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()
```

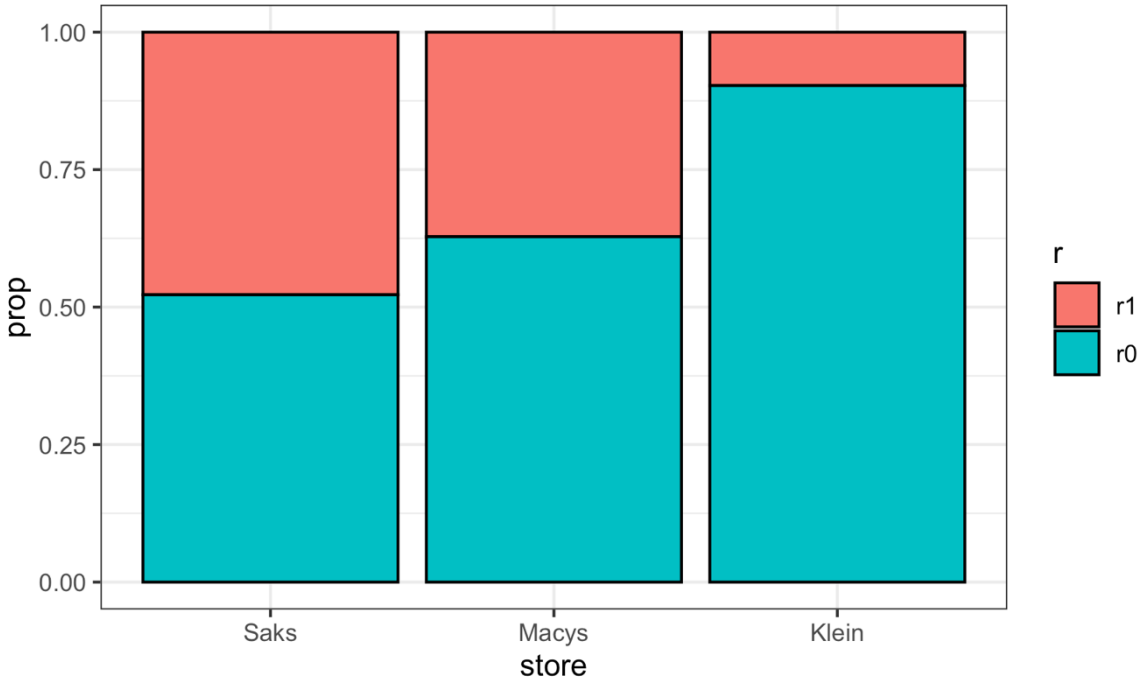

Figura 5.1: Gráfico de barras simples com ggplot. Fonte: própria.

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()
```

