO
N.SEG
##
##                        Df Deviance    AIC
## + SEXO.GENERO           1   9401.0 9431.0
## + POSICAO.R             1   9439.3 9469.3
## + CLASSE.MORFOLOGICA    5   9438.5 9476.5
## + FAIXA.ETARIA:REGIAO   2   9461.3 9493.3
## + TONICIDADE            1   9476.4 9506.4
## <none>                      9504.4 9532.4
##
## Step:  AIC=9430.96
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + FAIXA.ETARIA + CONT.FO
N.SEG +
##     SEXO.GENERO
##
##                        Df Deviance    AIC
## + POSICAO.R             1   9332.5 9364.5
## + CLASSE.MORFOLOGICA    5   9336.2 9376.2
## + FAIXA.ETARIA:REGIAO   2   9354.8 9388.8
## + TONICIDADE            1   9371.8 9403.8
## <none>                      9401.0 9431.0
##
## Step:  AIC=9364.48
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + FAIXA.ETARIA + CONT.FO
N.SEG +
##     SEXO.GENERO + POSICAO.R
##
##                        Df Deviance    AIC
## + CLASSE.MORFOLOGICA    5   9201.0 9243.0
## + FAIXA.ETARIA:REGIAO   2   9286.6 9322.6
## + TONICIDADE            1   9328.6 9362.6
## <none>                      9332.5 9364.5
##
## Step:  AIC=9242.96
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + FAIXA.ETARIA + CONT.FO
N.SEG +
##     SEXO.GENERO + POSICAO.R + CLASSE.MORFOLOGICA
##
##                        Df Deviance    AIC
## + FAIXA.ETARIA:REGIAO   2   9154.9 9200.9
## + TONICIDADE            1   9183.0 9227.0
## <none>                      9201.0 9243.0
##
## Step:  AIC=9200.92
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + FAIXA.ETARIA + CONT.FO
N.SEG +
##     SEXO.GENERO + POSICAO.R + CLASSE.MORFOLOGICA + REGIAO:FAIXA.ETA
RIA
##
##              Df Deviance    AIC
## + TONICIDADE  1   9137.3 9185.3
## <none>            9154.9 9200.9
##
```

```
## Step:  AIC=9185.26
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + FAIXA.ETARIA + CONT.FO
N.SEG +
##     SEXO.GENERO + POSICAO.R + CLASSE.MORFOLOGICA + TONICIDADE +
##     REGIAO:FAIXA.ETARIA
```

O resultado é bem mais longo do que havíamos visto para o modelo de regressão linear porque aqui incluímos mais variáveis previsoras. O resultado de step forward indica que todas as variáveis devem ser mantidas no modelo.

Visualize também o resultado guardado em `m.fw`.

```
m.fw

##
## Call:  glm(formula = VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + F
AIXA.ETARIA +
##     CONT.FON.SEG + SEXO.GENERO + POSICAO.R + CLASSE.MORFOLOGICA +
##     TONICIDADE + REGIAO:FAIXA.ETARIA, family = binomial, data = dad
os)
##
## Coefficients:
##                       (Intercept)                    CONT.FON.PRECe
##                            1.0941                            0.3175
##                   CONT.FON.PREC3                    CONT.FON.PRECa
##                            0.5276                            0.5040
##                   CONT.FON.PREC0                    CONT.FON.PRECo
##                            0.4788                           -0.3160
##                   CONT.FON.PRECu                      INDICE.SOCIO
##                           -0.9571                           -0.9014
##                 REGIAOperiferica                    FAIXA.ETARIA2a
##                            1.2861                           -0.1453
##                   FAIXA.ETARIA3a               CONT.FON.SEGcoronal
##                           -0.4594                            0.6786
##               CONT.FON.SEGdorsal                CONT.FON.SEGlabial
##                           -0.2168                            0.2368
##             SEXO.GENEROmasculino                   POSICAO.Rmedial
##                            0.5481                           -0.8282
##      CLASSE.MORFOLOGICAadverbio      CLASSE.MORFOLOGICAconj.prep
##                            1.0456                            0.4792
##      CLASSE.MORFOLOGICAmorf.inf    CLASSE.MORFOLOGICAsubstantivo
##                           -1.1081                            0.1249
##         CLASSE.MORFOLOGICAverbo                  TONICIDADEtonica
##                            0.4779                            0.2954
## REGIAOperiferica:FAIXA.ETARIA2a  REGIAOperiferica:FAIXA.ETARIA3a
##                           -0.7167                           -0.7676
##
## Degrees of Freedom: 9225 Total (i.e. Null);  9202 Residual
## Null Deviance:       10990
## Residual Deviance: 9137  AIC: 9185
```

`m.fw` mostra os coeficientes angulares do modelo final. Façamos agora o modelo "de trás para frente", a fim de verificar se as mesmas variáveis previsoras são selecionadas. Digite `m.bw <- step(modelo, direction = "backward")`.

```
m.bw <- step(modelo, direction = "backward")

## Start:  AIC=9185.26
## VD ~ SEXO.GENERO + FAIXA.ETARIA * REGIAO + INDICE.SOCIO + CONT.FON.
PREC +
##     CONT.FON.SEG + TONICIDADE + POSICAO.R + CLASSE.MORFOLOGICA
##
##                        Df Deviance    AIC
## <none>                      9137.3 9185.3
## - TONICIDADE            1   9154.9 9200.9
## - FAIXA.ETARIA:REGIAO   2   9183.0 9227.0
## - POSICAO.R             1   9222.9 9268.9
## - SEXO.GENERO           1   9247.3 9293.3
## - CLASSE.MORFOLOGICA    5   9282.9 9320.9
## - CONT.FON.SEG          3   9304.3 9346.3
## - CONT.FON.PREC         6   9369.8 9405.8
## - INDICE.SOCIO          1   9508.3 9554.3
```

Assim como fizemos na Lição 13, aqui partimos do modelo completo (`modelo`) e tentamos eliminar variáveis. O resultado final recomenda a manutenção de todas as variáveis previsoras.

Visualize os coeficientes guardados em `m.bw`.

```
m.bw

##
## Call:  glm(formula = VD ~ SEXO.GENERO + FAIXA.ETARIA * REGIAO + IND
ICE.SOCIO +
##     CONT.FON.PREC + CONT.FON.SEG + TONICIDADE + POSICAO.R + CLASSE.
MORFOLOGICA,
##     family = binomial, data = dados)
##
## Coefficients:
##                     (Intercept)             SEXO.GENEROmasculino
##                          1.0941                          0.5481
##                 FAIXA.ETARIA2a                  FAIXA.ETARIA3a
##                         -0.1453                         -0.4594
##               REGIAOperiferica                    INDICE.SOCIO
##                          1.2861                         -0.9014
##                 CONT.FON.PRECe                  CONT.FON.PREC3
##                          0.3175                          0.5276
##                 CONT.FON.PRECa                  CONT.FON.PREC0
##                          0.5040                          0.4788
##                 CONT.FON.PRECo                  CONT.FON.PRECu
##                         -0.3160                         -0.9571
##            CONT.FON.SEGcoronal               CONT.FON.SEGdorsal
```

```
##                            0.6786                            -0.2168
##               CONT.FON.SEGlabial                 TONICIDADEtonica
##                            0.2368                             0.2954
##                   POSICAO.Rmedial       CLASSE.MORFOLOGICAadverbio
##                           -0.8282                             1.0456
##     CLASSE.MORFOLOGICAconj.prep       CLASSE.MORFOLOGICAmorf.inf
##                            0.4792                            -1.1081
##   CLASSE.MORFOLOGICAsubstantivo          CLASSE.MORFOLOGICAverbo
##                            0.1249                             0.4779
## FAIXA.ETARIA2a:REGIAOperiferica  FAIXA.ETARIA3a:REGIAOperiferica
##                           -0.7167                            -0.7676
##
## Degrees of Freedom: 9225 Total (i.e. Null);  9202 Residual
## Null Deviance:       10990
## Residual Deviance: 9137  AIC: 9185
```

O objetivo aqui é verificar se os coeficientes do modelo "para frente" coincidem com aqueles do modelo "de trás para frente". Com um pouco de paciência, verificamos que os coeficientes são os mesmos.

Aplique também a função `step()` com direction = "both". Recorde-se que neste caso a função `step()` se inicia como a direção "forward" mas, a cada variável incluída no modelo, ele tenta excluir alguma que não seja mais relevante. Os argumentos são os mesmos que na direção "forward", com a exclusão de direction (já que a opção "both" é a *default*).

```
m.both <- step(m0,
               scope = ~
                 SEXO.GENERO +
                 FAIXA.ETARIA * REGIAO +
                 INDICE.SOCIO +
                 CONT.FON.PREC +
                 CONT.FON.SEG +
                 TONICIDADE +
                 POSICAO.R +
                 CLASSE.MORFOLOGICA)

## Start:  AIC=10995.12
## VD ~ 1
##
##                       Df Deviance   AIC
## + CONT.FON.PREC        6    10532 10546
## + INDICE.SOCIO         1    10596 10600
## + REGIAO               1    10635 10639
## + CONT.FON.SEG         3    10724 10732
## + CLASSE.MORFOLOGICA   5    10792 10804
## + FAIXA.ETARIA         2    10824 10830
## + TONICIDADE           1    10847 10851
## + SEXO.GENERO          1    10903 10907
```

```
## + POSICAO.R           1    10920 10924
## <none>                     10993 10995
##
## Step:  AIC=10546.06
## VD ~ CONT.FON.PREC
##
##                      Df Deviance   AIC
## + INDICE.SOCIO        1    10126 10142
## + REGIAO              1    10161 10177
## + FAIXA.ETARIA        2    10364 10382
## + CONT.FON.SEG        3    10390 10410
## + SEXO.GENERO         1    10451 10467
## + CLASSE.MORFOLOGICA  5    10476 10500
## + POSICAO.R           1    10497 10513
## + TONICIDADE          1    10510 10526
## <none>                     10532 10546
## - CONT.FON.PREC       6    10993 10995
##
## Step:  AIC=10141.82
## VD ~ CONT.FON.PREC + INDICE.SOCIO
##
##                      Df Deviance     AIC
## + REGIAO              1   9864.8  9882.8
## + FAIXA.ETARIA        2   9891.3  9911.3
## + CONT.FON.SEG        3   9988.5 10010.5
## + SEXO.GENERO         1  10008.9 10026.9
## + CLASSE.MORFOLOGICA  5  10074.1 10100.1
## + POSICAO.R           1  10091.1 10109.1
## + TONICIDADE          1  10098.4 10116.4
## <none>                   10125.8 10141.8
## - INDICE.SOCIO        1  10532.1 10546.1
## - CONT.FON.PREC       6  10596.1 10600.1
##
## Step:  AIC=9882.75
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO
##
##                      Df Deviance     AIC
## + FAIXA.ETARIA        2   9646.5  9668.5
## + CONT.FON.SEG        3   9720.5  9744.5
## + SEXO.GENERO         1   9752.4  9772.4
## + CLASSE.MORFOLOGICA  5   9814.5  9842.5
## + POSICAO.R           1   9827.8  9847.8
## + TONICIDADE          1   9840.2  9860.2
## <none>                    9864.8  9882.8
## - REGIAO              1  10125.8 10141.8
## - INDICE.SOCIO        1  10160.7 10176.7
## - CONT.FON.PREC       6  10347.8 10353.8
##
## Step:  AIC=9668.49
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + FAIXA.ETARIA
##
##                       Df Deviance     AIC
## + CONT.FON.SEG         3   9504.4  9532.4
## + SEXO.GENERO          1   9539.4  9563.4
```

