Jeidsan A. da C. Pereira

R para Data Science

Solução dos exercícios

To Shao Yong (邵雍),
for sharing a secret joy with simple words;

月到天心处，风来水面时。
一般清意味，料得少人知。

and

To Hongzhi Zhengjue (宏智禅师),
for sharing the peace of an ending life with simple words.

梦幻空华，六十七年；
白鸟淹没，秋水连天。

Conteúdo

Prefácio

Esta página serviu para estudo e prática com o pacote R Bookdown e contém a solução encontrada por mim para os exercícios propostos no livro R para Data Sciente, de Hadley Wickham e Garret Grolemund, publicado no Brasil em 2019 pela Alta Books Editora [Wickham and Grolemund, 2019].

Por se tratar de um produto construído durante o processo de aprendizagem, o conteúdo pode conter erros, tanto no texto em si, como na lógica utilizada para solução dos exercícios.

Dúvidas ou sugestões de melhoria podem ser encaminhadas para o e-mail *jeidsan.pereira@gmail.com*[1].

Pendências

- No PDF, o prefácio está sendo exibido duas vezes no sumário;
- Exercício 1.7.4;
- Exercício 2.3.3;
- Exercício 3.5.1;
- Exercício 3.7.1;
-

[1] mailto:jeidsan.pereira@gmail.com

Parte I

Explorar

1

Visualização de dados com `ggplot2`

Para a correta execução dos códigos desse capítulo, utilizaremos algumas configurações específicas.

Inicialmente, precisaremos carregar o pacote `nycflights13`, que contém os dados de todos os voos da cidade de Nova York em 2013.

```
library(nycflights13)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

1.1 Introdução

Não temos exercícios nesta seção.

1.2 Primeiros passos

Exercício 1.2.1

Execute `ggplot(data=mpg);`. O que você vê?

Solução.

```
ggplot(data=mpg) +
    tema
```


É exibido um quadro em branco. Este quadro contém o sistema de coordenadas sobre o qual serão desenhados os grpaficos que pretendemos exibir.

Exercício 1.2.2

Quantas linhas existem em `mtcars`? Quantas colunas?

Solução.

```
dim(mtcars)
```

```
## [1] 32 11
```

R.: Existem 32 linhas e 11 colunas.

Exercício 1.2.3

O que a variável `drv` descreve?

Solução. Executamos o comando `?mpg` no console no R e a página de ajuda foi aberta. Nela encontramos o significado de cada variável do conjunto de dados.

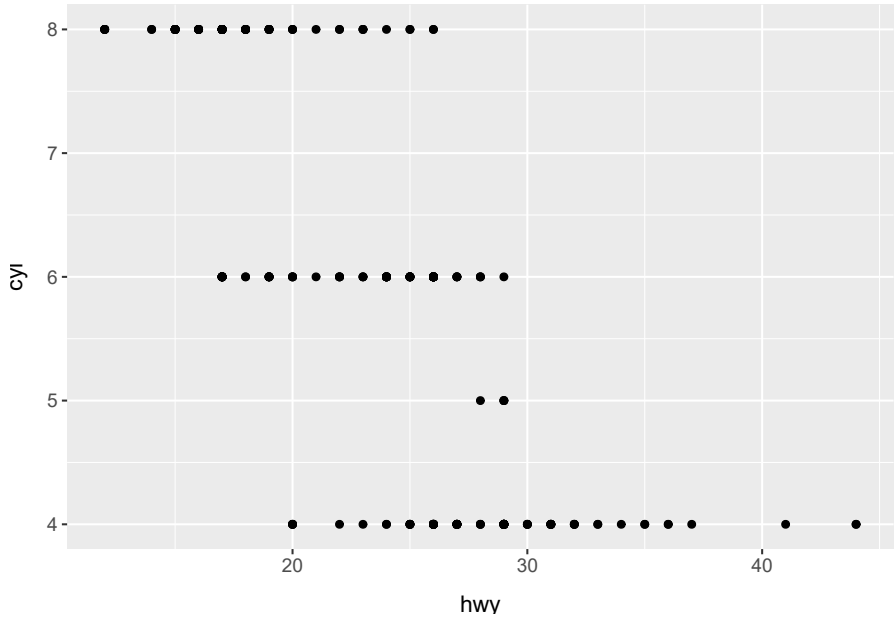
A varíável descreve o tipo de tração dos carros analisados, onde `f` significa tração dianteira, `r` significa tração traseira e `4` significa tração nas quatro rodas.

Exercício 1.2.4

Faça um gráfico de dispersão de `hwy` *versus* `cyl`.

Solução.

```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = hwy, y = cyl)) +
    tema
```

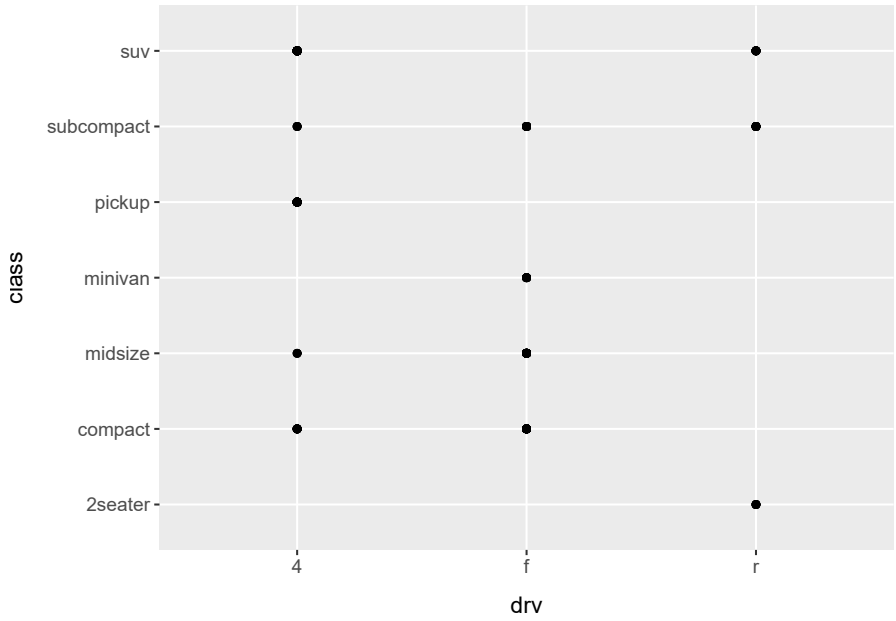


Exercício 1.2.5

O que acontece se você fizer um gráfico de dispersão de `class` *versus* `drv`? Por que esse gráfico não é útil?

Solução.

```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = drv, y = class)) +
    tema
```

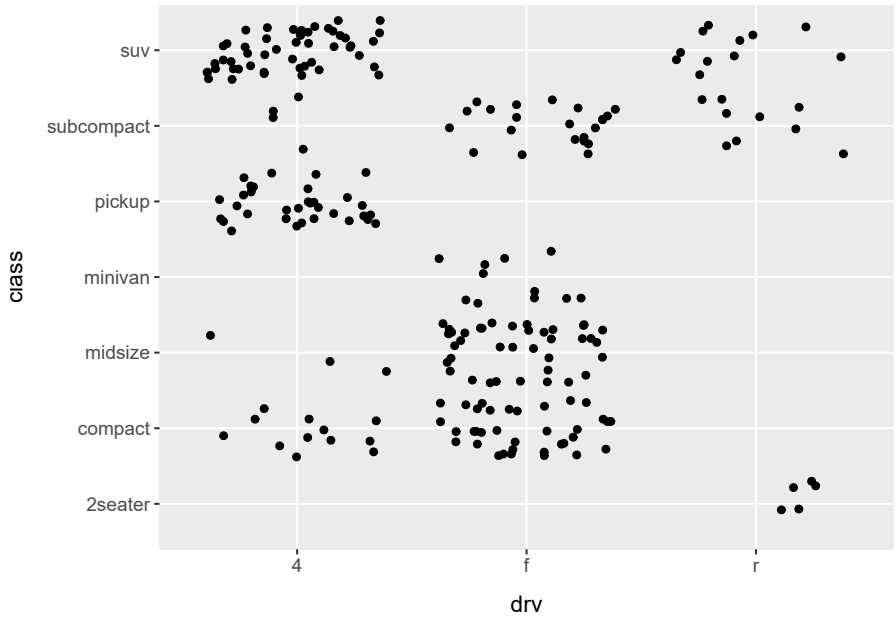


Apesar de serem exibidos dados no gráfico, nenhuma informação substancial é extraída, uma vez que o tipo de tração não está (a princípio) relacionado com a categoria do carro. Outro fator que torno o gráfico pouco informativo é que há, por exemplo, diversas SUVs com tração nas 4 rodas, contudo os valores ficam sobrepostos no gráfico, não dando dimensão do quanto de dados temos.

Abaixo seguem duas opções de como trazer mais informação ao gráfico:

- a primeira opção adiciona um ruído aos dados (`position = jitter` ou `geom_jitter()`) de modo que não haja sobreposição;

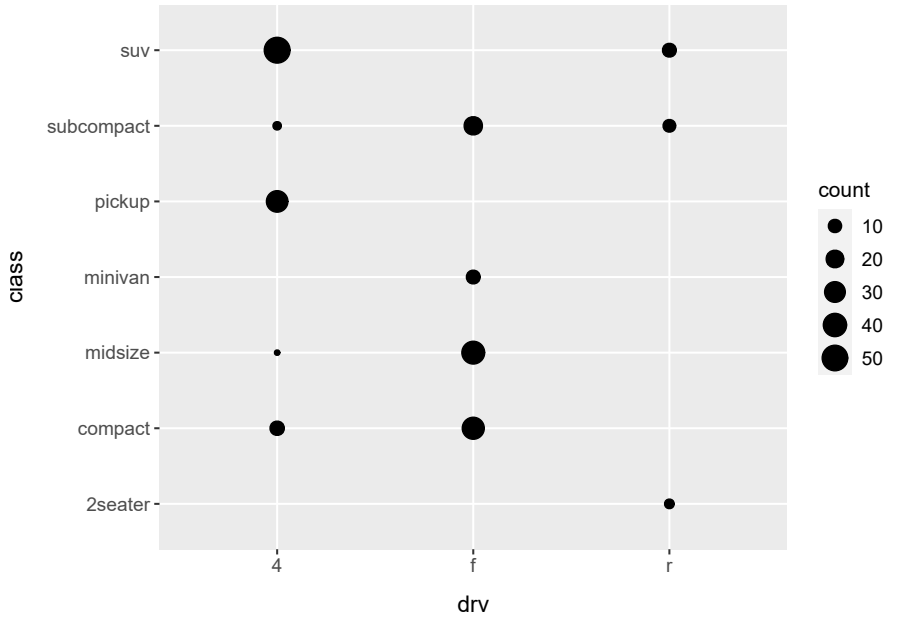
```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = drv, y = class), position = "jitter") +
    tema
```



- a segunda opção, bem mais avançada, adiciona uma estética de `size` considerando a quantidade de registros.