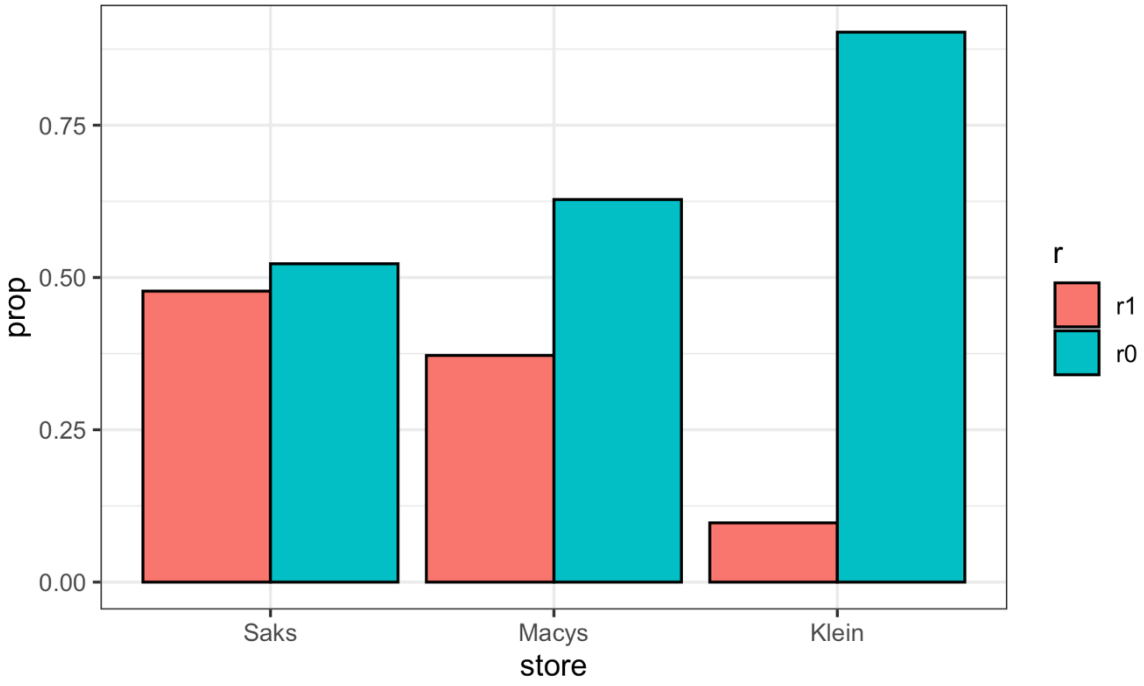

Figura 5.2: Gráfico de barras com position = "dodge". Fonte: própria.

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()
```

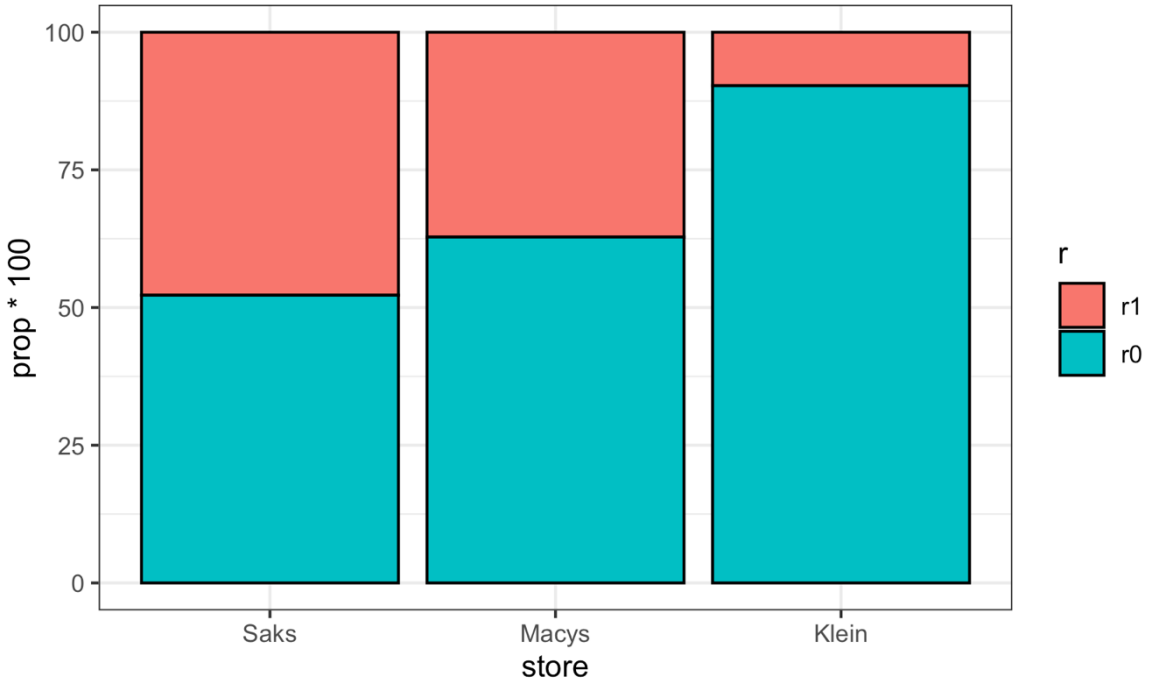

Figura 5.3: Gráfico de barras com proporção até 100. Fonte: própria.

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()
```