```
## + CLASSE.MORFOLOGICA   5   9596.4  9628.4
## + FAIXA.ETARIA:REGIAO  2   9603.1  9629.1
## + POSICAO.R            1   9606.8  9630.8
## + TONICIDADE           1   9621.0  9645.0
## <none>                     9646.5  9668.5
## - FAIXA.ETARIA         2   9864.8  9882.8
## - REGIAO               1   9891.3  9911.3
## - INDICE.SOCIO         1   9993.5 10013.5
## - CONT.FON.PREC        6  10133.3 10143.3
##
## Step:  AIC=9532.4
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + FAIXA.ETARIA + CONT.FO
N.SEG
##
##                        Df Deviance    AIC
## + SEXO.GENERO           1   9401.0 9431.0
## + POSICAO.R             1   9439.3 9469.3
## + CLASSE.MORFOLOGICA    5   9438.5 9476.5
## + FAIXA.ETARIA:REGIAO   2   9461.3 9493.3
## + TONICIDADE            1   9476.4 9506.4
## <none>                      9504.4 9532.4
## - CONT.FON.SEG          3   9646.5 9668.5
## - FAIXA.ETARIA          2   9720.5 9744.5
## - REGIAO                1   9755.3 9781.3
## - CONT.FON.PREC         6   9853.6 9869.6
## - INDICE.SOCIO          1   9845.5 9871.5
##
## Step:  AIC=9430.96
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + FAIXA.ETARIA + CONT.FO
N.SEG +
##     SEXO.GENERO
##
##                        Df Deviance    AIC
## + POSICAO.R             1   9332.5 9364.5
## + CLASSE.MORFOLOGICA    5   9336.2 9376.2
## + FAIXA.ETARIA:REGIAO   2   9354.8 9388.8
## + TONICIDADE            1   9371.8 9403.8
## <none>                      9401.0 9431.0
## - SEXO.GENERO           1   9504.4 9532.4
## - CONT.FON.SEG          3   9539.4 9563.4
## - FAIXA.ETARIA          2   9611.7 9637.7
## - REGIAO                1   9646.2 9674.2
## - CONT.FON.PREC         6   9745.6 9763.6
## - INDICE.SOCIO          1   9781.4 9809.4
##
## Step:  AIC=9364.48
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + FAIXA.ETARIA + CONT.FO
N.SEG +
##     SEXO.GENERO + POSICAO.R
##
##                        Df Deviance    AIC
## + CLASSE.MORFOLOGICA    5   9201.0 9243.0
## + FAIXA.ETARIA:REGIAO   2   9286.6 9322.6
## + TONICIDADE            1   9328.6 9362.6
```

```
## <none>                     9332.5 9364.5
## - POSICAO.R            1   9401.0 9431.0
## - SEXO.GENERO          1   9439.3 9469.3
## - CONT.FON.SEG         3   9497.5 9523.5
## - FAIXA.ETARIA         2   9546.1 9574.1
## - REGIAO               1   9582.9 9612.9
## - CONT.FON.PREC        6   9648.0 9668.0
## - INDICE.SOCIO         1   9715.5 9745.5
##
## Step:  AIC=9242.96
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + FAIXA.ETARIA + CONT.FO
N.SEG +
##     SEXO.GENERO + POSICAO.R + CLASSE.MORFOLOGICA
##
##                         Df Deviance    AIC
## + FAIXA.ETARIA:REGIAO  2   9154.9 9200.9
## + TONICIDADE           1   9183.0 9227.0
## <none>                     9201.0 9243.0
## - SEXO.GENERO          1   9306.9 9346.9
## - CLASSE.MORFOLOGICA   5   9332.5 9364.5
## - POSICAO.R            1   9336.2 9376.2
## - CONT.FON.SEG         3   9368.5 9404.5
## - FAIXA.ETARIA         2   9416.9 9454.9
## - REGIAO               1   9451.2 9491.2
## - CONT.FON.PREC        6   9481.9 9511.9
## - INDICE.SOCIO         1   9573.0 9613.0
##
## Step:  AIC=9200.92
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + FAIXA.ETARIA + CONT.FO
N.SEG +
##     SEXO.GENERO + POSICAO.R + CLASSE.MORFOLOGICA + REGIAO:FAIXA.ETA
RIA
##
##                         Df Deviance    AIC
## + TONICIDADE           1   9137.3 9185.3
## <none>                     9154.9 9200.9
## - REGIAO:FAIXA.ETARIA  2   9201.0 9243.0
## - SEXO.GENERO          1   9264.4 9308.4
## - CLASSE.MORFOLOGICA   5   9286.6 9322.6
## - POSICAO.R            1   9289.8 9333.8
## - CONT.FON.SEG         3   9321.8 9361.8
## - CONT.FON.PREC        6   9439.8 9473.8
## - INDICE.SOCIO         1   9520.7 9564.7
##
## Step:  AIC=9185.26
## VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + FAIXA.ETARIA + CONT.FO
N.SEG +
##     SEXO.GENERO + POSICAO.R + CLASSE.MORFOLOGICA + TONICIDADE +
##     REGIAO:FAIXA.ETARIA
##
##                         Df Deviance    AIC
## <none>                     9137.3 9185.3
## - TONICIDADE           1   9154.9 9200.9
## - REGIAO:FAIXA.ETARIA  2   9183.0 9227.0
```

```
## - POSICAO.R            1   9222.9 9268.9
## - SEXO.GENERO          1   9247.3 9293.3
## - CLASSE.MORFOLOGICA   5   9282.9 9320.9
## - CONT.FON.SEG         3   9304.3 9346.3
## - CONT.FON.PREC        6   9369.8 9405.8
## - INDICE.SOCIO         1   9508.3 9554.3
```

E veja o resultado de `m.both`.

```
m.both

##
## Call:  glm(formula = VD ~ CONT.FON.PREC + INDICE.SOCIO + REGIAO + F
AIXA.ETARIA +
##     CONT.FON.SEG + SEXO.GENERO + POSICAO.R + CLASSE.MORFOLOGICA +
##     TONICIDADE + REGIAO:FAIXA.ETARIA, family = binomial, data = dad
os)
##
## Coefficients:
##                       (Intercept)                    CONT.FON.PRECe
##                            1.0941                            0.3175
##                   CONT.FON.PREC3                    CONT.FON.PRECa
##                            0.5276                            0.5040
##                   CONT.FON.PREC0                    CONT.FON.PRECo
##                            0.4788                           -0.3160
##                   CONT.FON.PRECu                      INDICE.SOCIO
##                           -0.9571                           -0.9014
##                 REGIAOperiferica                    FAIXA.ETARIA2a
##                            1.2861                           -0.1453
##                   FAIXA.ETARIA3a               CONT.FON.SEGcoronal
##                           -0.4594                            0.6786
##               CONT.FON.SEGdorsal                CONT.FON.SEGlabial
##                           -0.2168                            0.2368
##             SEXO.GENEROmasculino                   POSICAO.Rmedial
##                            0.5481                           -0.8282
##      CLASSE.MORFOLOGICAadverbio      CLASSE.MORFOLOGICAconj.prep
##                            1.0456                            0.4792
##      CLASSE.MORFOLOGICAmorf.inf    CLASSE.MORFOLOGICAsubstantivo
##                           -1.1081                            0.1249
##         CLASSE.MORFOLOGICAverbo                 TONICIDADEtonica
##                            0.4779                            0.2954
## REGIAOperiferica:FAIXA.ETARIA2a  REGIAOperiferica:FAIXA.ETARIA3a
##                           -0.7167                           -0.7676
##
## Degrees of Freedom: 9225 Total (i.e. Null);  9202 Residual
## Null Deviance:       10990
## Residual Deviance: 9137  AIC: 9185
```

Todas as variáveis são novamente selecionadas e os coeficientes coincidem. Além da função `step()`, podemos aplicar a função `drop1()` também a modelos de regressão logística para verificar a significância de cada variável previsora no modelo e se alguma

deve ser descartada. Mas diferentemente do modelo de regressão linear, aqui usaremos test = "LR". Digite então `drop1(modelo, test = "LR")`.

```
drop1(modelo, test = "LR")
```

```
## Single term deletions
##
## Model:
## VD ~ SEXO.GENERO + FAIXA.ETARIA * REGIAO + INDICE.SOCIO + CONT.FON.
PREC +
##     CONT.FON.SEG + TONICIDADE + POSICAO.R + CLASSE.MORFOLOGICA
##                       Df Deviance    AIC    LRT  Pr(>Chi)
## <none>                     9137.3 9185.3
## SEXO.GENERO            1   9247.3 9293.3 110.02 < 2.2e-16 ***
## INDICE.SOCIO           1   9508.3 9554.3 371.00 < 2.2e-16 ***
## CONT.FON.PREC          6   9369.8 9405.8 232.50 < 2.2e-16 ***
## CONT.FON.SEG           3   9304.3 9346.3 167.05 < 2.2e-16 ***
## TONICIDADE             1   9154.9 9200.9  17.66 2.636e-05 ***
## POSICAO.R              1   9222.9 9268.9  85.63 < 2.2e-16 ***
## CLASSE.MORFOLOGICA     5   9282.9 9320.9 145.60 < 2.2e-16 ***
## FAIXA.ETARIA:REGIAO    2   9183.0 9227.0  45.69 1.198e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

O resultado de `drop1()` também informa que todas as variáveis previsoras do modelo são significativas e devem ser mantidas.

Ao incluir muitas variáveis previsoras num modelo, vale a pena verificar se não estamos "inflacionando" os parâmetros de previsão. Isso é chamado de "sobreajuste" (= overfitting), a inclusão de mais variáveis do que necessário, algo que viola o Princípio da Navalha de Occam. Isso pode ser testado por meio da função `validate()`, do pacote rms (que já carregamos acima). A função `validate()` precisa de um modelo criado com a função `lrm()` que, além da fórmula e do conjunto de dados, recebe mais dois argumentos: `x = T` e `y = T`. Rode esta linha de comando, que já está pronta.

```
mod.lrm <- lrm (VD ~
                  SEXO.GENERO +
                  FAIXA.ETARIA * REGIAO +
                  INDICE.SOCIO +
                  CONT.FON.PREC +
                  CONT.FON.SEG +
                  TONICIDADE +
                  POSICAO.R +
                  CLASSE.MORFOLOGICA,
                data = dados, x = T, y = T)
```

Podemos agora aplicar a função `validate()` ao modelo criado acima. O primeiro argumento da função é o modelo de regressão logística; o segundo argumento, B, é o número de vezes que o modelo será testado por meio de "bootstrapping"; o terceiro argumento é o modo de seleção. Vamos fazer isso 200 vezes, com modo de seleção "de trás para frente". Digite então `validate(mod.lrm, B = 200, bw = T)`. (Não se assuste se demorar. Enquanto houver um ícone redondo vermelho na parte superior da janela Console, aguarde!)

```
validate(mod.lrm, B = 200, bw = T)

##
##      Backwards Step-down - Original Model
##
## No Factors Deleted
##
## Factors in Final Model
##
##  [1] SEXO.GENERO           FAIXA.ETARIA          REGIAO
##  [4] INDICE.SOCIO          CONT.FON.PREC         CONT.FON.SEG
##  [7] TONICIDADE            POSICAO.R             CLASSE.MORFOLOGICA
## [10] FAIXA.ETARIA * REGIAO

##           index.orig training    test optimism index.corrected   n
## Dxy           0.5487   0.5521  0.5458   0.0063          0.5423 200
## R2            0.2617   0.2649  0.2589   0.0059          0.2558 200
## Intercept     0.0000   0.0000 -0.0098   0.0098         -0.0098 200
## Slope         1.0000   1.0000  0.9823   0.0177          0.9823 200
## Emax          0.0000   0.0000  0.0056   0.0056          0.0056 200
## D             0.2010   0.2037  0.1987   0.0050          0.1961 200
## U            -0.0002  -0.0002  0.0000  -0.0003          0.0000 200
## Q             0.2013   0.2039  0.1986   0.0052          0.1960 200
## B             0.1630   0.1623  0.1635  -0.0012          0.1642 200
## g             1.3331   1.3471  1.3234   0.0237          1.3095 200
## gp            0.2213   0.2223  0.2200   0.0023          0.2190 200
##
## Factors Retained in Backwards Elimination
##
##  SEXO.GENERO FAIXA.ETARIA REGIAO INDICE.SOCIO CONT.FON.PREC
##  *           *            *      *            *
##  *           *            *      *            *
##  *           *            *      *            *
##  *           *            *      *            *
##  *           *            *      *            *
##  *           *            *      *            *
##  *           *            *      *            *
##  *           *            *      *            *
##  *           *            *      *            *
```

```
## [...]
##  CONT.FON.SEG TONICIDADE POSICAO.R CLASSE.MORFOLOGICA
##  *            *          *         *
##  *            *          *         *
##  *            *          *         *
##  *            *          *         *
##  *            *          *         *
##  *            *          *         *
##  *            *          *         *
##  *            *          *         *
##  *            *          *         *
## [...]
##  FAIXA.ETARIA * REGIAO
##  *
##  *
##  *
##  *
##  *
##  *
##  *
##  *
##  *
## [...]
##
## Frequencies of Numbers of Factors Retained
##
##   9  10
##   1 199
```

N.B.: Resultado aqui abreviado.