```
mpg %>%
    group_by(class, drv) %>%
    summarize(count = n()) %>%
    ggplot(mapping = aes(x = drv, y = class, size = count)) +
        geom_point() +
        tema
```

```
## `summarise()` has grouped output by 'class'. You can override using the
## `.groups` argument.
```

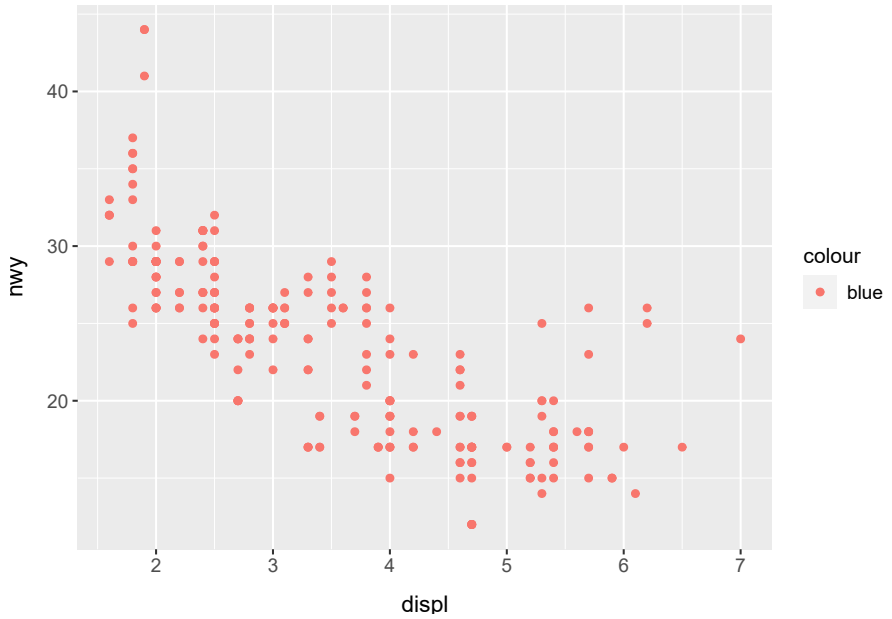


1.3 Mapeamentos estéticos

Exercício 1.3.1

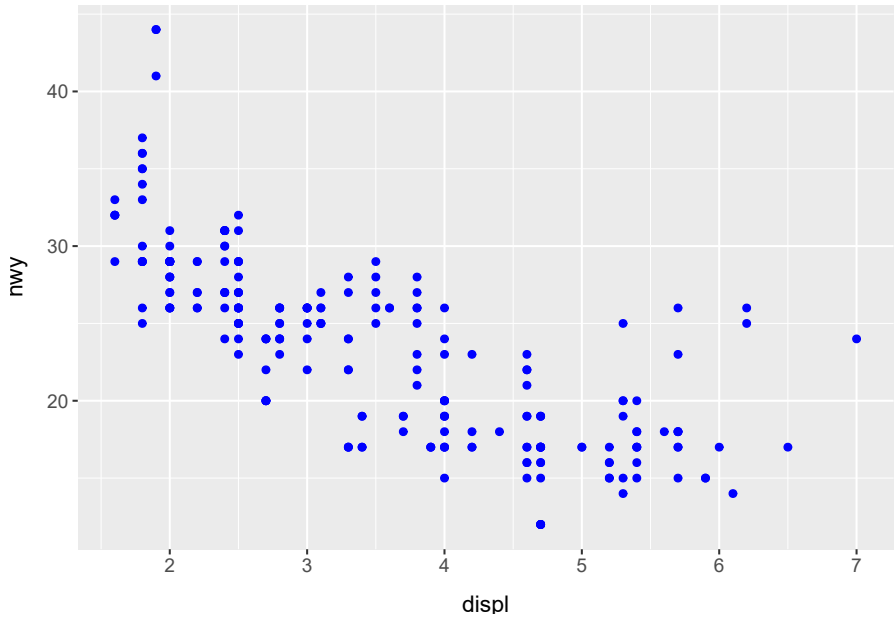
O que há de errado com este código? Por que os pontos não estão azuis?

```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy, color = "blue")) +
    tema
```



Solução. Ao invés de atribuir uma cor aos elementos de `geom_point`, o atributo `color` foi passado como uma estética. O gráfico deveria ser construído da seguinte maneira:

```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy), color = "blue") +
    tema
```



Exercício 1.3.2

Quais variáveis em `mpg` são categóricas? Quais variáveis são contínuas? Como você pode ver essa informação quando executa `mpg`?

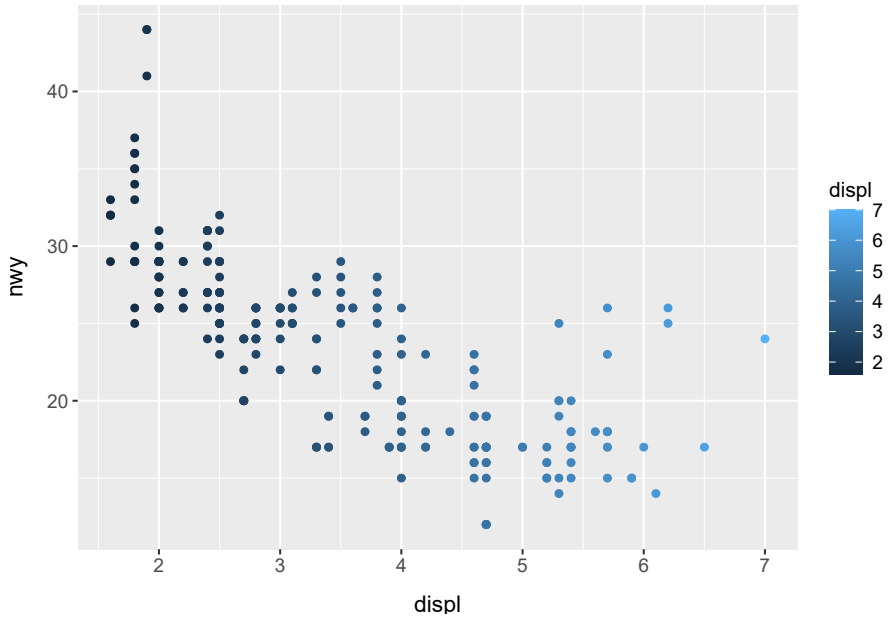
Solução. Usando `?mpg` vemos que as variáveis categóricas são: `manufacturer`, `model`, `trans`, `drv`, `fl` e `class`. As variáveis contínuas são: `displ`, `cty`, `hwy`.

Exercício 1.3.3

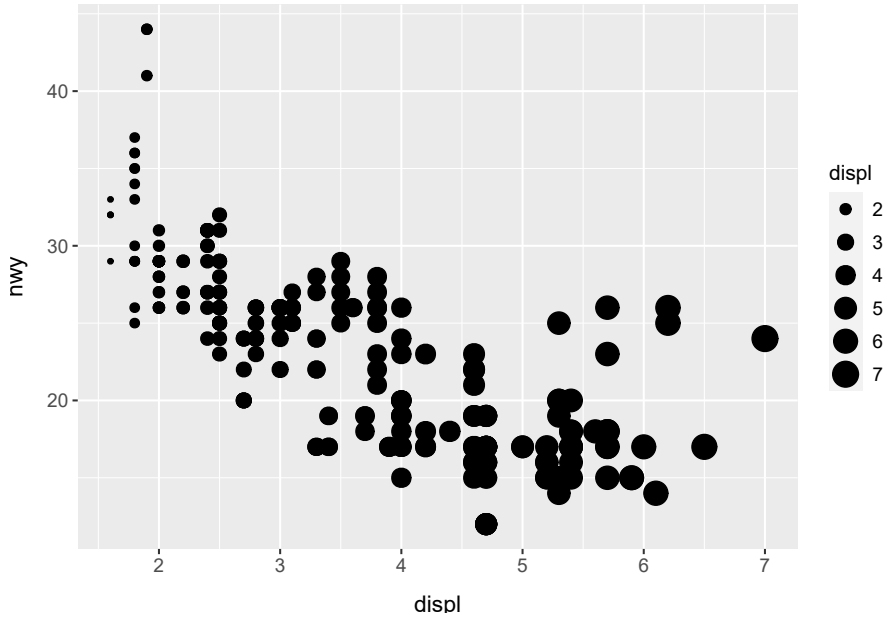
Mapeie uma variável contínua para `color`, `size` e `shape`. Como essas estéticas se comportam de maneira diferente para variáveis categóricas e contínuas?

Solução.

```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy, color = displ)) +
    tema
```




```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy, size = displ)) +
    tema
```



```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy, shape = displ)) +
    tema
```

```
## Error in `geom_point()`:
## ! Problem while computing aesthetics.
## i Error occurred in the 1st layer.
## Caused by error in `scale_f()`:
## ! A continuous variable cannot be mapped to the shape aesthetic
## i choose a different aesthetic or use `scale_shape_binned()`
```

Quando possível, a biblioteca *ggplot* apesenta a estética em um gradiente, como em color e size. Porém, nem sempre isso é possível, como vemos em `shape`, que só pode ser utilizada com variáveis discretas ou categóricas.

Exercício 1.3.4

O que acontece se você mapear a mesma variável a várias estéticas?

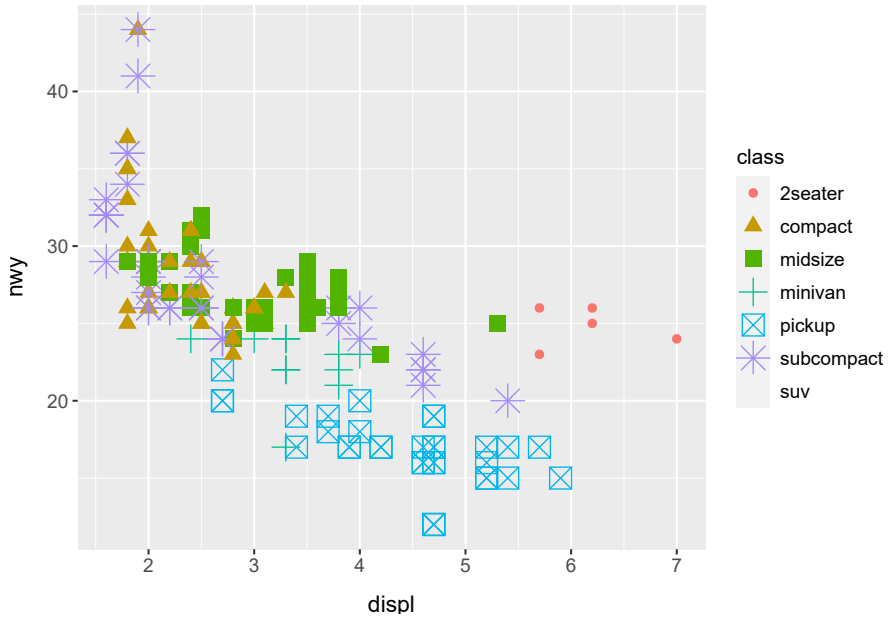
Solução.