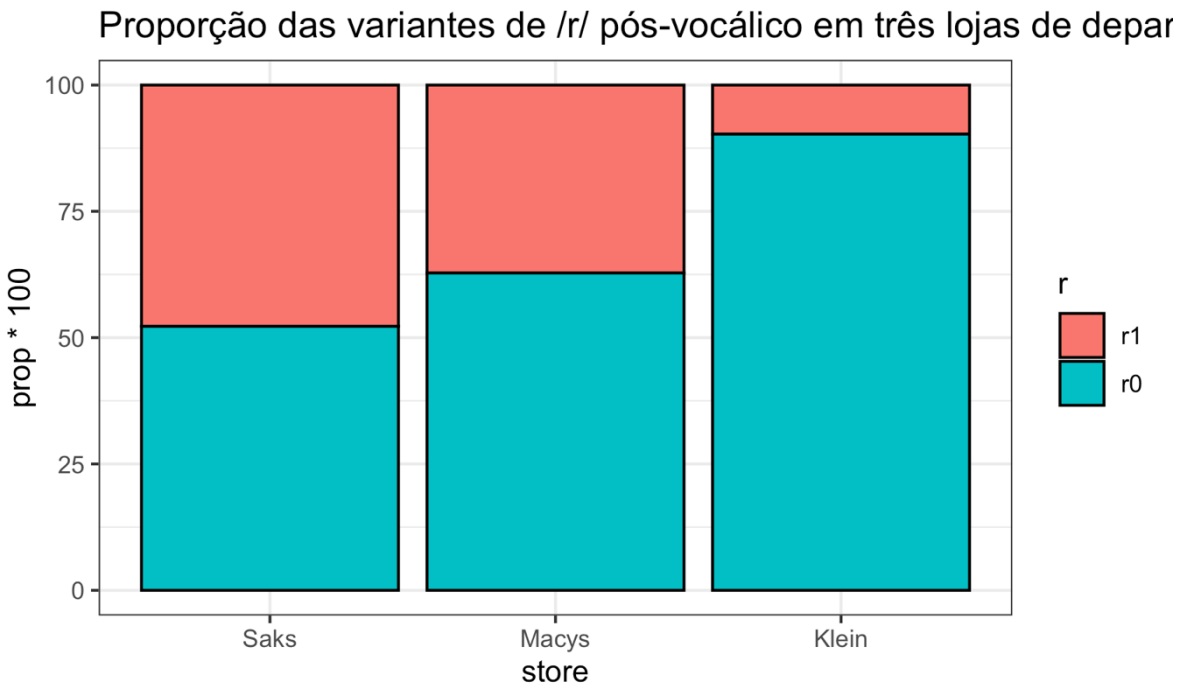

Figura 5.4: Gráfico de barras com título (1). Fonte: própria.

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()
```

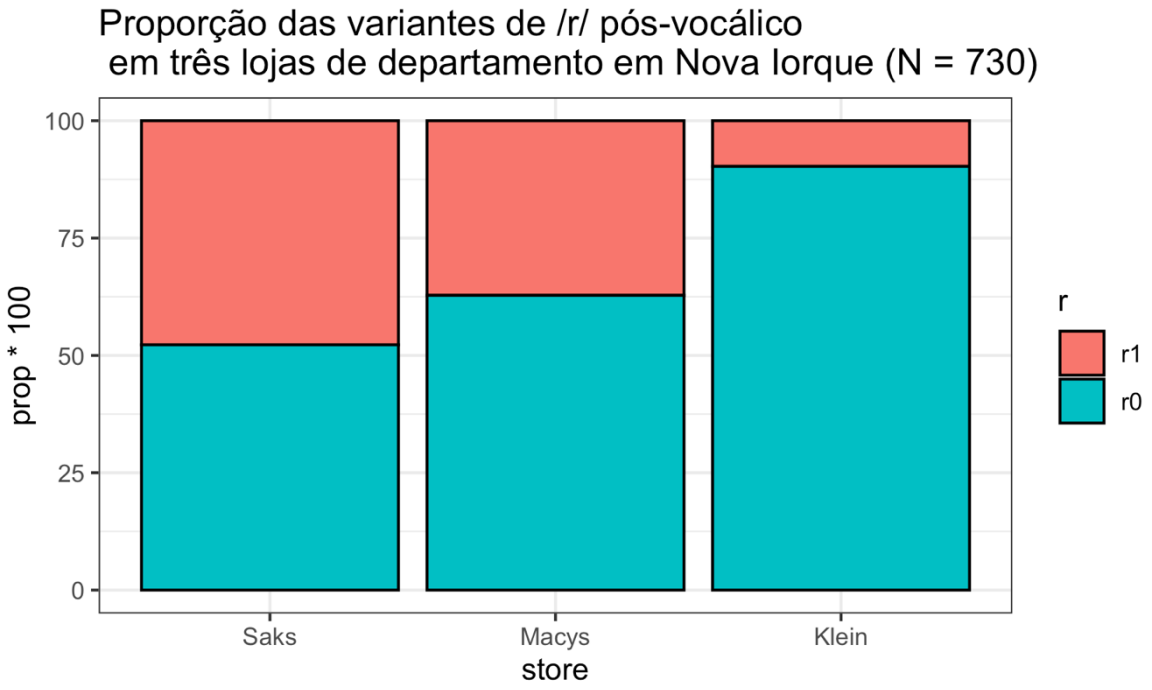

Figura 5.5: Gráfico de barras com título (2). Fonte: própria.

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()
```