"Bootstrapping", em informática, refere-se a um processo autossustentável que se implementa sem ajuda externa. O que a função `validate()` faz é selecionar randomicamente subamostras a partir do conjunto completo de dados, realizar o mesmo teste estatístico repetidas vezes (acima, fizemos 200), e verificar se os mesmos resultados se mantêm. Se há um número demasiado de variáveis previsoras para aquele número de dados, a função `validate()` vai acusar "otimismo" nos resultados. A coluna que nos interessa aqui é justamente "optimism", que mostra a diferença entre o treinamento do modelo e as medidas estatísticas calculadas. Um modelo válido tem valores abaixo de 0,05 na coluna "optimism".

Em seguida, o resultado mostra as variáveis que foram selecionadas em cada um dos 200 testes, por meio dos asteriscos (às vezes com linhas omitidas). Ao final, o teste informa quantas variáveis foram selecionadas quantas vezes. Neste conjunto de dados,

podemos ficar tranquilos que a inclusão de 10 variáveis previsoras não sobreajusta o modelo, pois o número de vezes que as 10 variáveis previsoras foram mantidas é maior do que o número de vezes que alguma foi eliminada. Caso isso não ocorresse, a solução seria buscar mais dados ou diminuir o número de variáveis previsoras do modelo. Aquelas selecionadas por último em step forward são as melhores candidatas à exclusão. Vale mencionar que a função `validate()` também pode ser aplicada a modelos lineares (ver Levshina, 2015, cap.7).

Sua análise de regressão logística ainda não terminou! Assim como nos modelos lineares, é importante checar se os pressupostos da regressão logística foram atendidos. Um deles é que a relação entre as estimativas e as variáveis previsoras numéricas é linear. Aqui, temos apenas uma variável previsora numérica, INDICE.SOCIO. Para fazer este teste, vamos aplicar a função `crPlot()` do pacote car, que toma um modelo e a variável previsora como argumentos. Contudo, infelizmente, a função `crPlot()` não aceita modelos com interação. Façamos então um novo modelo, chamado `modelo2`, que contém a fórmula que vimos usando, mas sem interação.

```
modelo2 <- glm(VD ~
                 SEXO.GENERO +
                 FAIXA.ETARIA +
                 REGIAO +
                 INDICE.SOCIO +
                 CONT.FON.PREC +
                 CONT.FON.SEG +
                 TONICIDADE +
                 POSICAO.R +
                 CLASSE.MORFOLOGICA,
               data = dados, family = binomial)
```

Aplique agora a função `crPlot()` a `modelo2`, com o segundo argumento var = "INDICE.SOCIO".

```
crPlot(modelo2, var = "INDICE.SOCIO")
```

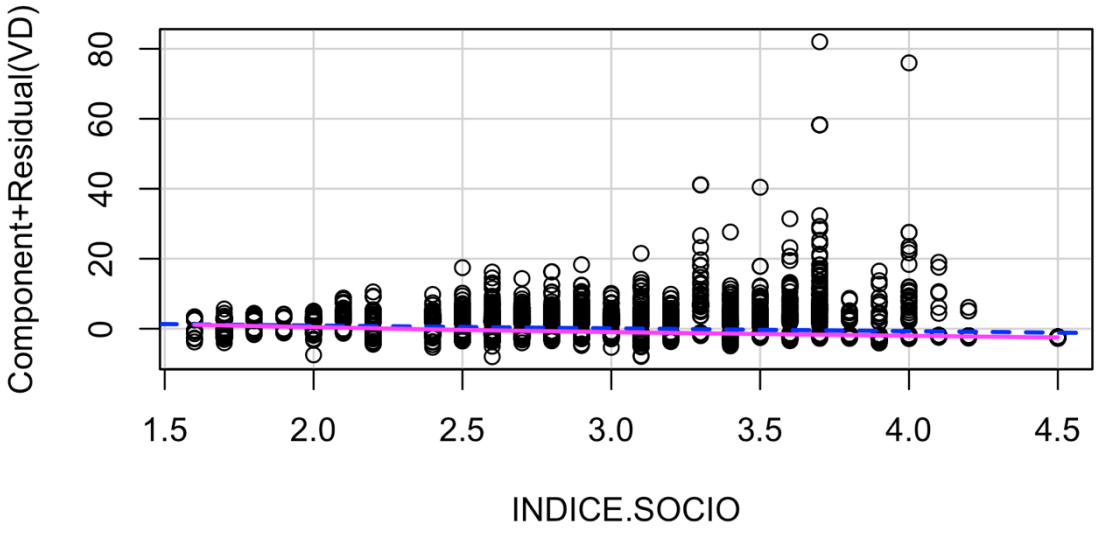

Figura 15.1: Plot dos valores previstos e observados de INDICE.SOCIO no modelo2. Fonte: própria.

Vemos que a linha pontilhada, de valores previstos, e a linha contínua, de valores observados, são praticamente coincidentes. Se não fossem, deveríamos considerar a possibilidade de retirar valores atípicos da distribuição (ver Lição 13).

Outro pressuposto de modelos de regressão logística é a inexistência de multicolinearidade, que pode ser testada pela função `vif()`. Digite então `car::vif(modelo)`. (Aqui, usamos a notação pacote::funcao pois a função `vif()` também existe no pacote rms, mas queremos usar a do pacote car.)

```
car::vif(modelo)

##                          GVIF Df GVIF^(1/(2*Df))
## SEXO.GENERO          1.032890  1        1.016312
## FAIXA.ETARIA         6.564032  2        1.600636
## REGIAO               2.806091  1        1.675139
## INDICE.SOCIO         1.076890  1        1.037733
## CONT.FON.PREC        3.985509  6        1.122123
## CONT.FON.SEG         2.028554  3        1.125117
## TONICIDADE           1.828523  1        1.352229
## POSICAO.R            2.095435  1        1.447562
## CLASSE.MORFOLOGICA   3.408764  5        1.130472
## FAIXA.ETARIA:REGIAO 10.667029  2        1.807219
```

Vemos aí que a maior parte dos valores estão bem abaixo de 5, o que indicia não colinearidade entre as variáveis do modelo. Os valores mais altos são de FAIXA.ETARIA e a interação FAIXA.ETARIA:REGIAO; aqui, é esperado que haja falta de colinearidade, já que a interação envolve a variável. A terceira coluna, de GVIF-ajustado, leva isso em conta e corrige a medida.

Por fim, deve-se checar se as observações do conjunto de dados são independentes umas das outras. Como já se notou na Lição 13, isso raramente é o caso de análises linguísticas, de modo que é sempre recomendável realizar *análises de efeitos mistos*, com a inclusão de efeitos aleatórios. Nos dados de /r/ em coda, as variáveis PARTICIPANTE e ITEM.LEXICAL representam efeitos aleatórios.

Vimos que modelos de efeitos mistos de regressão linear são criados por meio da função `lmer()`, por oposição à função `lm()`. Em modelos de efeitos mistos de regressão logística, a função é... `glmer()`! Lembre-se que efeitos aleatórios entram na fórmula com a notação (1|varaleatoria). Rode então a linha de comando a seguir. Uma nota: não estranhe se esse modelo demorar 3-4 minutos para rodar!

```
mod.glmer <- glmer(VD ~
                     SEXO.GENERO +
                     FAIXA.ETARIA * REGIAO +
                     INDICE.SOCIO +
                     CONT.FON.PREC +
                     CONT.FON.SEG +
                     TONICIDADE +
                     POSICAO.R +
                     CLASSE.MORFOLOGICA +
                     (1|PARTICIPANTE) +
                     (1|ITEM.LEXICAL),
                   data = dados, family = binomial)
```

Visualize agora o resultado com `summary(mod.glmer)`.

```
summary(mod.glmer)

## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula:
## VD ~ SEXO.GENERO + FAIXA.ETARIA * REGIAO + INDICE.SOCIO + CONT.FON.
PREC +
##     CONT.FON.SEG + TONICIDADE + POSICAO.R + CLASSE.MORFOLOGICA +
##     (1 | PARTICIPANTE) + (1 | ITEM.LEXICAL)
##    Data: dados
```

```
##
##      AIC      BIC   logLik deviance df.resid
##   7507.5   7692.9  -3727.8   7455.5     9200
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -7.8597 -0.4393 -0.2085  0.3127 11.0416
##
## Random effects:
##  Groups       Name        Variance Std.Dev.
##  ITEM.LEXICAL (Intercept) 0.4814   0.6939
##  PARTICIPANTE (Intercept) 2.0592   1.4350
## Number of obs: 9226, groups:  ITEM.LEXICAL, 1151; PARTICIPANTE, 118
##
## Fixed effects:
##                                 Estimate Std. Error z value Pr(>|z
|)
## (Intercept)                      0.439246   0.925710   0.474  0.635
15
## SEXO.GENEROmasculino             0.790969   0.275991   2.866  0.004
16
## FAIXA.ETARIA2a                   0.336203   0.505514   0.665  0.506
00
## FAIXA.ETARIA3a                  -0.126504   0.497683  -0.254  0.799
35
## REGIAOperiferica                 2.005981   0.496544   4.040 5.35e-
05
## INDICE.SOCIO                    -1.117505   0.237855  -4.698 2.62e-
06
## CONT.FON.PRECe                   0.284838   0.273786   1.040  0.298
17
## CONT.FON.PREC3                   0.589093   0.320176   1.840  0.065
78
## CONT.FON.PRECa                   0.661105   0.275211   2.402  0.016
30
## CONT.FON.PREC0                   0.641808   0.312345   2.055  0.039
90
## CONT.FON.PRECo                  -0.331198   0.280419  -1.181  0.237
57
## CONT.FON.PRECu                  -1.418130   0.338658  -4.188 2.82e-
05
## CONT.FON.SEGcoronal              0.659775   0.161881   4.076 4.59e-
05
## CONT.FON.SEGdorsal               0.008341   0.189613   0.044  0.964
91
## CONT.FON.SEGlabial               0.315251   0.177475   1.776  0.075
68
## TONICIDADEtonica                 0.558847   0.128725   4.341 1.42e-
05
## POSICAO.Rmedial                 -0.689526   0.162930  -4.232 2.32e-
05
## CLASSE.MORFOLOGICAadverbio       0.850362   0.486957   1.746  0.080
76
## CLASSE.MORFOLOGICAconj.prep      0.719349   0.317131   2.268  0.023
```

```
31
## CLASSE.MORFOLOGICAmorf.inf      -0.792194   0.262457  -3.018  0.002
54
## CLASSE.MORFOLOGICAsubstantivo    0.247449   0.134095   1.845  0.064
99
## CLASSE.MORFOLOGICAverbo          0.800943   0.161455   4.961 7.02e-
07
## FAIXA.ETARIA2a:REGIAOperiferica -1.128955   0.680637  -1.659  0.097
18
## FAIXA.ETARIA3a:REGIAOperiferica -1.258274   0.681851  -1.845  0.064
98
##
## (Intercept)
## SEXO.GENEROmasculino            **
## FAIXA.ETARIA2a
## FAIXA.ETARIA3a
## REGIAOperiferica                ***
## INDICE.SOCIO                    ***
## CONT.FON.PRECe
## CONT.FON.PREC3                  .
## CONT.FON.PRECa                  *
## CONT.FON.PREC0                  *
## CONT.FON.PRECo
## CONT.FON.PRECu                  ***
## CONT.FON.SEGcoronal             ***
## CONT.FON.SEGdorsal
## CONT.FON.SEGlabial              .
## TONICIDADEtonica                ***
## POSICAO.Rmedial                 ***
## CLASSE.MORFOLOGICAadverbio      .
## CLASSE.MORFOLOGICAconj.prep     *
## CLASSE.MORFOLOGICAmorf.inf      **
## CLASSE.MORFOLOGICAsubstantivo   .
## CLASSE.MORFOLOGICAverbo         ***
## FAIXA.ETARIA2a:REGIAOperiferica .
## FAIXA.ETARIA3a:REGIAOperiferica .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## optimizer (Nelder_Mead) convergence code: 0 (OK)
## Model failed to converge with max|grad| = 0.0387909 (tol = 0.002, c
omponent 1)
```

No modelo de efeitos mistos, quais variáveis deixam de ser significativamente correlacionadas com VD?