```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy, size = class, color = class, shape = class)) +
    tema
```

```
## Warning: Using size for a discrete variable is not advised.
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values because
## more than 6 becomes difficult to discriminate; you have 7. Consider
## specifying shapes manually if you must have them.
```

```
## Warning: Removed 62 rows containing missing values (`geom_point()`).
```



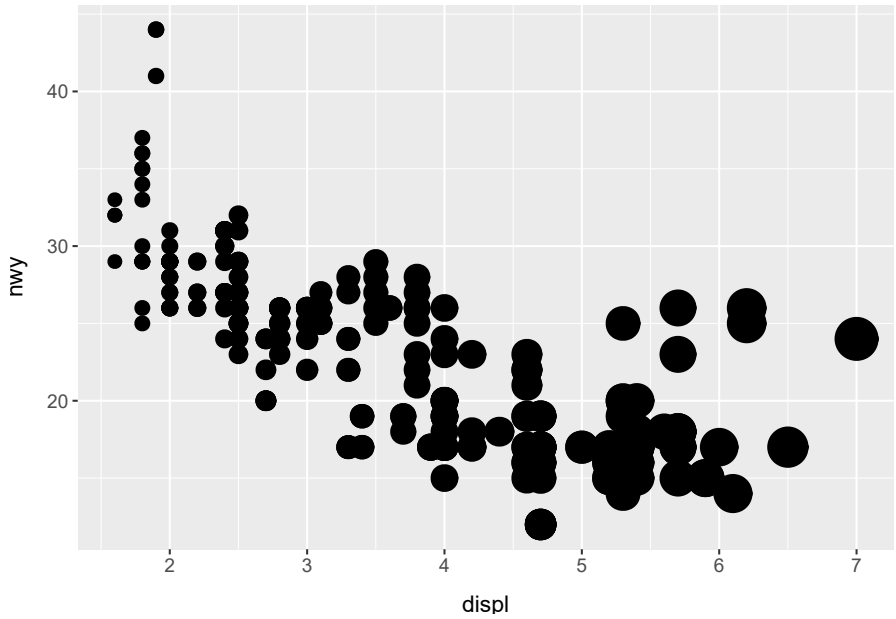
Os valores da variável serão representados de modo a atender todas as estéticas simultaneamente, por exemplo, no gráfico acima é dada uma cor, um formato e um tamanho específicos para cada classe de veículo. Os veículos de dois lugares são exibidos como um disco rosa pequeno.

Exercício 1.3.5

O que a estética `stroke` faz? com que formas ela trabalha?

Solução.

```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy, stroke = displ)) +
    tema
```



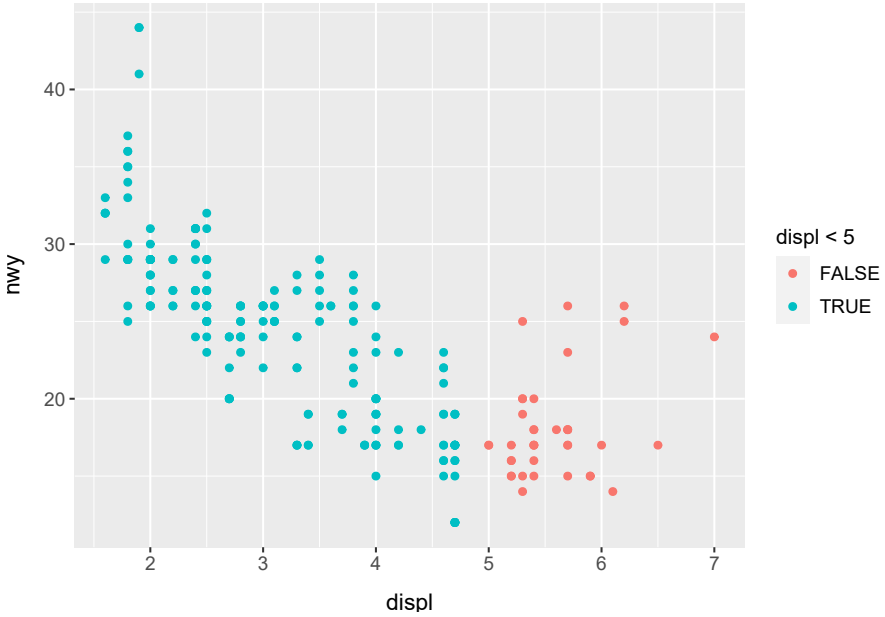
A estética `stroke` controla a espessura do ponto ou elemento a ser representado.

Exercício 1.3.6

O que acontece se você mapear uma estética a algo diferente de um nome de variável, como `aes(color = displ < 5)`?

Solução.

```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy, color = displ < 5)) +
    tema
```



A expressão é avaliada para cada um dos valores da variável e o resultado é utilizado para plotagem da estética no gráfico.

1.4 Problemas comuns

Não temos exercícios nessa seção.

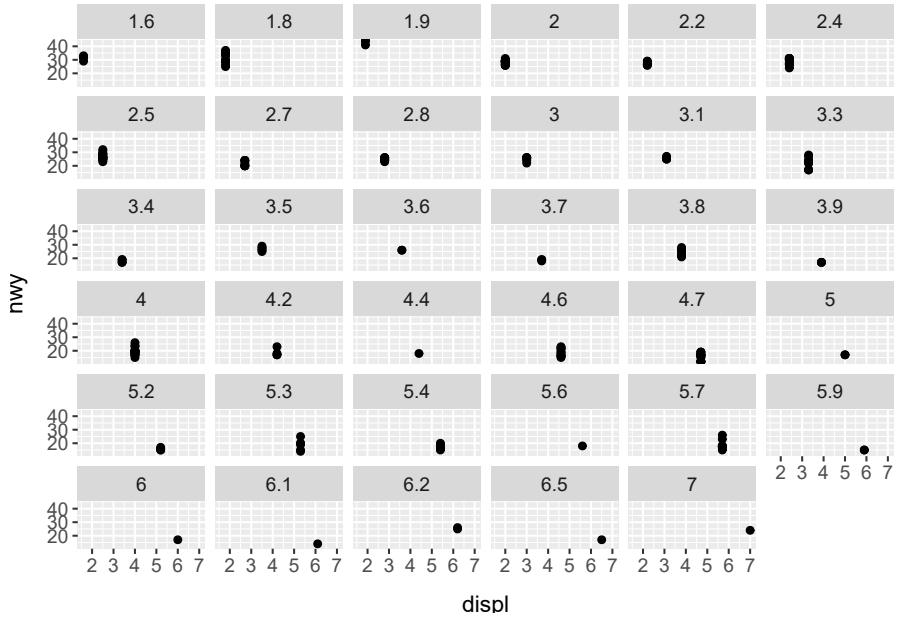
1.5 Facetas

Exercício 1.5.1

O que acontece se você criar facetas em uma variável contínua?

Solução.

```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy)) +
    facet_wrap(. ~ displ) +
    tema
```

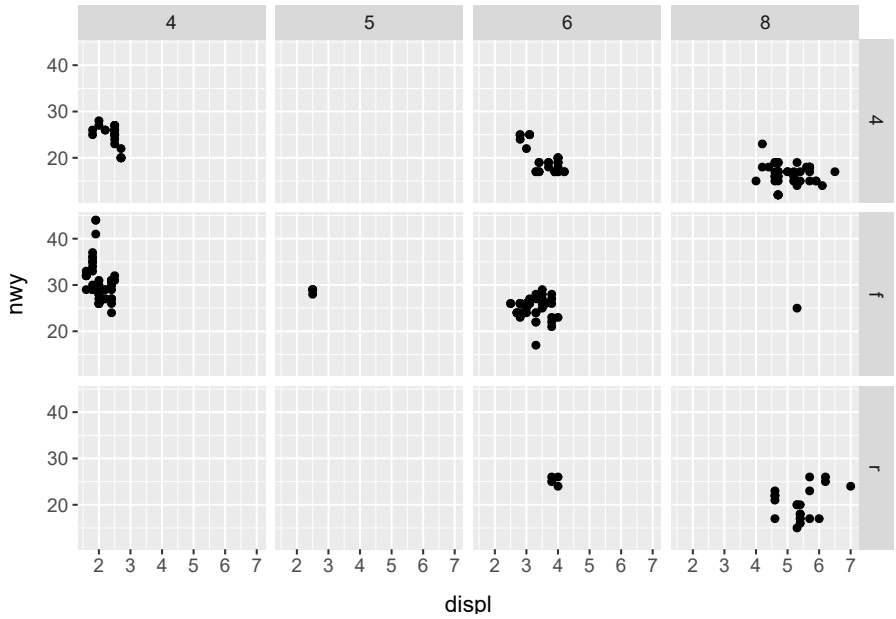


O *ggplot* se encarrega de dividir o conjunto em classes e toma o ponto médio de cada classe para realizar a quebra em facetas.

Exercício 1.5.2

O que significam as célula em branco em um gráfico com `facet_grid(drv ~ cyl)`? Como elas se relacionam a este gráfico?