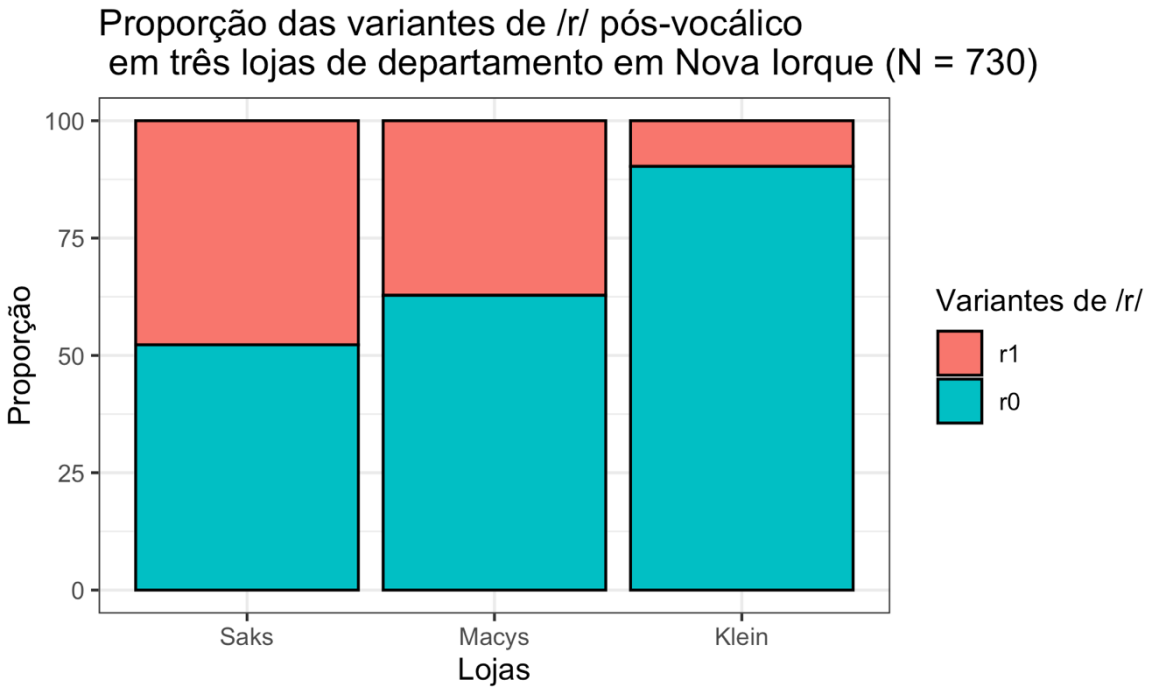

Figura 5.6: Gráfico de barras com nomes dos eixos customizados. Fonte: própria.