- CONT.FON.PREC e CONT.FON.SEG
- TONICIDADE e POSICAO.R
- CLASSE.MORFOLOGICA e CONT.FON.PREC
- FAIXA.ETARIA e a interação com REGIAO

Como já argumentado várias vezes neste curso, a interpretação de resultados numéricos é sempre mais fácil por meio de gráficos. Faça um gráfico de efeitos com `plot(allEffects(mod.glmer), type = "response")`.

```
plot(allEffects(mod.glmer), type = "response")
```

N.B.: Resultado aqui omitido.

Também se pode adicionar o argumento ask = T para selecionar manualmente quais efeitos você quer visualizar. A partir da linha de comando acima, adicione o novo argumento. Clique sobre a variável ou variáveis de interesse e clique sobre "cancelar" quando não quiser incluir mais nenhuma.

```
plot(allEffects(mod.glmer), type = "response", ask = T)
```

N.B.: Resultado aqui omitido.

Deixei disponível no Anexo D uma sugestão de roteiro de análise para variáveis nominais binárias, tomando o conjunto de dados da realização do /r/ em coda como exemplo. Assim como no roteiro da Lição 13, a ideia é sistematizar os passos de uma análise que ficaram espalhados por várias lições deste curso. Para consultar como os resultados de modelos de regressão logística podem ser reportados, reveja o final da Lição 13, sobretudo a Figura 13.5.

E isso conclui a Lição 15 e este curso. Mas há muito mais a se aprender! Espero que este curso tenha sido apenas o ponto de partida!

Para saber mais

Para saber mais sobre regressão logística e outros modelos aplicáveis a variáveis nominais, veja os capítulos 12 e 13 de Levshina (2015) e o capítulo 5 de Gries (2019).

Exercícios

Nesta lista, vamos trabalhar sobre os dados da realização de /r/ em lojas de Departamento em Nova Iorque, de Labov. Primeiro, carregue os dados da planilha LabovDS.csv. Após carregar o dataframe, cheque sua estrutura, como de praxe. Exclua os dados de d da variável r. Certifique-se de que o nível de referência é r1 – para que os

resultados de regressão logística sejam lidos em referência ao apagamento de /r/ (r0). Reorganize os níveis das variáveis store, word e emphasis em ordem alfabética.

Essa planilha contém os dados da realização de /r/ (r0 – apagado; r1 – realizado) por parte de funcionários de três lojas de departamento em Nova Iorque. As variáveis previsoras são store (Klein, Macy's, Saks), emphasis (casual ou emphasic) e word (fourth ou floor). Verifique se há interações entre essas três variáveis em modelos simples de regressão logística, que incluem apenas uma interação.

1. Entre qual par de variáveis há interação?
    a. nenhum
    b. emphasis e word
    c. store e emphasis
    d. store e word
2. Crie um modelo com todas as variáveis. Inclua a interação, se houver. Qual variável não apresenta correlação significativa com a pronúncia de /r/?
    a. emphasis
    b. store
    c. word
    d. todas têm correlação significativa
3. Qual é o índice C deste modelo?
4. Qual é o valor de R^2 deste modelo?
5. Este modelo, como um todo, é significativo? Justifique sua resposta.
6. Os funcionários de qual loja mais favorecem o apagamento de /r/?
    a. Klein
    b. Macy's
    c. Saks
7. Qual item lexical mais favorece o apagamento de /r/?
    a. fourth
    b. floor

8. Calcule a probabilidade (de 0% a 100%) de se realizar o apagamento de /r/ na palavra fourth. Reporte o resultado em número decimal.
9. Acima se testou se há interações. Teste se há colinearidade entre as variáveis previsoras.
    a. não há
    b. há entre emphasis e word
    c. há entre store e emphasis
    d. há entre store e word
10. Neste conjunto de dados, que tipo de variável é word?
    a. efeito fixo
    b. efeito aleatório

Referências

BAAYEN, R. Harald. *Analysing linguistic data*: a practical introduction to Statistics. Cambridge: Cambridge University Press, 2008.

BARBOSA, Plinio A.; MADUREIRA, Sandra. *Manual de Fonética acústica experimental.* Aplicações a dados do português. São Paulo: Cortez, 2015.

CHANG, Winston. *R Graphics Cookbook*. Sebastopol: O'Reilley, 2013.

CLEMENTS, G. N.; HUME, Elizabeth V. The internal organization of speech sounds. In: Goldsmith, John A. (Ed.), *The handbook of phonological theory*. Oxford: Blackwell, 1995, p. 245–306.

CRAWLEY, Michael J. *The R Book*. West Sussex: Wiley, 2013.

DALGAARD, Peter. *Introductory statistics with R*. New York: Springer, 2008.

FOWLER, Joy. *The social stratification of /r/ in New York City departament stores, 24 years after Labov*. Ms, 1986.

GIGERENZER, Gerd. Mindless statistics. *Journal of Socio-Economics*, vol. 33, 587–606, 2004.

GRIES, Stefan Th. *Estatística com R para a linguística.* Belo Horizonte: FALE/UFMG, 2019. Disponível em http://www.letras.ufmg.br/site/e-livros/Estat%C3%ADstica_com_R_Gries_%20Mello_et%20al.pdf. Último acesso em 2 dez. 2021.

GUY, Gregory; FURUKAWA, Alex; HACKNEY, Sarah; MAHBOOB, Rubaiyat; MANDEL, Ethan; WADDY, Kylie. Replicating Labov's study of (r) in New York City department stores, 2008. Ms.

LABOV, William. The intersection of sex and social class in the course of linguistic change. *Language Variation and Change*, vol. 2, 205–254, 1990.

LABOV, William. A estratificação social do (r) nas lojas de departamentos na cidade de Nova York. In: *Padrões sociolinguísticos*. São Paulo: Parábola, 2008, p. 63–90.

LEVSHINA, Natalia. *How to do Linguistics with R*. Data exploration and statistical analysis. Amsterdam: John Benjamins, 2015.

LOBANOV, B. M. Classification of Russian vowels spoken by different speakers. *The Journal of the Acoustical Society of America*, vol. 49(2), 606–608, 1971.

OUSHIRO, Livia. *Identidade na pluralidade*: avaliação, produção e percepção linguística na cidade de São Paulo. Tese de Doutorado. São Paulo: FFLCH-USP, 2015. 390f.

OUSHIRO, Livia. Social and structural constraints in lectal cohesion. *Lingua*, vol. 172–3, 116–130, 2016.

OUSHIRO, Livia. A computational approach for modeling the indexical field. *Revista de Estudos da Linguagem*, vol. 27(4), 1737–1786, 2019.

WALKER, Abby; GARCÍA, Christina; CORTÉS, Yomi; CAMPBELL-KIBLER, Kathryn. Comparing social meanings across listener and speaker groups: the indexical field of Spanish /s/. *Language Variation and Change* 25, 169–189, 2014.

WICKHAM, Hadley. Tidy data. *Journal of statistical software*, vol. 59(10), 1–23, 2014.

Pacotes

ADLER, Daniel; MURDOCH, Duncan. rgl: 3D Visualization Using OpenGL, 2021. R package version 0.108.3. Disponível em https://CRAN.R-project.org/package=rgl

AUGUIE, Baptiste. gridExtra: Miscellaneous Functions for "Grid"Graphics, 2017. R package version 2.3. Disponível em https://CRAN.R-project.org/package=gridExtra

BAAYEN, R. H.; SHAFAEI-BAJESTAN, Elnaz. languageR: Analyzing Linguistic Data: A Practical Introduction to Statistics, 2019. R package version 1.5.0. Disponível em https://CRAN.R-project.org/package=languageR

BARTÓN, Kamil. MuMIn: Multi-Model Inference, 2020. R package version 1.43.17. Disponível em https://CRAN.R-project.org/package=MuMIn

BATES, Douglas; MAECHLER, Martin, BOLKER, Ben; WALKER, Steven. lme4: Linear Mixed-Effects Models using Eigen and S4, 2021. R package version 1.1-27.1. Disponível em https://github.com/lme4/lme4/

FOX, John; WEISBERG, Sanford; PRICE, Brad. car: Companion to Applied Regression, 2021. R package version 3.0-11. Disponível em https://CRAN.R-project.org/package=car

FOX, John; WEISBERG, Sanford; PRICE, Brad; FRIENDLY, Michael; HONG, Jangman. effects: Effect Displays for Linear, Generalized Linear, and Other Models, 2020. R package version 4.2-0. Disponível em https://CRAN.R-project.org/package=effects

HARRELL JR., Frank E. rms: Regression Modeling Strategies, 2021. R package version 6.2-0. Disponível em https://CRAN.R-project.org/package=rms

HOTHORN, Torsten; BRETZ, Frank; WESTFALL, Peter. multcomp: Simultaneous Inference in General Parametric Models, 2021. R package version 1.4-17. Disponível em https://CRAN.R-project.org/package=multcomp

KROSS, Sean; CARCHEDI, Nick; BAUER, Bill; GRDINA, Gina. swirl: Learn R, in R, 2020. R package version 2.4.5. Disponível em http://swirlstats.com

KUZNETSOVA, Alexandra; BRUUN BROCKHOFF, Per; HAUBO BOJESEN CHRISTENSEN, Rune. lmerTest: Tests in Linear Mixed Effects Models, 2020. R package version 3.1-3. Disponível em https://github.com/runehaubo/lmerTestR

LESNOFF, Matthieu; LANCELOT, Renaud. aod: Analysis of Overdispersed Data, 2019. R package version 1.3.1. Disponível em https://cran.r-project.org/package=aod

LÜDECKE, Daniel. sjPlot: Data Visualization for Statistics in Social Science, 2021. R package version 2.8.10. Disponível em https://strengejacke.github.io/sjPlot/

NEUWIRTH, Erich. RColorBrewer: ColorBrewer Palettes, 2014. R package version 1.1-2. Disponível em https://CRAN.R-project.org/package=RColorBrewer

PENA, Edsel A.; SLATE, Elizabeth H. gvlma: Global Validation of Linear Models Assumptions, 2019. R package version 1.0.0.3. Disponível em https://CRAN.R-project.org/package=gvlma

SCHLOERKE, Barret; COOK, Di; LARMARANGE, Joseph; BRIATTE, Francois; MARBACH, Moritz; THOEN, Edwin; ELBERG, Amos; CROWLEY, Jason. GGally: Extension to ggplot2, 2021. R package version 2.1.2. Disponível em https://CRAN.R-project.org/package=GGally

URBANECK, Simon. base64enc: Tools for base64 encoding, 2015. R package version 0.1-3. Disponível em http://www.rforge.net/base64enc

WICKHAM, Hadley. tidyverse: Easily Install and Load the Tidyverse, 2021. R package version 1.3.1. Disponível em https://CRAN.R-project.org/package=tidyverse

Referências úteis

Existem muitos manuais sobre Estatística e sobre R, a grande maioria em inglês. A obra de Gries (2019) foi traduzida para o português em projeto coordenado por Heliana Mello. Em língua inglesa, recomendo consultar Baayen (2008) e Levshina (2015), que tratam de análises linguísticas. Dalgaard (2008) é um manual bastante didático mas, diferentemente das três obras acima, não voltado para a área de Linguística. Uma referência bastante completa sobre a linguagem R se encontra em Crawley (2013). Para a feitura de gráficos, Chang (2013) é referência indispensável.