```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy)) +
    facet_grid(drv ~ cyl) +
    tema
```

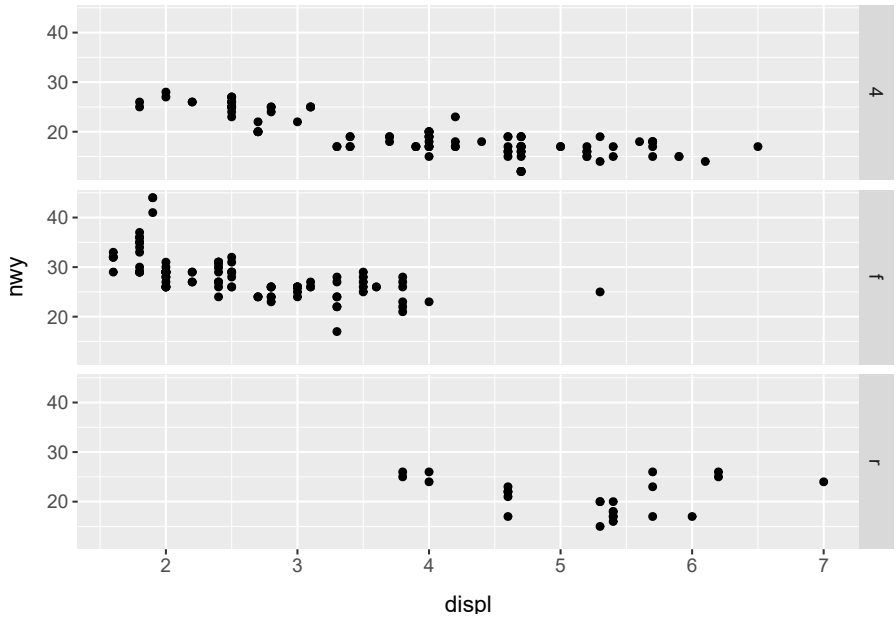


Solução. Significa que para aquela combinação de variáveis, não há nenhum valor observado. Por exemplo, não há nenhum veículo com 5 cilindros e tração nas quatro rodas.

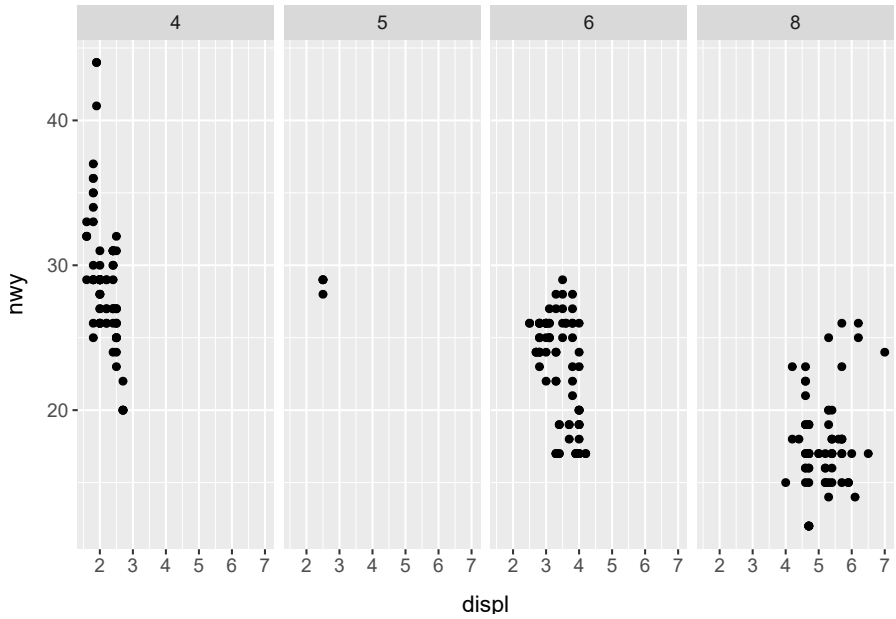
Exercício 1.5.3

Que gráficos o código a seguir faz? O que . faz?

```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy)) +
    facet_grid(drv ~ .) +
    tema
```




```
ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy)) +
    facet_grid(. ~ cyl) +
    tema
```

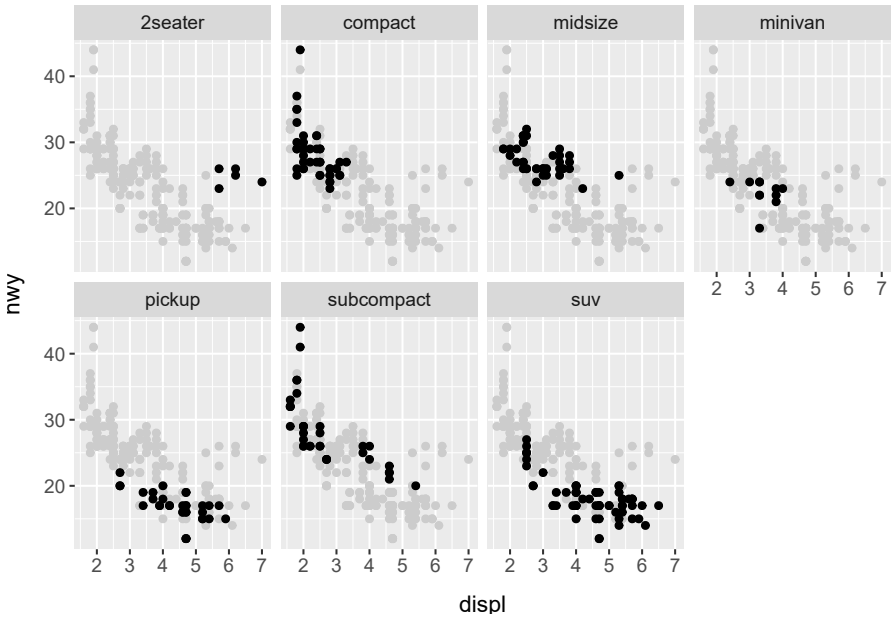


Solução. São gerados os gráficos de dispersão segregados pelas variáveis `drv` e `cyl`, respectivamente. O . indica que não queremos considerar nenhuma segregação naquela dimensão do *grid* (linha ou coluna).

Exercício 1.5.4

Pegue o primeiro gráfico em facetas dessa seção.

```
ggplot(data = mpg) +
    geom_point(data = transform(mpg, class = NULL), mapping = aes(x = displ, y = hwy), color = "gray80") +
    geom_point(mapping = aes(x = displ, y = hwy)) +
    facet_wrap(~ class, nrow = 2) +
    tema
```



Quais são as vantagens de usar facetas, em vez de estética de cor? Quais são as desvantagens? Como o equilíbrio poderia mudar se você tivesse um conjunto de dados maior?

Solução. A principal vantagem no uso de facetas é que fica mais fácil analisar os dados quando eles estão separados em seu próprio contexto, contudo visualizá-los assim dificulta a comparação entre grupos.

Exercício 1.5.5

Leia `?facet_wrap`. O que `nrow` faz? o que `ncol` faz? Quais outras opções controlam o layout de paineis individuais? Por que `facet_grid()` não tem variáveis `nrow`e `ncol`?

Solução.

```
?facet_wrap
```

Os atributos `ncol` e `nrow` são utilizados pelo `facet_wrap` para determinar o número de colunas ou linhas (respectivamente) nas quais serão distribuídos os gráficos segregados. Esses atributos não figuram em `facet_grid` pelo fato deste já organizar as facetas retangularmente.

Exercício 1.5.6

Ao usar `facet_grid()` você normalmente deveria colocar a variável com níveis mais singulares nas colunas. Por quê?

Solução. Para melhor aproveitamento do espaço em tela.

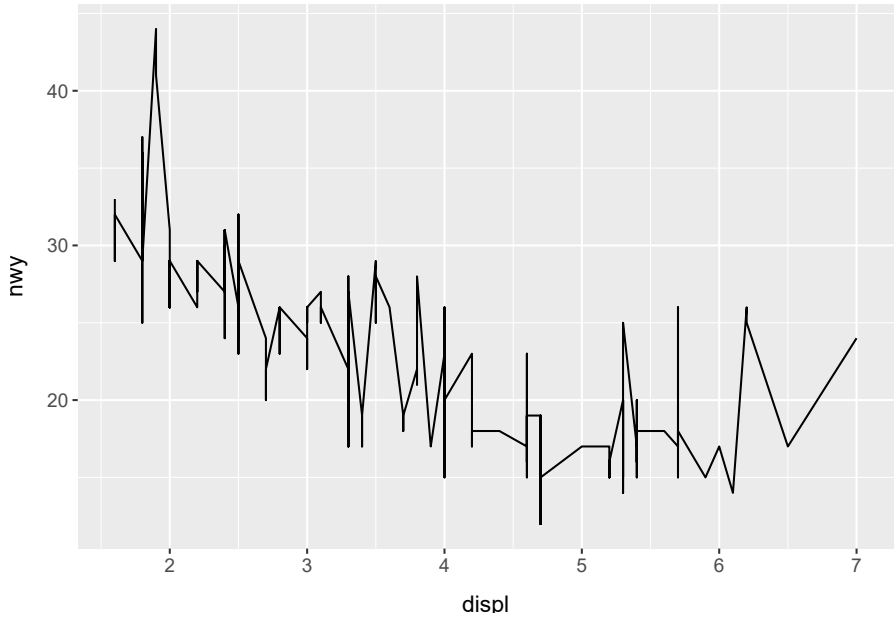
1.6 Objetos geométricos

Exercício 1.6.1

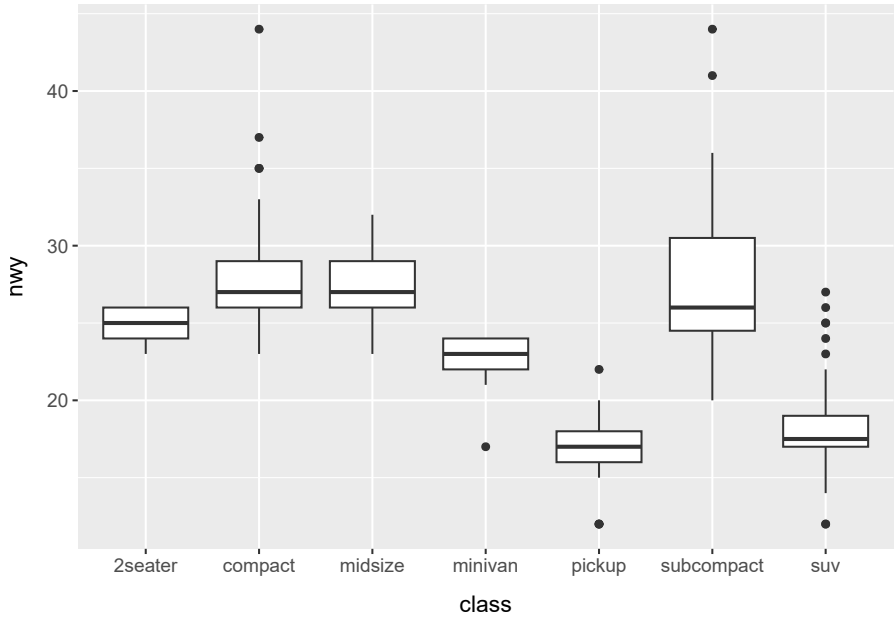
Que *geom* você usaria para desenhar um gráfico de linha? Um diagrama de caixas (*boxplot*)? Um histograma? Um gráfico de área?

Solução.

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
    geom_line() +
    tema
```

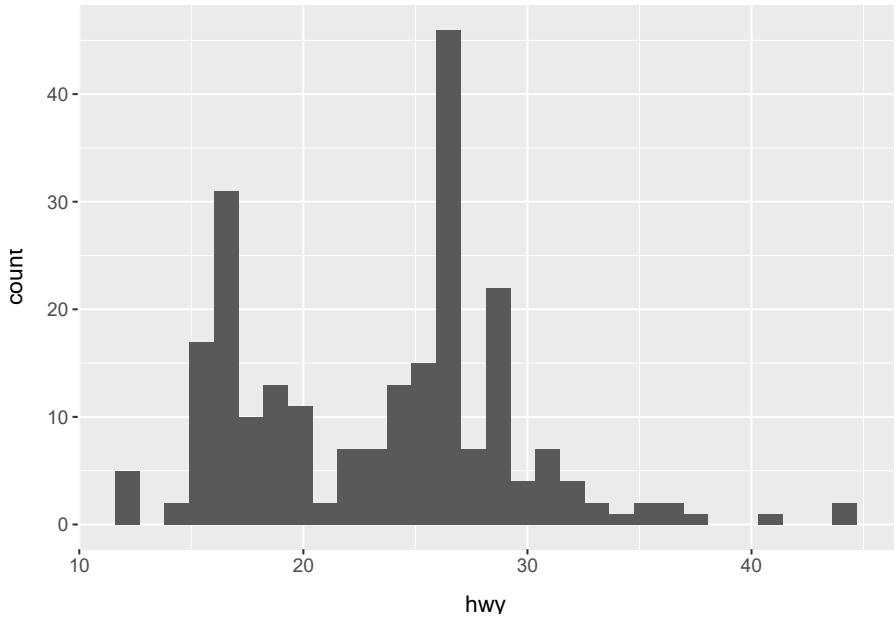


```
ggplot(data = mpg) +
    geom_boxplot(mapping = aes(y = hwy, x = class)) +
    tema
```