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()
```

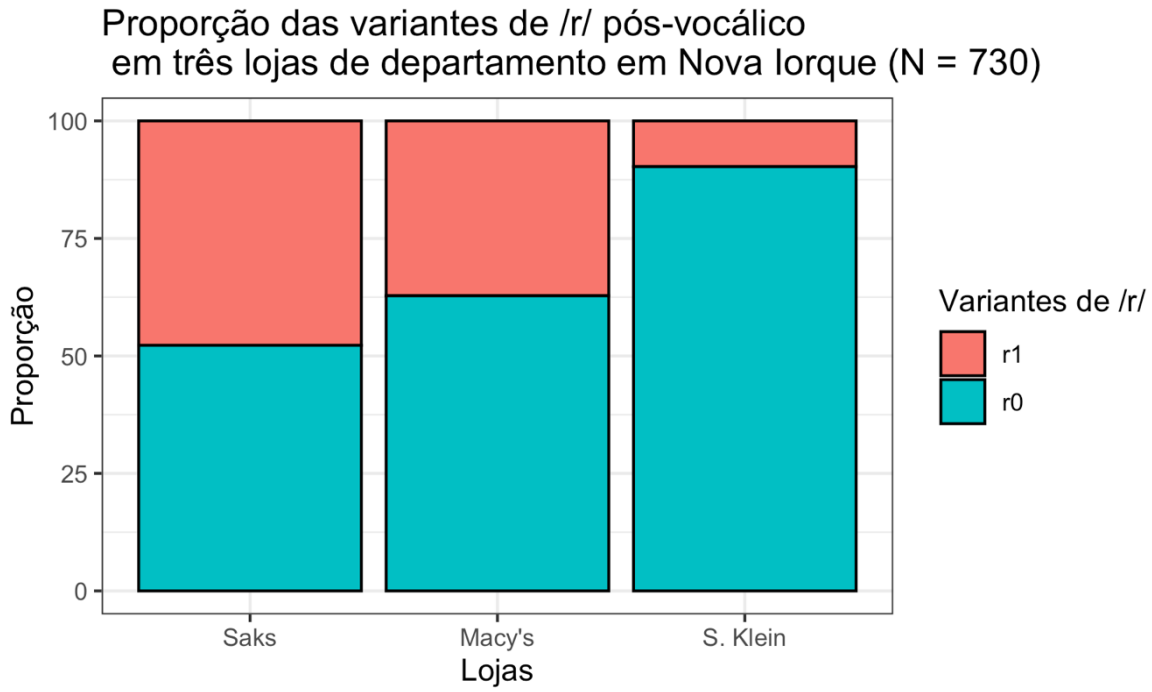

Figura 5.7: Gráfico de barras com rótulos das barras customizados. Fonte: própria.

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()
```

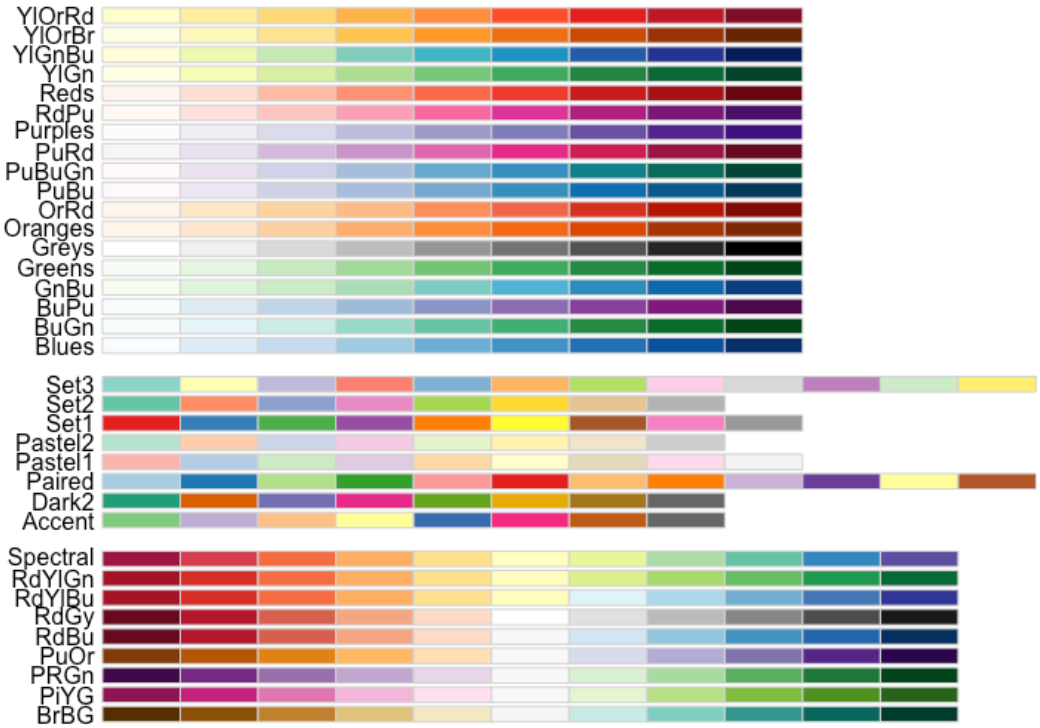

Figura 5.8: Resultado de `display.brewer.all()`. Fonte: própria.