Recomendo também a consulta a vários materiais de qualidade e grupos de discussão que podem ser encontrados na Internet:

- **R para Linguistas**: https://www.facebook.com/groups/RparaLinguistas/
- **Bookdown**: https://bookdown.org/
- **Tutorial de ggplot2** de Mahayana Godoy: https://github.com/mahayanag/rworkshop
- **Quantitative Language Data Analysis in R**, de Adriana Picoral Scheidegger: https://picoral.github.io/variable-rule-in-R/
- **Introdução a modelos de regressão em R** (Ronaldo Lima Jr, Guilherme Garcia e Bernhard Angele): https://www.youtube.com/playlist?list=PL3Qku9eEGkK1TF274nuIva85i4RaeIvOw
- Betsy Sneller's **Introduction to Data Analysis with R**: https://betsysneller.github.io/RCourse/RCourse.html
- Joe Fruehwald's **R Study Group**: http://www.ling.upenn.edu/~joseff/rstudy/
- **swirl**: http://swirlstats.com/

Links para *scripts* das lições

Lição 1: https://github.com/oushiro/IELv2.0/blob/main/Licao01-IEL.R

Lição 2: https://github.com/oushiro/IELv2.0/blob/main/Licao02-IEL.R

Lição 3: https://github.com/oushiro/IELv2.0/blob/main/Licao03-IEL.R

Lição 4: https://github.com/oushiro/IELv2.0/blob/main/Licao04-IEL.R

Lição 5: https://github.com/oushiro/IELv2.0/blob/main/Licao05-IEL.R

Lição 6: https://github.com/oushiro/IELv2.0/blob/main/Licao06-IEL.R

Lição 7: https://github.com/oushiro/IELv2.0/blob/main/Licao07-IEL.R

Lição 8: https://github.com/oushiro/IELv2.0/blob/main/Licao08-IEL.R

Lição 9: https://github.com/oushiro/IELv2.0/blob/main/Licao09-IEL.R

Lição 10: https://github.com/oushiro/IELv2.0/blob/main/Licao10-IEL.R

Lição 11: https://github.com/oushiro/IELv2.0/blob/main/Licao11-IEL.R

Lição 12: https://github.com/oushiro/IELv2.0/blob/main/Licao12-IEL.R

Lição 13: https://github.com/oushiro/IELv2.0/blob/main/Licao13-IEL.R

Lição 14: https://github.com/oushiro/IELv2.0/blob/main/Licao14-IEL.R

Lição 15: https://github.com/oushiro/IELv2.0/blob/main/Licao15-IEL.R

Posfácio

Ronaldo Lima Jr. (UFC)

Sinto-me extremamente honrado em redigir estas palavras finais para o livro da Livia Oushiro. Seu trabalho teve bastante impacto no meu, e, por isso, sou muito grato às suas empreitadas em prol da capacitação de linguistas em análise quantitativa de dados.

Conheci a Livia Oushiro e seu material "Introdução à Estatística para Linguistas" em 2017, por meio do grupo de Facebook que ela havia criado, "R para Linguistas". A data é memorável, pois, no segundo semestre daquele ano, ministrei pela primeira vez parte de uma disciplina sobre análise quantitativa de dados no Programa de Pós-Graduação em Linguística (PPGL) da minha universidade, a Universidade Federal do Ceará (UFC), e a versão em swirl do material, em que é possível fazer as aulas de forma interativa dentro do próprio R, caiu como uma luva. Ao consultar a Livia se poderia usar seu material com meus alunos na disciplina, percebi que estava diante de uma professora genuína, com a generosidade acadêmica que tanto admiro e da qual a academia carece. Eu fiz todas as aulas em swirl para ver como ela tinha organizado o curso, e a forma dialogada de suas aulas me chamou a atenção. Acabei planejando minha parte da disciplina seguindo a mesma sequência proposta pela Livia em seu material, utilizei diversos dos seus dados e exemplos ao longo das aulas, e as tarefas de casa dos meus alunos eram o caderno de exercícios criado por ela também em swirl. O programa da recém-criada disciplina "Análise Quantitativa de Dados em Linguística" do PPGL da UFC ainda contém muito dessa estrutura inicial. Além disso, sempre que ministro algum curso ou palestra sobre assunto[10], recomendo os materiais da Livia, e agora será um prazer recomendá-lo como livro publicado pela Editora da Abralin.

[10] Disponíveis em https://ronaldolimajr.github.io.

A importância da análise quantitativa de dados na Linguística

Se você chegou até o posfácio desta obra, provavelmente não precisa de muito convencimento sobre a importância de análises estatísticas na ciência como um todo, e, consequentemente, na Linguística. Mesmo assim, gostaria de registrar que acredito que o papel de uma capacitação em análise quantitativa de dados vai além de sua aplicação direta a dados quantitativos. Há subáreas da Linguística que exigem análises estatísticas mais do que outras (como a Fonética, a Psicolinguística, a Sociolinguística e, mais recentemente, a Linguística de Corpus), mas acredito que mesmo pesquisadores de subáreas que conduzem estudos exclusivamente qualitativos podem se beneficiar do conhecimento em análise quantitativa de dados. Isso porque o raciocínio sobre desenho experimental como um todo pode auxiliar no estabelecimento de objetivos e perguntas de pesquisa coerentes, em que a operacionabilidade do estudo é posta à prova pelo próprio pesquisador ao considerar questões como correlações espúrias, variáveis de confusão (*confounding variables*), limitações da amostra, e inferências sobre as relações de causalidade (que são de responsabilidade exclusiva do pesquisador, não informadas pelos dados observados).

Um exemplo: um alienígena observando um humano abrindo os olhos exatamente quando um despertador toca todas as manhãs poderia se questionar se os olhos abrindo estão fazendo o relógio despertar, ou se o despertador tocando está fazendo os olhos abrirem. Este é um questionamento que nunca faríamos, mas que deveríamos praticar fazer principalmente ao elaborar as questões de pesquisa, de maneira a torná-las operacionalizáveis. Humanos sabem que é o despertador que causa os olhos abrirem, que é um acidente de trânsito que causa o congestionamento (e raramente o contrário), e que estar gripado nos faz tossir (e não tossir nos causa gripe). Contudo, precisamos nos lembrar que desconhecemos as relações de causa do nosso objeto de pesquisa (por isso o pesquisamos), e que, sendo assim, precisamos de cautela nas inferências. Parece ser muito lógico, por exemplo, ao observar que aprendizes motivados

apresentam maior desenvolvimento em uma língua estrangeira, concluir que a motivação causa aprendizagem. Contudo, o raciocínio estatístico leva o pesquisador a questionar questões como: mas e os alunos motivados que não desenvolvem tanto a L2? E os desmotivados que aprendem a L2 mesmo assim? Perceber o desenvolvimento de sucesso na nova língua não aumenta também a motivação? Não haveria outros fatores que poderiam estar causando tanto a motivação como o maior desenvolvimento linguístico? Ou seja, mesmo diante de dados qualitativos (se fosse o caso aqui), ou mesmo antes de se coletar os dados, um pesquisador com conhecimento e pensamento estatístico é capaz de levantar questões sobre a metodologia de sua própria pesquisa que podem torná-la mais robusta e, consequentemente, com inferências mais confiáveis e replicáveis.

Dada essa importância do pensamento quantitativo na análise de dados linguísticos, é inquestionável o importante papel deste livro para a Linguística. Concluo, portanto, meu posfácio com breves sugestões para pesquisadores em diferentes níveis de proficiência em análise quantitativa de dados. Como iniciantes, penso em absolutamente qualquer pessoa que tomou interesse pelo assunto e que tenha apenas vagas lembranças da matemática aprendida na escola (público-alvo potencial deste livro). Como intermediário, me refiro a alguém que chegou até a Lição 11 deste livro com segurança e que talvez ainda esteja firmando seus conhecimentos das demais lições. Avançado aqui se refere aos que terminarem as Lições 12–15 com bastante familiaridade e confiança.

Para os iniciantes: meu testemunho sobre os primeiros passos em estatística

Alguns estudantes e colegas linguistas que estão buscando seus primeiros passos em análise quantitativa de dados se identificam com o início da minha trajetória. Por isso, aproveito para registrá-la como forma de motivação e encorajamento, juntamente com algumas sugestões para quem está no início da caminhada em análise estatística.

Logo no início do meu doutorado me dei conta que precisaria de alguma análise estatística dos dados que estava planejando coletar e analisar – dados acústicos de produção de fala de alunos de inglês como L2. Lendo outros trabalhos da área, descobri que precisaria de um tal de teste-t. Comprei, então, o livro "*A Guide to Doing Statistics in Second Language Research Using SPSS*" (LARSON-HALL, 2010), já que o título parecia perfeito para a minha pesquisa, e, com pressa, fui direto ao capítulo sobre teste-t. Resultado: não entendi nada! Percebi que precisaria ler desde o início para saber minimamente o que eram "variáveis", "amostra" e outros conceitos básicos. Sabia, por exemplo, que um valor de *p* menor que 0,05 indicaria uma diferença significativa entre meus grupos de aprendizes, mas não fazia ideia do que realmente ele representava (hoje percebo, que perigo!). Li o livro do início e confesso que compreendi apenas uns 50%. Passei, então, a fazer cursos on-line e a assistir videoaulas, e, quando me senti com mais base, reli o livro e experimentei um momento "aha!", quando tudo passou a fazer sentido!

Não acho que essa deva ser a rota de todos, e nem acho que o livro que utilizei seja o melhor para começar, mas registro aqui minha experiência inicial para mostrar que percebi que o conhecimento em análise quantitativa de dados é cumulativo e normalmente requer um estudo em ciclos de idas e vindas. Algumas lições que tenho aprendido na minha jornada em análise quantitativa de dados e que gostaria de deixar para quem está começando são:

- Não tem como fugir do básico! Antes de conduzir sua análise, mesmo sabendo qual será e qual valor você espera como resultado, é preciso compreender conceitos como amostra e população, (tipos de) variáveis, tamanho de efeito, intervalo de confiança, e compreender, minimamente, como se chega ao valor que você vai reportar (como o valor de *p*);
- Nem sempre compreendemos um conceito de primeira, mas isso não é motivo para desistir, e vale a pena insistir. Dê um tempo e depois volte ao conceito – procure por pessoas diferentes, em vídeos e páginas eletrônicas, explicando o mesmo conceito de maneiras diferentes, com outros exemplos;

- Às vezes um conceito que ficou muito claro em um momento passa a ficar menos claro com o tempo – volte ao conceito e busque por exemplos para reativar sua memória;
- Persista e capacite-se para realizar suas próprias análises, sem contar com programas automáticos ou com pessoas contratadas para isso, pois conduzir suas próprias análises trará uma compreensão muito maior sobre os seus dados e sobre os fenômenos linguísticos sob investigação.