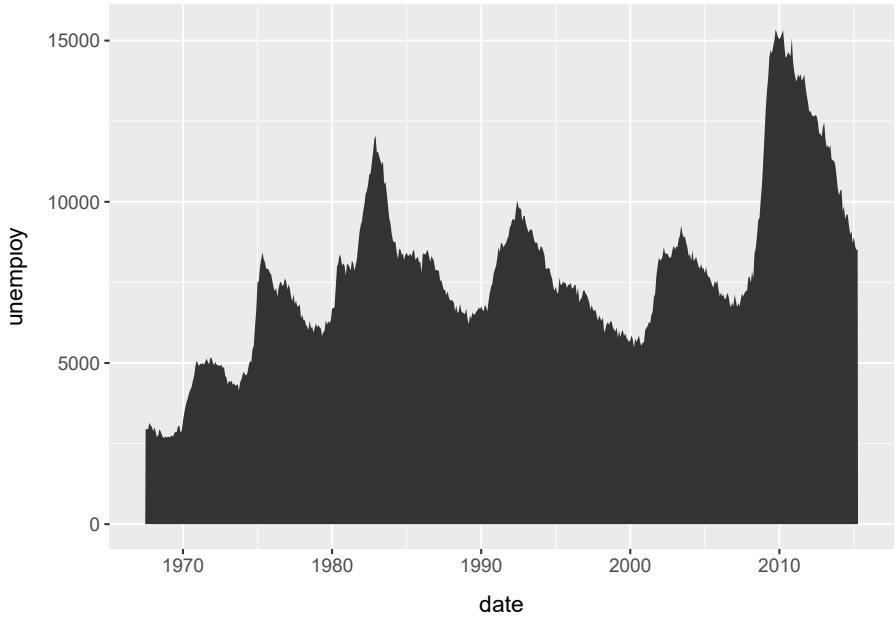



```
ggplot(data = mpg, mapping = aes(x = hwy)) +
    geom_histogram() +
    tema
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(data = economics, mapping = aes(x = date, y = unemploy)) +
    geom_area() +
    tema
```



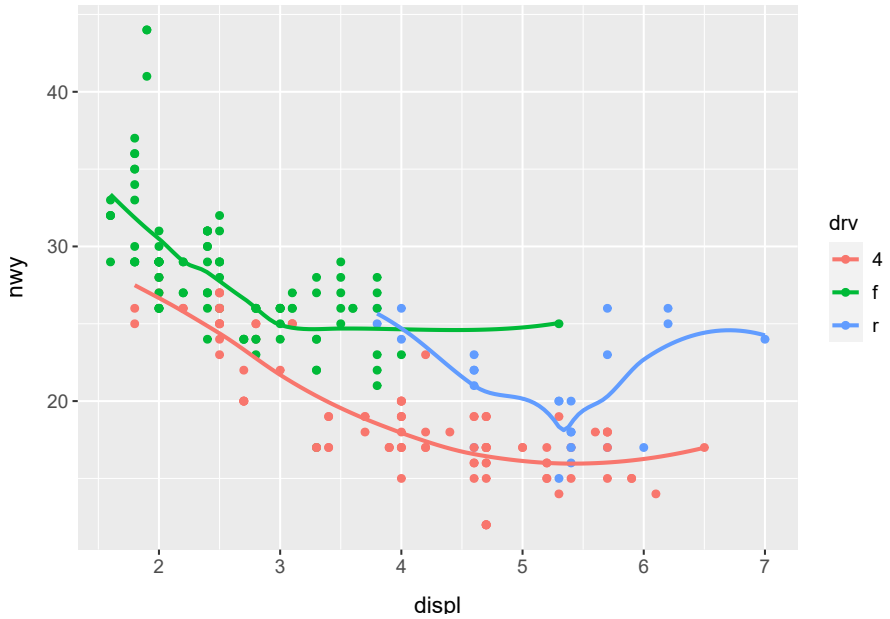
Podem ser utilizados, respectivamente as *geoms*: *line*, *boxplot*, *histogram* e *area*.

Exercício 1.6.2

Execute este código em sua cabeça e preveja como será o resultado. Depois execute o código no R e confira suas previsões:

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +
    geom_point() +
    geom_smooth(se = FALSE) +
    tema
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Solução. O gráfico bateu com a expectativa.

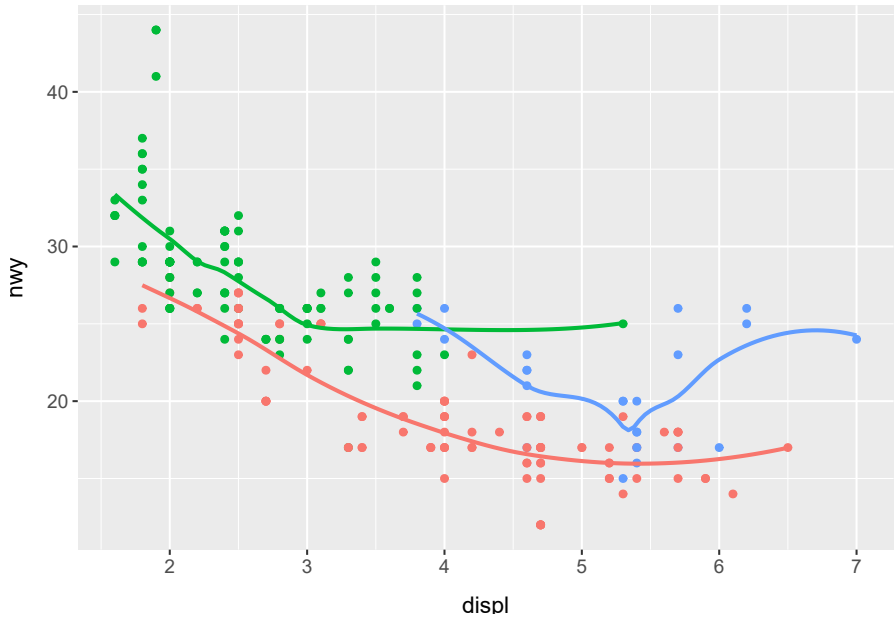
Exercício 1.6.3

O que o `show.legend = FALSE` faz? O que acontece se você removê-lo? Por que você acha que usei isso anteriormente no capítulo?

Solução.

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +
    geom_point(show.legend = FALSE) +
    geom_smooth(se = FALSE, show.legend = FALSE) +
    tema
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Ele indica que, para a camada à qual se aplica, não serão geradas as legendas de identificação.

Exercício 1.6.4

O que o argumento `se` para `geom_smooth` faz?

Solução.

```
?geom_smooth
```

Esse argumento indica se o intervalo de confiança utilizado no processo de suavização da linha deve ou não ser exibido no gráfico.

Exercício 1.6.5

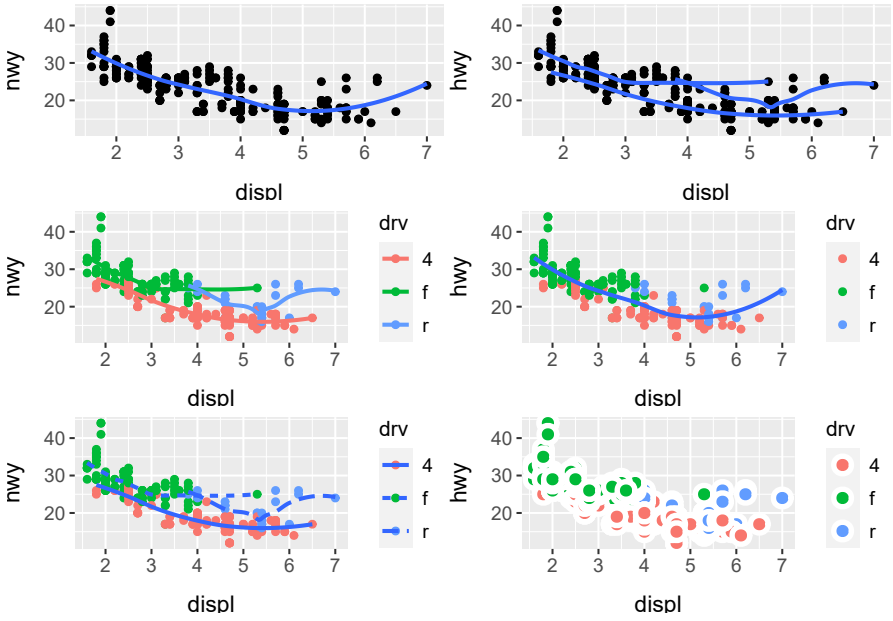
Esses dois gráficos serão diferentes? Por quê/por que não?

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
    geom_point() +
    geom_smooth() +
    tema

ggplot() +
    geom_point(data = mpg, mapping = aes(x = displ, y = hwy)) +
    geom_smooth(data = mpg, mapping = aes(x = displ, y = hwy)) +
    tema
```

Solução. Os gráficos serão iguais. Ao informar os parâmetros `data` e `mapping` na função `ggplot` essas atributos serão considerados como globais, sendo utilizado em todos as camadas do gráfico, a menos que alguma das camadas os sobrescreva. No segundo gráfico, não são definidos parâmetros globais, porém, o mesmo parâmetro é passado para ambas as camadas, sendo assim, a única diferença é o código estar duplicado.

Exercício 1.6.6

Recrie o código R necessário para gerar os seguintes gráficos:



Solução.