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()
```

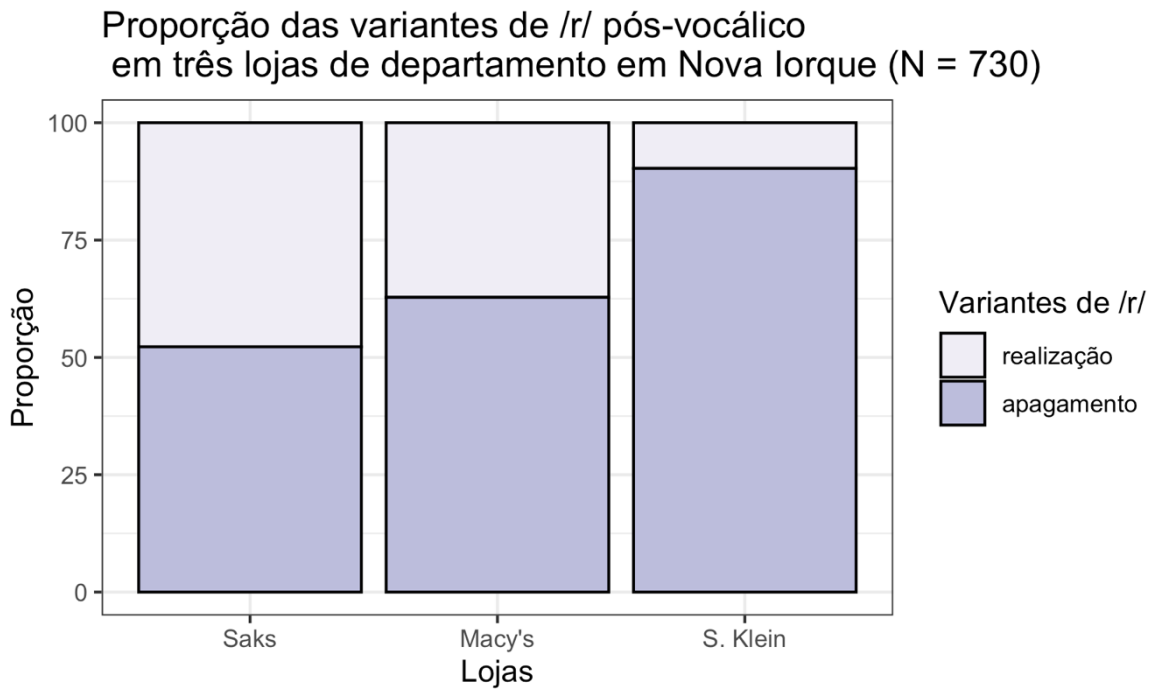

Figura 5.9: Gráfico de barras com cores das barras customizadas. Fonte: própria.

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()
```

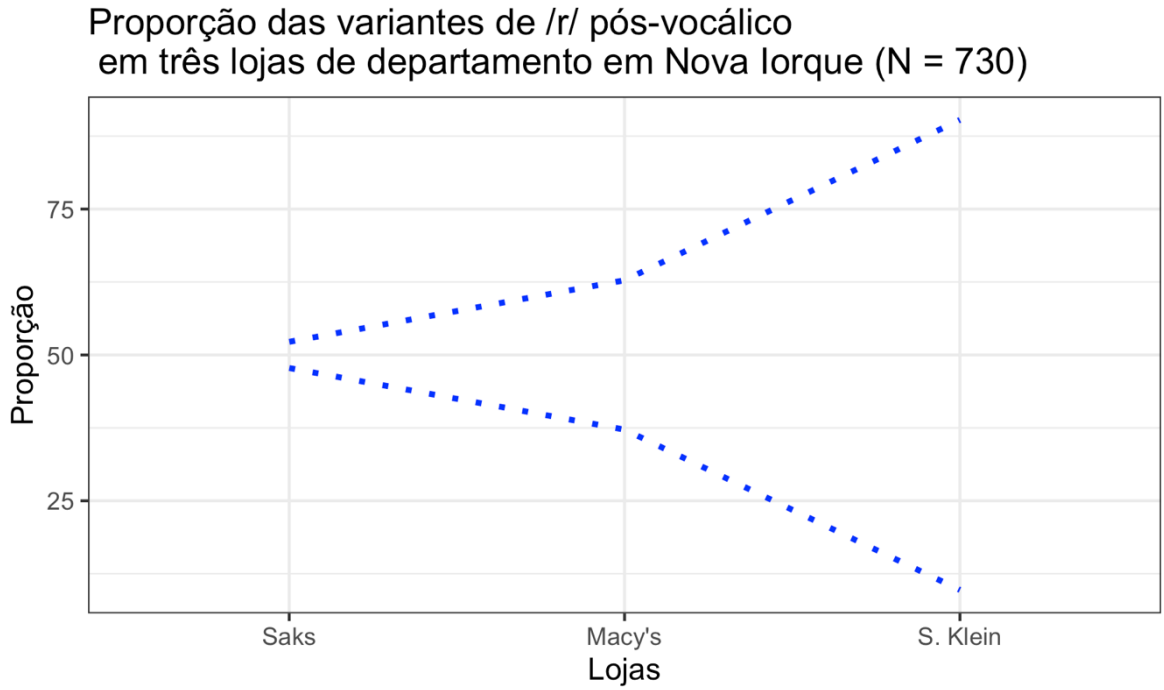

Figura 5.10: Gráfico de linhas simples com ggplot. Fonte: própria.