Aos poucos a análise quantitativa de dados passou a fazer parte dos meus interesses acadêmicos, e, por isso, continuo estudando e colocando esses passos em prática. Em muitos momentos a curva de aprendizagem é bastante íngrime, mas compensa persistir e retornar aos trechos desafiadores.

Para os intermediários: cautelas na análise quantitativa de dados

É importante manter em mente que, ao mesmo tempo que a análise estatística dos dados é absolutamente necessária em vários tipos de estudos, ela é apenas um instrumento, uma ferramenta. Sendo assim, quanto maior o domínio da ferramenta, maior será a compreensão de suas atribuições, capacidades e limitações. Por um lado, pesquisadores iniciantes em análises quantitativas podem inadvertidamente propor inferências e conclusões de maneira muito incisiva e categórica com base em uma análise frágil. Um valor de p abaixo de 0,05, por exemplo, nem sempre é suficiente para se determinar um efeito de maneira contundente. Por outro lado, pesquisadores com bastante bagagem estatística podem esquecer-se de que suas pesquisas em Linguística não devem ter como fim a análise, e que um retorno à teoria e às contribuições para o conhecimento de Linguística devem ser sempre priorizados. Sendo assim, deixo aqui duas recomendações para quem já domina testes de hipótese ou até mesmo rode alguns modelos de regressão:

Não dependa exclusivamente do valor de p

Como a Livia Oushiro deixa claro ao longo do livro, há diversas limitações numa interpretação de análise estatística exclusivamente dependente do resultado do valor de *p*.

Lembre-se que o valor de *p* é apenas a probabilidade dos seus dados diante da hipótese nula e que, por isso, ele:

- não diz nada sobre sua hipótese de trabalho (a hipótese alternativa);
- não diz nada sobre o tamanho do efeito;
- não tem gradiência (e, por isso, valores próximos ao limiar estabelecido não estão "aproximando significância");
- tem um limiar (normalmente 5% na Linguística) que é arbitrário.

Veja, por exemplo, que o pacote lme4 do R não dá valores de *p* em seus resultados de modelos com efeitos mistos[11], deixando a interpretação para ser feita com base nos coeficientes e nos erros-padrão (que permitem calcular os intervalos de confiança).

Um dos problemas com a prática de se tirar conclusões baseadas exclusivamente em valores de *p* é que, muitas vezes, é possível ter um valor de *p* baixo em um estudo com tamanho de efeito muito pequeno e/ou com baixo poder estatístico[12]. Além disso, como Lima Jr. e Garcia (2021) demonstram, diferentes análises podem levar a resultados categoricamente diferentes se baseados apenas em valores de *p*. Por fim, o desejo por um valor de *p* baixo pode levar, mesmo que inadvertidamente, no caso de pesquisadores iniciantes, à prática antiética de *p-hacking*, em que pequenos ajustes nos dados ou decisões de técnicas diferentes de análise podem ser aplicados para abaixar o valor de *p*.

Por isso, a sugestão é de, inicialmente, utilizar o valor de *p* juntamente com as demais informações de testes e modelos estatísticos para se chegar às inferências, em especial o tamanho do efeito, os intervalos de confiança, o tamanho da amostra, e o poder

[11] Neste caso é por não haver uma única metodologia para o cálculo do valor de *p* nesses modelos, com diferentes aproximações sendo utilizadas por diferentes pacotes, como o lmerTest e o sjPlot.

[12] A chance de detectar o efeito de um teste quando de fato houver um efeito.

estatístico. Com o tempo, conforme consta na próxima seção, a ideia é abandonar por completo o valor de *p* das análises inferenciais.

Cuidado com práticas antiéticas

Existe outra prática antiética adotada por pesquisadores muitas vezes por falta de conhecimento: o *HARKing – Hypothesizing After Results are Known* (Hipotetizar depois que os resultados são conhecidos). Umas das causas das crises de replicabilidade que têm afligido diversas áreas do conhecimento, recentemente a psicologia, é o estabelecimento de hipóteses depois que se veem alguns resultados. Essa prática aumenta as chances de Erro de Tipo I, pois são hipóteses que podem ser criadas com base em resultados espúrios, chegados ao acaso quando se testava outra hipótese.

A melhor maneira de se evitar essa prática é por meio da elaboração cuidadosa e meticulosa das hipóteses antes de se conduzir o estudo, de forma que mudanças nas hipóteses não ocorrerão ao longo da análise. Essas hipóteses podem ser registradas em ferramentas on-line criadas especificamente para isso (como o https://osf.io), e etapas de qualificação de projetos de pesquisa, no caso de estudantes de pós-graduação, também podem servir esse propósito.

Para os que estão prontos para avançar: próximos passos

Para os que já dominam os modelos ensinados pela Livia Oushiro nas Lições 12–15, minha sugestão é que o próximo passo seja no investimento de mais conhecimento sobre modelos de regressão, para que, aos poucos, passem a utilizar exclusivamente modelos em suas análises. Modelos de regressão com efeitos mistos[13] deveriam ser a prática padrão por qualquer pesquisador (MCELREATH, 2020). Portanto, saber rodar e interpretar modelos lineares simples e múltiplos, modelos logísticos, multinomiais, de

[13] Também chamados de modelos de efeitos variáveis, modelos hierárquicos, modelos multinível, modelos aninhados etc.

poisson e ordinal, com a inclusão de *intercepts* e *slopes* aleatórios sempre que necessário, é crucial. Explorar as possibilidades de *contrast coding* bem como de centralização e padronização das variáveis também é desejável. Para isso sugiro três materiais, em ordem crescente de complexidade: Garcia (2021), Sonderegger (2022)[14] e Gelman, Hill e Vehtari (2020).

Como passo seguinte, sugiro investir em inferência bayesiana. A estatística bayesiana tem diversas vantagens, sendo a primeira delas a própria definição de probabilidade. Em análises bayesianas, a probabilidade não é vista como a frequência de ocorrência de um fenômeno, mas como a expectativa de sua ocorrência[15]. Essa aparente pequena diferença fez com que estatísticos frequentistas não conseguissem calcular a probabilidade de um acidente quando usinas nucleares começaram a ser construídas nos EUA, já que não tinham observado nenhum acidente ainda; por isso, a *RAND Corporation* precisou utilizar métodos bayesianos para avaliar a probabilidade de acidentes nucleares antes de acontecer um (MCGRAYNE, 2011).

Outra vantagem é que, enquanto análises frequentistas medem a probabilidade dos dados diante de uma hipótese nula, que é a própria definição de valor de *p*, análises bayesianas calculam a probabilidade das hipóteses de trabalho (hipóteses alternativas) diante dos dados, fazendo a sua interpretação ser mais direta e intuitiva. Com isso, não há valores de *p* em análises bayesianas, e os resultados não são dados como *point estimates*, mas como distribuições de probabilidade das hipóteses de trabalho. Isso retira a decisão dicotômica entre ter ou não ter efeito, e traz nuance, incerteza e gradiência para as inferências, algo desejado quando se busca inferir parâmetros de uma população com base em uma amostra.

Por fim, modelos bayesianos permitem incluir conhecimento prévio da área ou expectativas de valores plausíveis das variáveis, por meio das distribuições *a priori*. Em

[14] A ser publicado pela MIT Press, disponível publicamente no momento da escrita deste texto em https://osf.io/pnumg/

[15] Por isso análises não bayesianas são comumente chamadas de frequentistas.

análises frequentistas, todos os valores dos parâmetros são igualmente prováveis *a priori*. Sendo assim, em uma análise de tempo de reação, um tempo de 10 milissegundos e um de 10 minutos são igualmente prováveis *a priori*; ou, em um estudo fonético, um valor de 100Hz e um de 5000Hz para F1 são igualmente prováveis, e sabemos que não deveriam ser. Em uma análise bayesiana, antes mesmo de se olhar para os dados, podemos informar o modelo sobre uma distribuição de probabilidade de tempos de reação plausíveis ou de valores de F1 plausíveis com base na literatura e em estudos prévios.

As recomendações que deixo para iniciar estudos de estatística bayesiana são um artigo introdutório de Lima Jr. e Garcia (2021), e os livros de McElreath (2020)[16] e Kruschke (2015).

Conclusão

Finalizo reforçando a alegria que sinto em escrever este posfácio, dada a relevância deste livro para a Linguística no Brasil. Tenho certeza que, assim como tem sido para os alunos da Livia e para os meus, este livro será útil para muitos que estão buscando seus primeiros passos para ter o controle das análises quantitativas de seus dados.

Referências

GARCIA, G. D. *Data visualization and analysis in second language research*. New York: Routledge, 2021.

GELMAN, A.; HILL, J.; VEHTARI, A. *Regression and other stories*. Cambridge: Cambridge University Press, 2020.

KRUSCHKE, J. *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. 2nd. ed. London: Academic Press, 2015.

[16] O Richard McElreath tem um canal no YouTube com videoaulas que acompanham os capítulos do seu livro.

LARSON-HALL, J. *A guide to doing statistics in second language research using SPSS and R*. New York: Routledge, 2010.

LIMA JR., R. M.; GARCIA, G. D. Diferentes análises estatísticas podem levar a conclusões categoricamente distintas. *Revista da ABRALIN*, v. 20, n. 1, p. 1–19, ago. 2021. Disponível em: https://revista.abralin.org/index.php/abralin/article/view/1790.

MCELREATH, R. *Statistical rethinking: A Bayesian course with examples in R and Stan*. 2nd. ed. Boca Raton: CRC press, 2020.

MCGRAYNE, S. B. *The theory that would not die*. New Haven: Yale University Press, 2011.

SONDEREGGER, M. *Regression modeling for linguistic data*. [S.l.]: Version 1.0. https://osf.io/pnumg/, 2022.

Anexos

Anexo A: Cálculo de r de Pearson e de r ajustado

Arquivo .R disponível em https://github.com/oushiro/IEL/blob/master/Licao11-demonstracoes.R.

```
###Cálculo do r de Pearson

idade <- c(1, 2, 3, 4, 5, 5, 5, 6, 7, 8, 8, 9, 11, 12, 12)
altura <- c(60, 65, 97, 98, 100, 105, 107, 105, 119, 122, 125, 132, 14
2, 147, 153)

#numero de observacoes
n <- 15
#soma dos valores da variavel x
SX <- sum(idade)
#soma dos valores da variavel y
SY <- sum(altura)
#soma dos valores da variavel x elevados ao quadrado
S.X2 <- sum(idade^2)
#soma dos valores da variavel y elevados ao quadro
S.Y2 <- sum(altura^2)
#soma de x * y
XY <- idade * altura
SXY <- sum(XY)
#r de Pearson
r <- ((n * SXY) - (SX*SY)) / sqrt(((n*S.X2)-(SX^2))*((n*S.Y2)-(SY^2)))
r

#comparar com cor.test:
cor.test(altura, idade)

###Calculo de R^2 ajustado:

#valor de R^2
R2 <- 0.943201
#numero de observacoes
n <- 15
#numero de variaveis independentes / predictors
k <- 1

#formula de R^2 ajustado
1 - (1-R2) * ((n-1)/(n-k-1))
```