```
a <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
        geom_point() +
        geom_smooth(se = FALSE) +
        tema

b <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
        geom_point() +
        geom_smooth(mapping = aes(group = drv), se = FALSE) +
        tema

c <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +
        geom_point() +
        geom_smooth(se = FALSE) +
        tema

d <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
        geom_point(mapping = aes(color = drv)) +
        geom_smooth(se = FALSE) +
        tema

e <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
        geom_point(mapping = aes(color = drv)) +
        geom_smooth(mapping = aes(linetype = drv), se = FALSE) +
        tema

f <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy, fill = drv)) +
        geom_point(color = "white", shape = 21, size = 3, stroke = 2) +
        tema
```

1.7 Transformações estatísticas

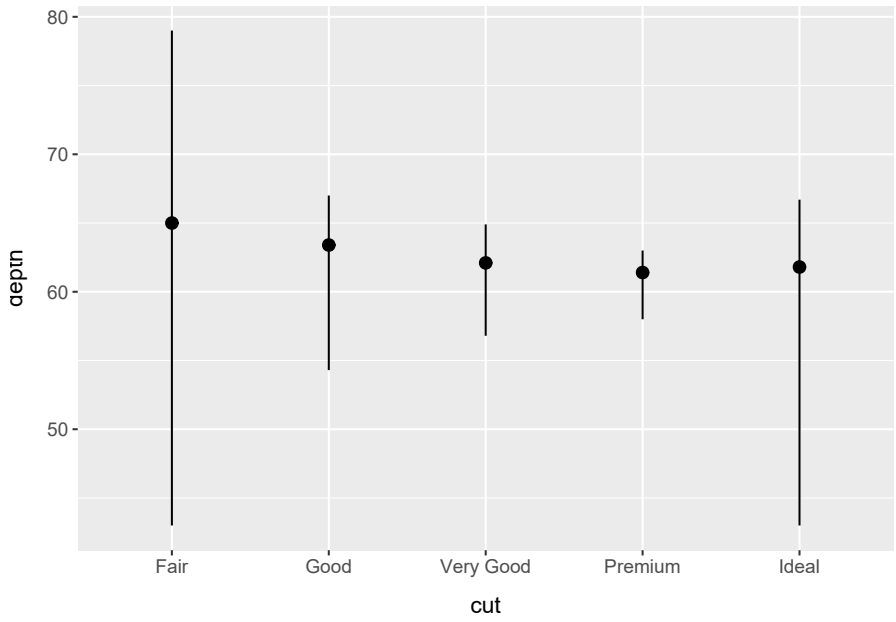
Exercício 1.7.1

Qual é o `geom` padrão associado ao `stat_summary()`? Como você poderia reescrever o gráfico anterior usando essa função `geom`, em vez da função `stat`?

Solução.

```
?stat_summary
```

```
ggplot(data = diamonds) +
    stat_summary(
        mapping = aes(x = cut, y = depth),
        fun.min = min,
        fun.max = max,
        fun = median
    ) +
    tema
```



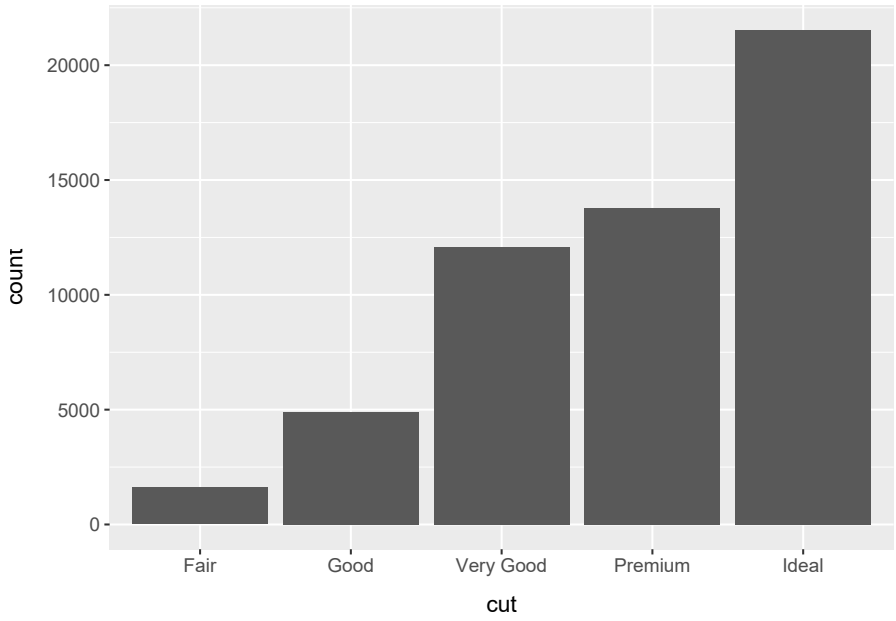
A geom associada é a geom_pointrange e o gráfico poderia ser reescrito da seguinte maneira.

Exercício 1.7.2

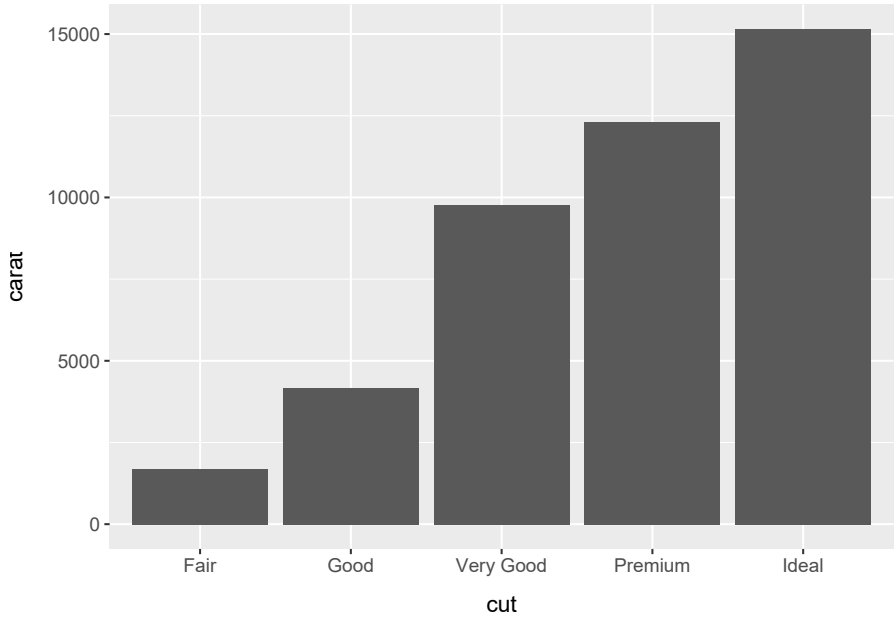
O que geom_col() faz? Qual é a diferença entre ele e geom_bar()?

Solução.

```
ggplot(data = diamonds, mapping = aes(x = cut)) +
    geom_bar() +
    tema
```




```
ggplot(data = diamonds, mapping = aes(x = cut, y = carat)) +
    geom_col() +
    tema
```



Enquanto no `geom_bar` a altura das barras representa uma transformação estatística relacionada às observações (como `count`, por exemplo), no `geom_col` podemos exibir o acumulado (soma) de uma variável para cada categoria exibida.

Exercício 1.7.3

A maioria dos `geoms` e `stats` vem em pares, que são quase sempre usados juntos. Leia a documentação e faça uma lista de todos os pares. O que eles têm em comum?

Solução.

#	Geom	Stat
01	Blank	Identity
02	Curve	Identity
03	Segment	Identity
04	Path	Identity
05	Line	Identity
06	Step	Identity
07	Poligon	Identity
08	Raster	Identity
09	Rect	Identity
10	Tile	Identity
11	Ribbon	Identity
12	Area	Identity
13	Align	?
14	ABLine	?
15	HLine	?
16	Density	Density
17	DotPlot	?
18	Freqpoly	Bin
19	Histogram	Bin
20	Col	Identity
21	Bar	Count
22	Label	Identity
23	Text	Identity
24	Jitter	Identity
25	Point	Identity
26	Quantile	Quantile
27	Rug	Identity
28	Boxplot	Boxplot
29	Violin	YDensity
30	Count	Sum
31	Bin 2D	Bin 2D
32	Density 2D	Density 2D

#	Geom	Stat
33	Hex	Bin Hex
34	Cross Bar	Identity
35	Error Bar	Identity
36	Line Range	Identity
37	Point Range	Identity
38	Map	Identity
39	Contour	Contour
40	Contour Filled	Contour Filled

Exercício 1.7.4

Quais variáveis `stat_smooth()` calcula? Quais parâmetros controlam seu comportamento?

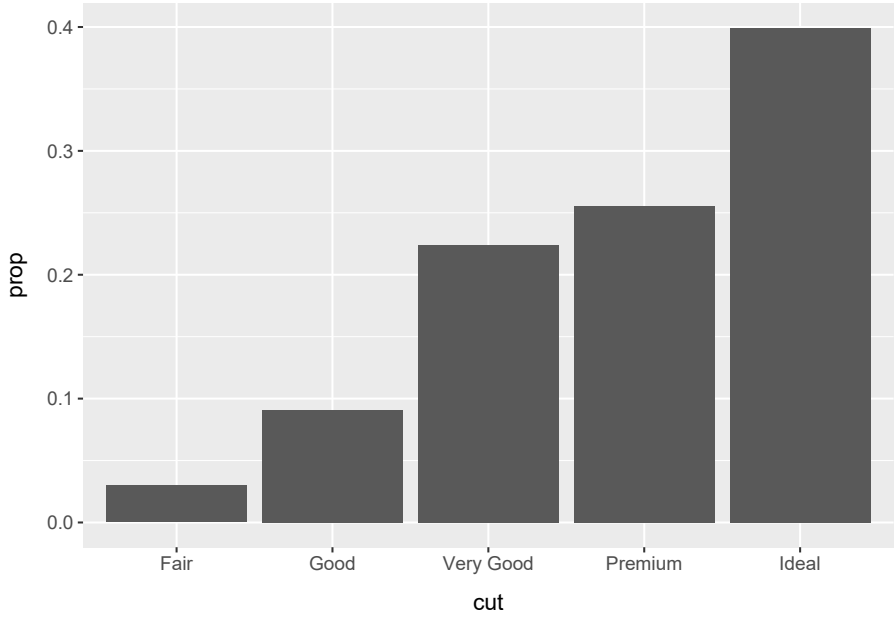
Solução.

```
?stat_smooth
```

Exercício 1.7.5

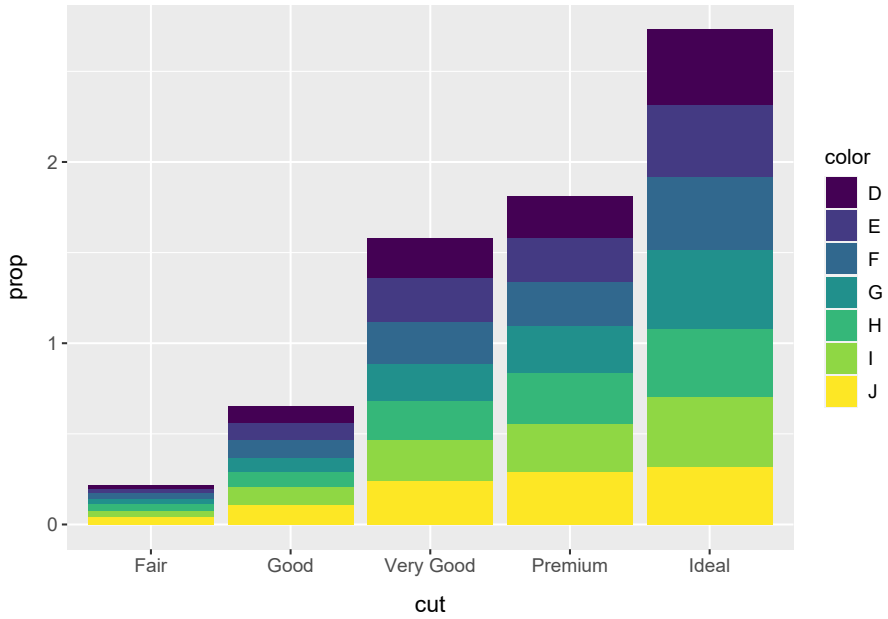
Em nosso gráfico de barra de *proportion*, precisamos configurar `group = 1`. Por quê? Em outras palavras, qual é o problema com esses dois gráficos?