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()
```

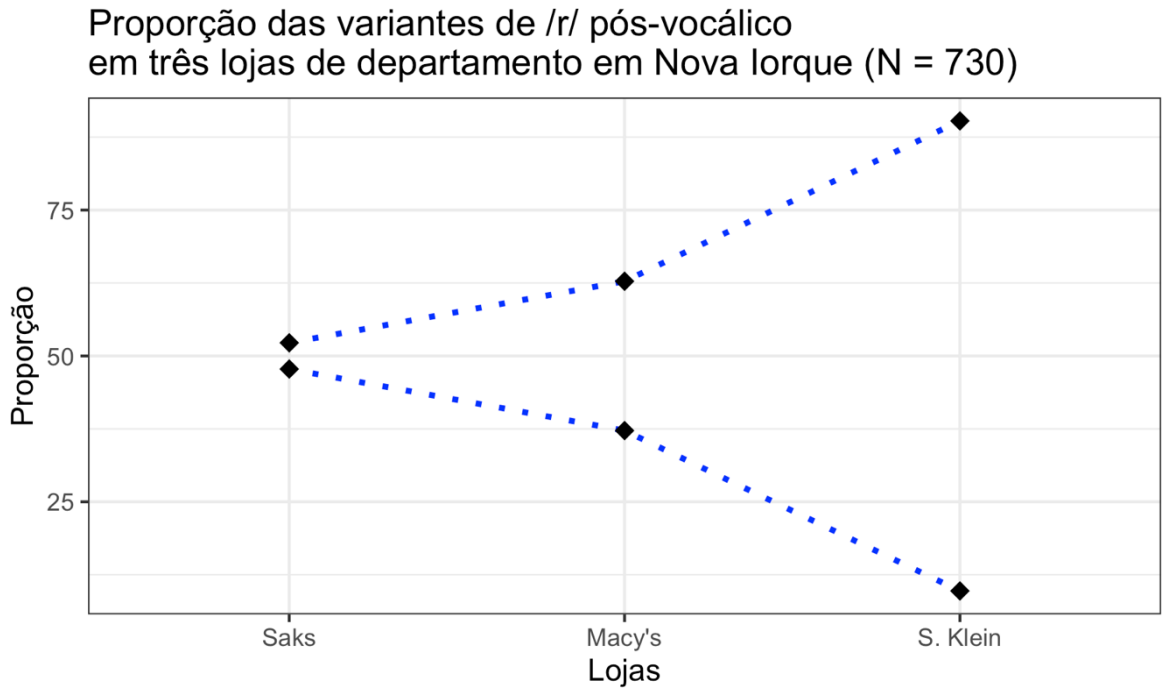

Figura 5.11: Gráfico de linhas com adição de `geom_point()`. Fonte: própria.

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()
```