Anexo B: Roteiro para análise de variáveis numéricas

Arquivo .R disponível em

https://github.com/oushiro/Introducao_a_Estatistica_para_Linguistas/blob/master/scripts/Licao13-IEL-roteiroAnalise-VRnumerica.R.

```
### Introdução à Estatística para Linguística ###
### L. Oushiro ###
### Roteiro para Análise de Variáveis Numéricas ###

### Preliminares: carregar pacotes e dados; ajustar dados ####
### Carregar pacotes necessários
library(tidyverse)
library(effects)
library(car)
library(lme4)
library(lmerTest)

### Definir diretório de trabalho ####
#setwd()

### Carregar dados ####
pretonicas <- read_csv("Pretonicas.csv",
                       col_types = cols(.default = col_factor(),
                                        VOGAL = col_factor(levels = c(
"i", "e", "a", "o", "u")),
                                        F1 = col_double(),
                                        F2 = col_double(),
                                        F1.NORM = col_double(),
                                        F2.NORM = col_double(),
                                        F1.SIL.SEG = col_double(),
                                        F2.SIL.SEG = col_double(),
                                        F1.SEG.NORM = col_double(),
                                        F2.SEG.NORM = col_double(),
                                        DIST.TONICA = col_double(),
                                        Begin.Time.s = col_double(),
                                        End.Time.s = col_double(),
                                        Duration.ms = col_double(),
                                        IDADE = col_integer(),
                                        IDADE.CHEGADA = col_integer(),
                                        ANOS.SP = col_integer()
                                        )
                       )

### Ajustar dados ####
pretonicas$CONT.PREC <-  fct_collapse(pretonicas$CONT.PREC,
                          dental.alveolar = c("t", "d", "n", "l"),
                          labial = c("p", "b", "m", "f", "v"),
                          palatal.sibilante = c("S", "Z", "L", "s", "z
"),
                          velar = c("k", "g"),
                          vibrante = c("h", "R")
```

```
)

pretonicas$CONT.PREC <- fct_relevel(pretonicas$CONT.PREC, "dental.alve
olar", "labial", "palatal.sibilante", "velar", "vibrante")

pretonicas$CONT.SEG <- fct_collapse(pretonicas$CONT.SEG,
                    dental.alveolar = c("t", "d", "n", "l"),
                    labial = c("p", "b", "m", "f", "v"),
                    palatal.sibilante = c("S", "Z", "L", "N", "s", "z
"),
                    velar = c("k", "g"),
                    vibrante = c("r", "h", "R")
)

pretonicas$CONT.SEG <- fct_relevel(pretonicas$CONT.SEG, "dental.alveol
ar", "labial", "palatal.sibilante", "velar", "vibrante")

### Subconjuntos de dados ####
VOGAL_e <- filter(pretonicas, VOGAL == "e") %>%
  droplevels()

PBSP_e <- filter(pretonicas, VOGAL == "e" & AMOSTRA == "PBSP")

SP2010_e <- filter(pretonicas, VOGAL == "e" & AMOSTRA == "SP2010")

### Checar dados
str(VOGAL_e)
#View(VOGAL_e)
str(PBSP_e)
str(SP2010_e)

### Análises descritivas e univariadas ####
#Cálculo de média, mediana, desvio padrão
pretonicas %>%
  group_by(VOGAL, AMOSTRA) %>%
  summarize(media_F1 = mean(F1.NORM),
            mediana_F1 = median(F1.NORM),
            sd_F1 = sd(F1.NORM))

# Espaços vocálicos de PBSP e SP2010
ggplot(medias, aes(x = media_F2, y = media_F1, color = AMOSTRA, label
= VOGAL)) +
  geom_line() +
  geom_label() +
  scale_x_reverse() +
  scale_y_reverse() +
  ggtitle("Valores médios de F1 e F2 normalizados nas amostras PBSP e
SP2010") +
  labs(x = "F2 normalizado", y = "F1 normalizado") +
  theme_bw()

### F1.NORM ~ AMOSTRA (vogal /e/)
#Teste-t
```

```
# Checar normalidade da distribuição com Teste de Shapiro
shapiro.test(PBSP_e$F1.NORM)
shapiro.test(SP2010_e$F1.NORM)
# t.test(F1.NORM ~ AMOSTRA, data = VOGAL_e)
wilcox.test(F1.NORM ~ AMOSTRA, data = VOGAL_e, conf.int = T)

### Outras vogais?... Outras variáveis previsoras?...

# Gráfico de dispersão das medições de F1 e F2 normalizados em PBSP e
SP2010
ggplot(pretonicas, aes(x = F2.NORM, y = F1.NORM, color = VOGAL)) +
  geom_point() +
  scale_x_reverse() +
  scale_y_reverse() +
  facet_grid(. ~ AMOSTRA) +
  ggtitle("Dispersão das medidas de F1 e F2 normalizados nas amostras
PBSP e SP2010") +
  labs(x = "F2 normalizado", y = "F1 normalizado") +
  theme_bw()

# Boxplots das medidas de F1.NORM por VOGAL e por AMOSTRA
ggplot(pretonicas, aes(x = AMOSTRA, y = F1.NORM, color = VOGAL)) +
  geom_boxplot(notch = TRUE) +
  scale_y_reverse() +
  labs(x = "Amostra", y = "F1 normalizado") +
  facet_grid(. ~ VOGAL) +
  theme_bw()

# Histograma das medições de F1.NORM das cinco vogais dos dados de PBS
P e SP2010
ggplot(pretonicas, aes(x = F1.NORM, fill = AMOSTRA)) +
  geom_histogram(binwidth = 10, position = "identity", alpha = 0.4) +
  labs(x = "F1 normalizado", y = "Frequência") +
  facet_grid(VOGAL ~ .) +
  theme_bw()

# Scatterplot de F1.NORM por F1.SEG.NORM
ggplot(VOGAL_e, aes(x = F1.SEG.NORM, y = F1.NORM)) +
  geom_point() +
  scale_y_reverse() +
  facet_grid(. ~ AMOSTRA) +
  geom_smooth(method = "lm", se = TRUE, color = "lightgrey")

#Teste de correlação de Pearson
# Checar normalidade das distribuições com Teste de Shapiro
shapiro.test(VOGAL_e$F1.NORM)
shapiro.test(VOGAL_e$F1.SEG.NORM)

cor.test(VOGAL_e$F1.NORM, VOGAL_e$F1.SEG.NORM, method = "spearman") #
para distribuição não normal - teste não paramétrico

#Modelo de regressão linear
mod <- lm(F1.NORM ~ F1.SEG.NORM, data = VOGAL_e)
```

```
summary(mod)
plot(effect("F1.SEG.NORM", mod), grid = T, ylim = c(460, 410))

### Análises multivariadas ####

### Modelo de regressão linear ####
mod <- lm(F1.NORM ~ AMOSTRA + SEXO + F1.SEG.NORM + CONT.PREC + CONT.SE
G, data = VOGAL_e2)
summary(mod)

### Função step() ####
m0 <- lm(F1.NORM ~ 1, data = VOGAL_e2)
m.fw <- step(m0, direction = "forward", scope = ~ AMOSTRA + SEXO + F1.
SEG.NORM + CONT.PREC + CONT.SEG)
m.fw

m.bw <- step(mod, direction = "backward")
m.bw

m.both <- step(m0, scope = ~ AMOSTRA + SEXO + F1.SEG.NORM + CONT.PREC
+ CONT.SEG)
m.both

### Função drop1() ####
drop1(mod, test = "F")

### Novo modelo linear sem variável SEXO
modelo <- lm(F1.NORM ~ F1.SEG.NORM + AMOSTRA + CONT.SEG + CONT.PREC, d
ata = VOGAL_e2)
summary(modelo)

### Checagem de pressupostos ####

### (a) relação entre variável resposta e variável previsora numérica
é linear?
# Aplicar função crPlot() (depende do pacote car)
crPlot(modelo, var = "F1.SEG.NORM") # valores atípicos F1.SEG.NORM > 5
00Hz

VOGAL_e3 <- filter(VOGAL_e2, F1.SEG.NORM < 500)

# Novo modelo linear
modelo2 <- lm(F1.NORM ~ AMOSTRA + F1.SEG.NORM + CONT.PREC + CONT.SEG,
data = VOGAL_e3)
summary(modelo2)
crPlot(modelo2, var = "F1.SEG.NORM")

### (b) Há multicolinearidade? (função vif() depende do pacote car)
vif(modelo2)

### (c) Resíduos têm distribuição normal?
shapiro.test(modelo2$residuals)
```

```
### (d) Observações são independentes? -- em dados linguísticos, quase
nunca são! --> MODELOS DE EFEITOS MISTOS

### Criar modelo linear de efeitos mistos ###
# função lmer() depende dos pacotes lme4 e lmerTest
mod1.lmer <- lmer(F1.NORM ~ AMOSTRA + F1.SEG.NORM + CONT.PREC + CONT.S
EG + (1|INFORMANTE) + (1|PALAVRA), data = VOGAL_e3)
summary(mod1.lmer)

# Função step backward
m.bw.lmer <- step(mod1.lmer, direction = "backward")
m.bw.lmer

### ~  Plotar estimativas do modelo: plot_model() ~ ####
#https://cran.r-project.org/web/packages/sjPlot/vignettes/plot_model_e
stimates.html (requer pacote sjPlot); somente para efeitos fixos
library(sjPlot)

plot_model(mod, transform = NULL, show.values = T, value.offset = .3)
```

Anexo C: Cálculo do valor de significância para modelo

Arquivo .R disponível em

https://github.com/oushiro/Introducao_a_Estatistica_para_Linguistas/blob/master/scripts/Licao14-IEL-calculoSignificanciaModelo.R.

```
## Cálculo do valor de significância para modelo logístico como um todo

# visualizar dados do modelo
modelo$null.deviance
modelo$deviance
modelo$df.null
modelo$df.residual

# Função pchisq faz teste de qui-quadrado e cálculo de valor-p a partir dos parâmetros acima
pchisq(modelo$null.deviance - modelo$deviance, modelo$df.null - modelo$df.residual, lower.tail=F)
```