```
ggplot(data = diamonds) +
    geom_bar(mapping = aes(x = cut, y = after_stat(prop), group = 1)) +
    tema
```



Solução.

```
ggplot(data = diamonds) +
    geom_bar(mapping = aes(
        x = cut,
        fill = color,
        y = after_stat(prop),
        group = color
    )) +
    tema
```



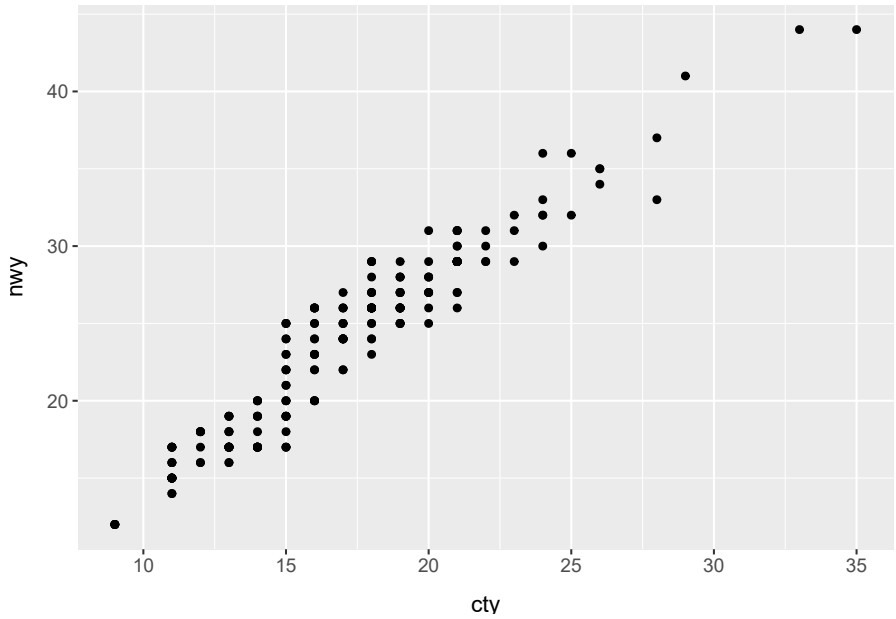
Quando estamos trabalhando com proporções (ou estátisticas em geral), é importante destacar para o `ggplot` qual agrupamento ele deve considerar, caso contrário ele irá considerar um único grupo e dará uma impressão incorreta ao gráfico. No primeiro exemplo, foi utilizado `group = 1` (e, na verdade, poderia ser qualquer valor) apenas para indicar que deveria ser realizado um agrupamento.

1.8 Ajustes de posição

Exercício 1.8.1

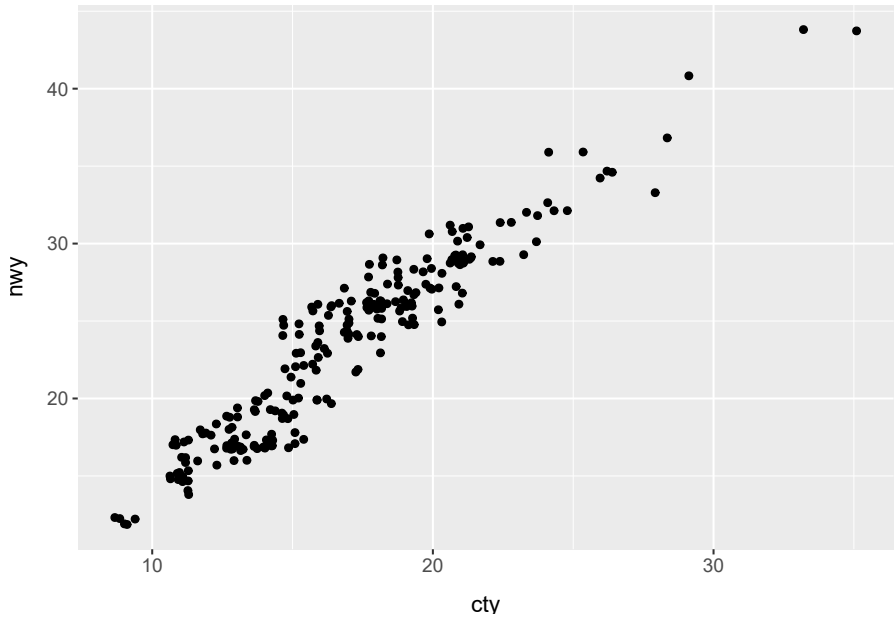
Qual é o problema com este gráfico? Como você poderia melhorá-lo?

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
    geom_point() +
    tema
```



Solução. Há pontos sobrepostos. Uma melhoria poderia ser usar `geom_jitter` em lugar de `geom_point`.

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
    geom_jitter() +
    tema
```



Exercício 1.8.2

Quais parâmetros para `geom_jitter` controlam a quantidade de oscilação?

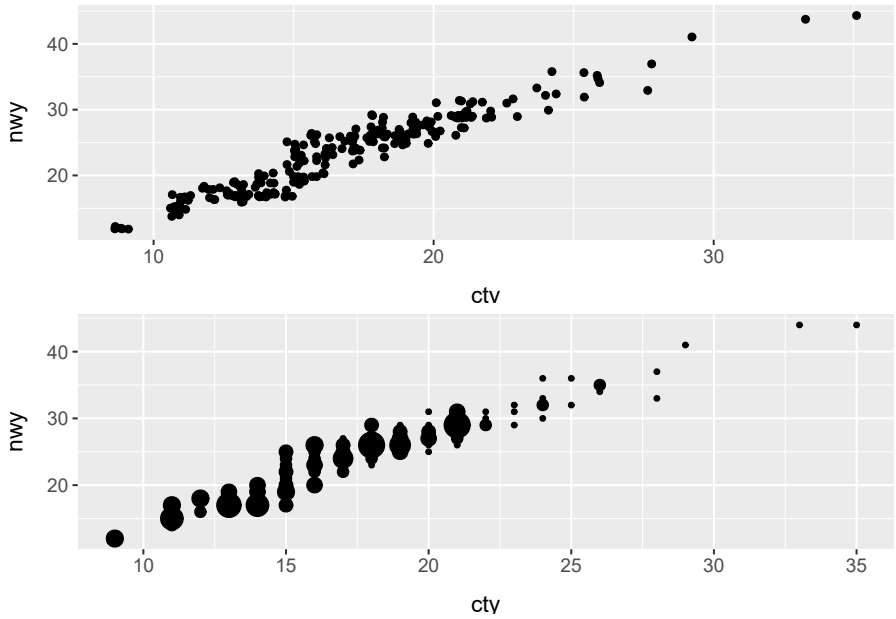
Solução. Conforme a documentação disposta em `?geom_jitter`, são utilizados os parâmetros `width` e `height`.

Exercício 1.8.3

Compare o contraste entre `geom_jitter` e `geom_count`.

Solução.

```
a <- ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
      geom_jitter() +
      tema

b <- ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
      geom_count(show.legend = FALSE) +
      tema

grid.arrange(a, b, nrow = 2)
```



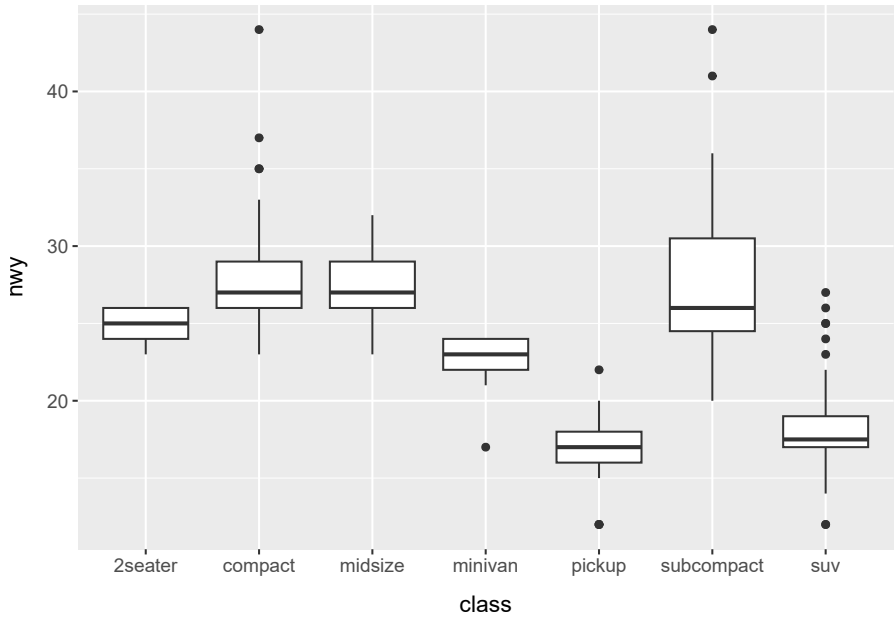
Para contornar o problema da sobreposição de pontos, `geom_jitter` adiciona um pequeno ruído aleatório aos dados, enquanto o `geom_count` contabiliza os pontos sobrepostos e altera o tamanho dos pontos conforme a quantidade.

Exercício 1.8.4

Qual é o ajuste de posição padrão para `geom_boxplot()`? Crie uma visualização do conjunto de dados `mpg` que demonstre isso.

Solução. Conforme pode ser visto em `?geom_boxplot`, a `position` padrão é a `dodge2`.

```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
    geom_boxplot() +
    tema
```



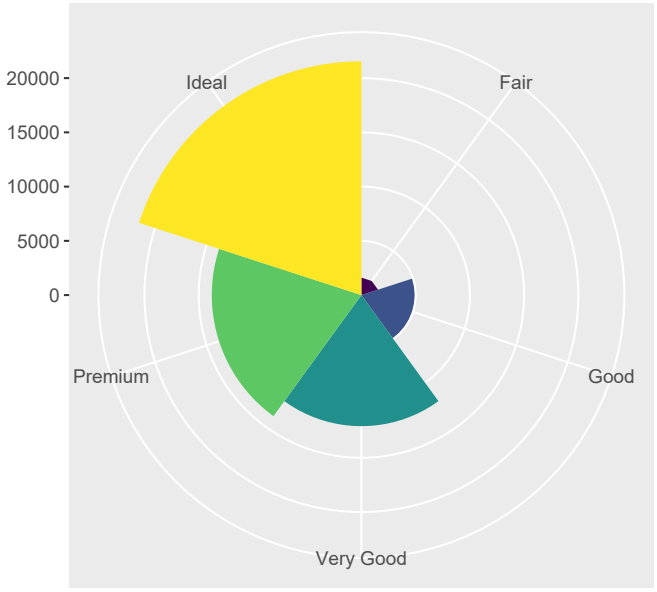
1.9 Sistemas de coordenadas

Exercício 1.9.1

Transforme um gráfico de barras empilhadas em um gráfico de pizza usando `coord_polar()`.

Solução.

