# 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