Anexo D: Roteiro para análise de variáveis nominais binárias

Arquivo .R disponível em

https://github.com/oushiro/Introducao_a_Estatistica_para_Linguistas/blob/master/scripts/Licao15-IEL-roteiroAnalise-VRnominal.R.

```
### Introdução à Estatística para Linguística ###
### L. Oushiro ###
### Roteiro para Análise de Variáveis Nominais Binárias ###

### Preliminares: carregar pacotes e dados; ajustar dados ####
### Carregar pacotes necessários ####
library(rms)
library(effects)
library(car)
library(lme4)
library(lmerTest)

### Definir diretório de trabalho ####
# setwd()

### Carregar dados ####
dados <- read_csv("DadosRT.csv",
                  col_types = cols(.default = col_factor(),
                                   VD = col_factor(levels = c("tepe",
"retroflexo")),
                                   FAIXA.ETARIA = col_factor(levels =
c("1a", "2a", "3a")),
                                   ESCOLARIDADE = col_factor(levels =
c("fundamental", "medio", "superior")),
                                   REGIAO = col_factor(levels = c("cen
tral", "periferica")),
                                   CONT.FON.PREC = col_factor(levels =
c("i", "e", "3", "a", "0", "o", "u")),
                                   TONICIDADE = col_factor(levels = c(
"atona", "tonica")),
                                   POSICAO.R = col_factor(levels = c("
final", "medial")),
                                   CLASSE.MORFOLOGICA = col_factor(lev
els = c("adjetivo", "adverbio", "conj.prep", "morf.inf", "substantivo"
, "verbo")),
                                   IDADE = col_integer(),
                                   INDICE.SOCIO = col_double(),
                                   FREQUENCIA = col_double()
                  )
)

### Ajustar dados ####
dados$CONT.FON.SEG <- fct_collapse(dados$CONT.FON.SEG,
                                   pausa = "#",
                                   coronal = c("t", "d", "s", "z", "x"
, "j", "ts", "dz", "l", "n"),
```

```
                                    labial = c("p", "b", "f", "v", "m")
,
                                    dorsal = c("k", "g", "h")
)

dados$CONT.FON.SEG <- fct_relevel(dados$CONT.FON.SEG, "pausa", "corona
l", "dorsal", "labial")

### Checar dados ####
str(dados)
View(dados)

### Análises descritivas e univariadas ####
# VD ####
dados %>%
  count(VD) %>%
  mutate(prop = prop.table(n))

# VD ~ SEXO.GENERO ####
tab.prop.SEXO.GENERO <- dados %>%
  count(SEXO.GENERO, VD) %>%
  group_by(SEXO.GENERO) %>%
  mutate(prop = prop.table(n)) %>%
  print()

ggplot(tab.prop.SEXO.GENERO, aes(x = SEXO.GENERO, y = prop * 100, fill
= VD)) +
  geom_bar(stat = "identity", color = "black") +
  ggtitle("Proporção das variantes de /r/ por Sexo/Gênero do falante")
+
  labs(x = "Sexo", y = "Proporção", fill = "Variantes de /r/") +
  scale_x_discrete(labels = c("feminino", "masculino")) +
  scale_fill_brewer(palette = "Purples", labels = c("tepe", "retroflex
o")) +
  theme_bw()

tab.SEXO.GENERO <- with(dados, table(SEXO.GENERO, VD)); tab.SEXO.GENER
O
prop.SEXO.GENERO <- with(dados, prop.table(tab.SEXO.GENERO) * 100); pr
op.SEXO.GENERO

chisq.test(tab.SEXO.GENERO)
mod <- glm(VD ~ SEXO.GENERO, data = dados, family = binomial)
summary(mod)
plot(allEffects(mod), type = "response")

# VD ~ FAIXA.ETARIA ####
tab.prop.FAIXA.ETARIA <- dados %>%
  count(FAIXA.ETARIA, VD) %>%
  group_by(FAIXA.ETARIA) %>%
  mutate(prop = prop.table(n)) %>%
  print()
```

```
ggplot(tab.prop.FAIXA.ETARIA, aes(x = FAIXA.ETARIA, y = prop * 100, fi
ll = VD)) +
  geom_bar(stat = "identity", color = "black") +
  ggtitle("Proporção das variantes de /r/ por Faixa Etária do falante"
) +
  labs(x = "Faixa Etária", y = "Proporção", fill = "Variantes de /r/")
+
  scale_x_discrete(labels = c("20-34", "35-59", "60+")) +
  scale_fill_brewer(palette = "Purples", labels = c("tepe", "retroflex
o")) +
  theme_bw()

tab.prop.FAIXA.ETARIA %>%
  filter(VD == "retroflexo") %>%
  ggplot(., aes(x = FAIXA.ETARIA, y = prop * 100, group = VD)) +
  geom_line(linetype = "dotted", size = 1, color = "blue") +
  geom_point(shape = 18, size = 3, fill = "black") +
  ggtitle("Proporção de retroflexo por Faixa Etária do falante") +
  labs(x = "Faixa Etária", y = "Proporção", fill = "Variantes de /r/")
+
  scale_x_discrete(labels = c("1a", "2a", "3a")) +
  ylim(0, 50) +
  theme_bw()

tab.FAIXA.ETARIA <- with(dados, table(FAIXA.ETARIA, VD)); tab.FAIXA.ET
ARIA
prop.FAIXA.ETARIA <- with(dados, prop.table(tab.FAIXA.ETARIA) * 100);
prop.FAIXA.ETARIA

chisq.test(tab.FAIXA.ETARIA)
chisq.test(tab.FAIXA.ETARIA[c(2, 3), ]) # 2a vs 3a
mod <- glm(VD ~ FAIXA.ETARIA, data = dados, family = binomial)
summary(mod)
plot(allEffects(mod), type = "response")

# VD ~ INDICE.SOCIO ####
mod <- glm(VD ~ INDICE.SOCIO, data = dados, family = binomial)
summary(mod)
plot(allEffects(mod), type = "response")

# Outras variáveis?... ESCOLARIDADE, REGIAO, ORIGEM.PAIS, CONT.FON.PRE
C...

### Análises multivariadas ####
# Checar ortogonalidade entre variáveis previsoras
# CONT.FON.SEG e POSICAO.R
with(dados, table(CONT.FON.SEG, POSICAO.R))

# CLASSE.MORFOLOGICA e POSICAO.R
with(dados, table(CLASSE.MORFOLOGICA, POSICAO.R))

# CLASSE.MORFOLOGICA e TONICIDADE
with(dados, table(CLASSE.MORFOLOGICA, TONICIDADE))
```

```
# Modelo com TONICIDADE, POSICAO.R, CLASSE.MORFOLOGIA e CONT.FON.SEG p
ara verificar multicolinearidade
mod <- glm(VD ~
             TONICIDADE +
             POSICAO.R +
             CLASSE.MORFOLOGICA +
             CONT.FON.SEG,
           data = dados, family = binomial)

summary(mod)

# Função vif() para avaliar multicolinearidade (requer pacote car)
car::vif(mod)

### Modelo de regressão logística ####
modelo <- glm(VD ~
                SEXO.GENERO +
                FAIXA.ETARIA * REGIAO +
                INDICE.SOCIO +
                CONT.FON.PREC +
                CONT.FON.SEG +
                TONICIDADE +
                POSICAO.R +
                CLASSE.MORFOLOGICA,
              data = dados, family = binomial)

summary(modelo)
lrm(VD ~
      SEXO.GENERO +
      FAIXA.ETARIA * REGIAO +
      INDICE.SOCIO +
      CONT.FON.PREC +
      CONT.FON.SEG +
      TONICIDADE +
      POSICAO.R +
      CLASSE.MORFOLOGICA,
    data = dados)

### Função step() ####
m0 <- glm(VD ~ 1, data = dados, family = binomial)
m.fw <- step(m0,
             direction = "forward",
             scope = ~
               SEXO.GENERO +
               FAIXA.ETARIA * REGIAO +
               INDICE.SOCIO +
               CONT.FON.PREC +
               CONT.FON.SEG +
               TONICIDADE +
               POSICAO.R +
               CLASSE.MORFOLOGICA)
m.fw
```

```
m.bw <- step(modelo, direction = “backward”)
m.bw

m.both <- step(m0, scope = ~
                  SEXO.GENERO +
                  FAIXA.ETARIA * REGIAO +
                  INDICE.SOCIO +
                  CONT.FON.PREC +
                  CONT.FON.SEG +
                  TONICIDADE +
                  POSICAO.R +
                  CLASSE.MORFOLOGICA)
m.both

### Função drop1() ####
drop1(modelo, test = “LR”)

### Testar overfitting
mod.lrm <- lrm (VD ~
                   SEXO.GENERO +
                   FAIXA.ETARIA * REGIAO +
                   INDICE.SOCIO +
                   CONT.FON.PREC +
                   CONT.FON.SEG +
                   TONICIDADE +
                   POSICAO.R +
                   CLASSE.MORFOLOGICA,
                 data = dados, x = T, y = T)

### Função validate() - requer pacote rms
validate(mod.lrm, B = 200, bw = T)

### Checagem de pressupostos ####

### (a) A relação entre o logit e as variáveis previsoras numéricas é
linear?
# Fazer modelo sem interação para aplicar crPlot
modelo2 <- glm(VD ~
                  SEXO.GENERO +
                  FAIXA.ETARIA +
                  REGIAO +
                  INDICE.SOCIO +
                  CONT.FON.PREC +
                  CONT.FON.SEG +
                  TONICIDADE +
                  POSICAO.R +
                  CLASSE.MORFOLOGICA,
                data = dados, family = binomial)

# Aplicar crPlot() ao modelo (requer pacote car)
crPlot(modelo2, variable = “INDICE.SOCIO”)
```

```
### (b) Há multicolinearidade?
car::vif(modelo)

### (c) Observações são independentes? -- em dados linguísticos, quase
nunca são! --> MODELOS DE EFEITOS MISTOS

### Criar modelo linear de efeitos mistos ###
# função glmer() depende dos pacotes lme4 e lmerTest
mod.glmer <- glmer(VD ~
                     SEXO.GENERO +
                     FAIXA.ETARIA * REGIAO +
                     INDICE.SOCIO +
                     CONT.FON.PREC +
                     CONT.FON.SEG +
                     TONICIDADE +
                     POSICAO.R +
                     CLASSE.MORFOLOGICA +
                     (1|INFORMANTE) +
                     (1|ITEM.LEXICAL),
                   data = dados, family = binomial)

# Aplicar summary() a mod.glmer
summary(mod.glmer)

# Visualizar resultados numéricos em gráfico de efeitos (requer pacote
effects)
plot(allEffects(mod.glmer), type = "response")

# Gráficos de efeitos com argumento ask = T
plot(allEffects(mod.glmer), type = "response", ask = T)
```

SOBRE A AUTORA

Livia Oushiro

Livia Oushiro é doutora em Linguística pela USP. Atualmente é docente do Departamento e do Programa de Pós-Graduação em Linguística da UNICAMP e coordenadora do Laboratório VARIEM (Variação, Identidade, Estilo e Mudança). Sua pesquisa recente trata da fala de migrantes em situação de contato dialetal, de avaliações e percepções sociolinguísticas, e de modelos computacionais de padrões de variação.

OBRA JÁ PUBLICADAS

COLEÇÃO ALTOS ESTUDOS EM LINGUÍSTICA
A aventura de Saussure
Eliane Silveira

"Ai, se seu te pego...": aspectos prosódicos de estruturas desgarradas em língua portuguesa
Aline Ponciano dos Santos Silvestre

Aquisição atípica da linguagem: modelos linguísticos e prática clínica
Cristiane Lazzarotto-Volcão, Marian Oliveira e Maria João Freitas

Educação intercultural, letramentos de resistência e formação docente
Rodriana Dias Coelho Costa, Kléber Aparecido da Silva e Edinei Carvalho dos Santos

Formas de tratamento e "cordialidade": mudança linguística e conceptualizações culturais
Geisa Mara Batista

Gramaticalização e gramática gerativa
Lorenzo Teixeira Vitral

Linguagem, cognição e ensino: reflexão sobre a linguagem em crianças com e sem diagnósticos
Thalita Cristina Souza Cruz e Fernanda Moraes D'Olivo

Manual de Prosódia Experimental
Plinio A. Barbosa

Monotongação de ditongos orais no português brasileiro: uma revisão sistemática da literatura
Nancy Mendes Torres Vieira

O caso mais grosseiro da semiologia: o que Saussure pode nos dizer sobre os nomes próprios?
Stefania Montes Henriques

Uma abordagem da cena genérica como embreante paratópico: em pauta as cartas privadas de Mário, Drummond, Freud, Sêneca e John Wesley
Manuel Veronez

COLEÇÃO LINGUÍSTICA EM AÇÃO
Introdução à estatística para linguistas
Livia Oushiro

Investigando os sons de línguas não nativas: uma introdução
Felipe Flores Kupske, Ubiratã Kickhöfel Alves e Ronaldo Mangueira Lima Jr.

Linguística no feminino. Vozes femininas que fizeram a linguística no Brasil
Danniel Carvalho e Raquel Freitag

Manual de Morfologia Distribuída
Ana Paula Scher, Indaiá de Santana Bassani e Paula Roberta Gabbai Armelin

REVISÃO
Livia Oushiro

CAPA E PROJETO GRÁFICO
Ad&a Studio

FICHA CATALOGRÁFICA

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Oushiro, Livia

Introdução à estatística para linguistas [livro eletrônico] / Livia Oushiro. -- 1. ed. -- Campinas, SP : Editora da Abralin, 2022. -- (Linguística em Ação).

PDF

Bibliografia.

ISBN 978-85-68990-20-9

1. Estatística 2. Língua e linguagem 3. Linguistas4. Linguística 5. Sociolinguística 6. Proporção 7. Variáveis (Matemática) I. Título. II. Série.

CDD-410.7

23-144515 -519.507

Índices para catálogo sistemático:

1. Linguística : Estudo e ensino 410.7

2. Estatística : Estudo e ensino 519.507

Henrique Ribeiro Soares - Bibliotecário - CRB-8/9314

DOI 10.25189/9788568990209

EDITORA DA **ABRALIN**

editora.abralin.org