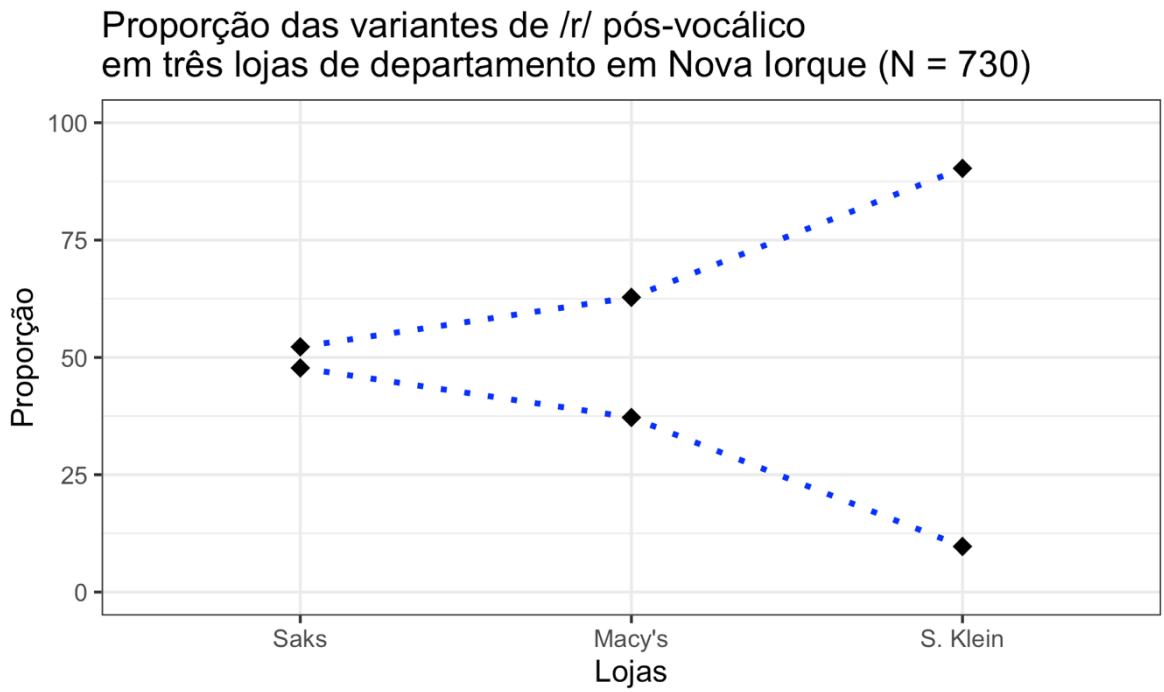

Figura 5.12: Gráfico de linhas com definição dos limites do eixo y. Fonte: própria.

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()
```

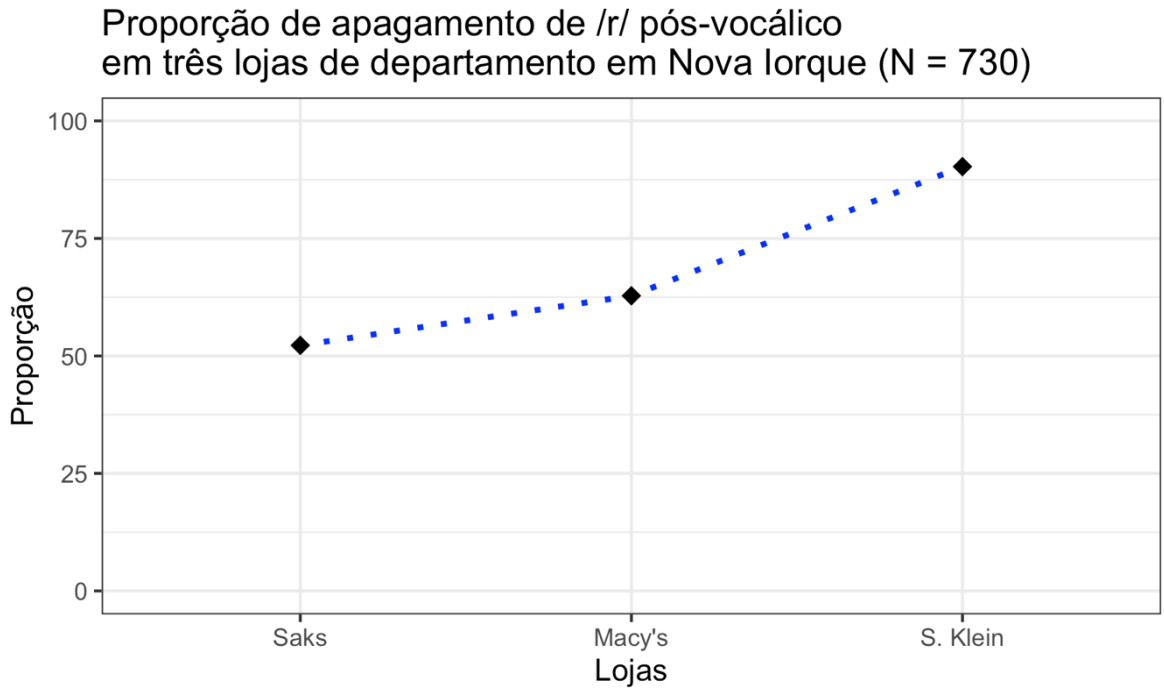

Figura 5.13: Gráfico de linhas com dados de apenas uma variante. Fonte: própria.

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!