```
ggplot(data = diamonds, mapping = aes(x = cut, fill = cut)) +
    geom_bar(show.legend = FALSE, width = 1) +
    coord_polar() +
    labs(x = NULL, y = NULL) +
    theme(aspect.ratio = 1) +
    tema
```



Exercício 1.9.2

O que `labs()` faz? Leia a documentação.

Solução. Usando o comando `?labs`, vimos que esta função é utilizada para definir labels do gráfico, como título, subtítulo, títulos de eixos, etc.

Exercício 1.9.3

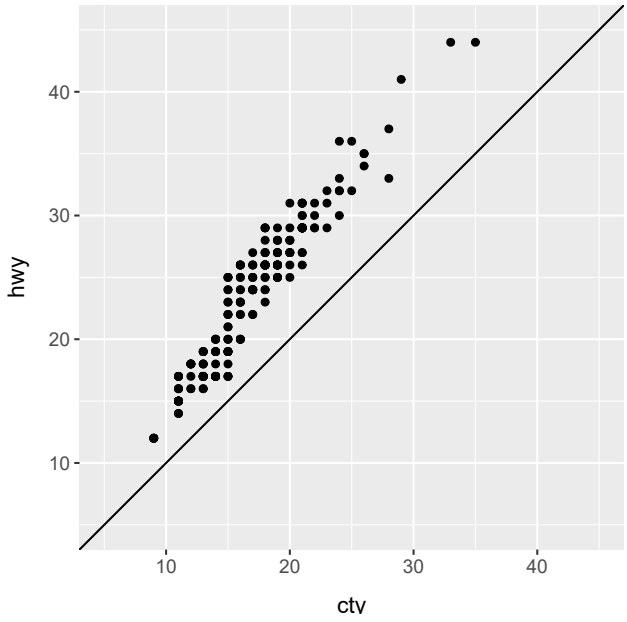
Qual é a diferença entre `coord_quickmap()` e `coord_map()`?

Solução. Usando o comando `?coord_map`, notamos que a diferença é que enquanto `coord_map()` não preserva linhas retas, sendo assim, mais custoso computacionalmente, o `coord_quickmap()` o faz.

Exercício 1.9.4

O que o gráfico a seguir lhe diz sobre a relação entre `mpg` de cidade e estrada? Por que `coord_fixed()` é importante? O que `geom_abline()` faz?

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
    geom_point() +
    geom_abline() +
    coord_fixed(ratio = 1, xlim = c(5, 45), ylim = c(5, 45)) +
    tema
```



Solução. O gráfico mostra a relação entre a eficiência na cidade e na estrada. O `coord_fixed()` força que seja mantida uma proporção entre os eixos x e y, isto é, garante que uma unidade no eixo y corresponda a um número determinado de unidades no eixo x. A razão padrão é 1. Já o `geom_abline()` define uma linha de referência diagonal ao gráfico, no nosso caso, a linha é a reta dada por $y - x = 0$.

1.10 A gramática em camadas de gráficos

Não temos exercícios nesta seção.

2

Fluxo de trabalho: o básico

2.1 O básico de programação

Não temos exercícios nesta seção.

2.2 O que há em um nome?

Não temos exercícios nesta seção.

2.3 Chamando funções

Exercício 2.3.1

Por que esse código não funciona?

```
my_variable <- 10
my_varIable
```

Solução. Foi atribuído um valor à variável `my_variable`, contudo depois tentou-se utilizar essa variável, porém a escrita está incorreta e o `R` não reconheceu a variável. O `R` diferencia letras maiúsculas e minúsculas, isto é, as variáveis `my_variable` e `my_varIable` são distintas.

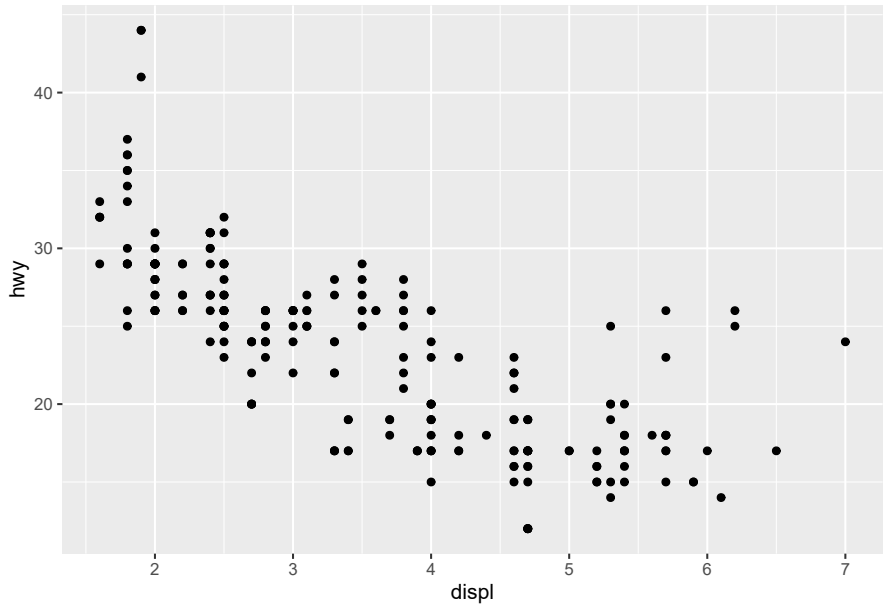
Exercício 2.3.2

Ajuste cada um dos seguintes comandos de `R` para que executem corretamente.

```
library(tidyverse)

ggplot(dota = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy))

filter(mpg, cyl = 8)
filter(diamond, carat > 3)
```

Solução.

```
library(tidyverse)

ggplot(data = mpg) +
    geom_point(mapping = aes(x = displ, y = hwy))
```




```
filter(mpg, cyl == 8)
```

```
## # A tibble: 70 x 11
##    manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
##    <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
##  1 audi         a6 quattro   4.2  2008     8 auto~ 4        16    23 p     mids~
##  2 chevrolet    c1500 sub~   5.3  2008     8 auto~ r        14    20 r     suv
```

```
##  3 chevrolet    c1500 sub~   5.3  2008     8 auto~ r        11    15 e     suv
##  4 chevrolet    c1500 sub~   5.3  2008     8 auto~ r        14    20 r     suv
##  5 chevrolet    c1500 sub~   5.7  1999     8 auto~ r        13    17 r     suv
##  6 chevrolet    c1500 sub~   6    2008     8 auto~ r        12    17 r     suv
##  7 chevrolet    corvette     5.7  1999     8 manu~ r        16    26 p     2sea~
##  8 chevrolet    corvette     5.7  1999     8 auto~ r        15    23 p     2sea~
##  9 chevrolet    corvette     6.2  2008     8 manu~ r        16    26 p     2sea~
## 10 chevrolet    corvette     6.2  2008     8 auto~ r        15    25 p     2sea~
## # i 60 more rows
```

```
filter(diamonds, carat > 3)
```

```
## # A tibble: 32 x 10
##    carat cut     color clarity depth table price     x     y     z
##    <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  3.01 Premium I     I1       62.7    58  8040  9.1   8.97  5.67
##  2  3.11 Fair    J     I1       65.9    57  9823  9.15  9.02  5.98
##  3  3.01 Premium F     I1       62.2    56  9925  9.24  9.13  5.73
##  4  3.05 Premium E     I1       60.9    58 10453  9.26  9.25  5.66
##  5  3.02 Fair    I     I1       65.2    56 10577  9.11  9.02  5.91
##  6  3.01 Fair    H     I1       56.1    62 10761  9.54  9.38  5.31
##  7  3.65 Fair    H     I1       67.1    53 11668  9.53  9.48  6.38
##  8  3.24 Premium H     I1       62.1    58 12300  9.44  9.4   5.85
##  9  3.22 Ideal   I     I1       62.6    55 12545  9.49  9.42  5.92
## 10  3.5  Ideal   H     I1       62.8    57 12587  9.65  9.59  6.03
## # i 22 more rows
```

Exercício 2.3.3

Pressione Alt-Shift-K. O que acontece? Como você pode chegar ao mesmo resultado usando os menus?

Solução. x

3

Transformação de dados com `dplyr`

3.1 Introdução

Não temos exercícios nesta seção.

3.2 Filtrar linhas com `filter()`

Não temos exercícios nesta seção.

3.3 Comparações

Exercício 3.3.1

Encontre todos os voos que:

a. Tiveram um atraso de duas horas ou mais na chegada.
b. Foram para Houston (IAH ou HOU).
c. Foram operados pela United, American ou Delta.
d. Partiram em julho, agosto e setembro.
e. Chegaram com mais de duas horas de atraso, mas não saíram atrasados.
f. Atrasaram pelo menos uma hora, mas compensaram mais de 30 minutos durante o trajeto.
g. Saíram entre meia-noite e 6h (incluindo esses horários).

Solução.

a. Tiveram um atraso de duas horas ou mais na chegada.

```
filter(flights, arr_delay >= 120)
```

```
## # A tibble: 10,200 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      811            630       101     1047            830
##  2  2013     1     1      848           1835       853     1001           1950
##  3  2013     1     1      957            733       144     1056            853
##  4  2013     1     1     1114            900       134     1447           1222
##  5  2013     1     1     1505           1310       115     1638           1431
##  6  2013     1     1     1525           1340       105     1831           1626
##  7  2013     1     1     1549           1445        64     1912           1656
##  8  2013     1     1     1558           1359       119     1718           1515
##  9  2013     1     1     1732           1630        62     2028           1825
## 10  2013     1     1     1803           1620       103     2008           1750
## # i 10,190 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

b. Foram para Houston (IAH ou HOU).

```
filter(flights, dest %in% c("IAH", "HOU"))
```

```
## # A tibble: 9,313 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      623            627        -4      933            932
##  4  2013     1     1      728            732        -4     1041           1038
##  5  2013     1     1      739            739         0     1104           1038
##  6  2013     1     1      908            908         0     1228           1219
##  7  2013     1     1     1028           1026         2     1350           1339
##  8  2013     1     1     1044           1045        -1     1352           1351
##  9  2013     1     1     1114            900       134     1447           1222
## 10  2013     1     1     1205           1200         5     1503           1505
## # i 9,303 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

c. Foram operados pela United, American ou Delta.

```
filter(flights, carrier %in% c("AA", "DL", "UA"))
```

```
## # A tibble: 139,504 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      554            600        -6      812            837
##  5  2013     1     1      554            558        -4      740            728
##  6  2013     1     1      558            600        -2      753            745
##  7  2013     1     1      558            600        -2      924            917
##  8  2013     1     1      558            600        -2      923            937
##  9  2013     1     1      559            600        -1      941            910
## 10  2013     1     1      559            600        -1      854            902
## # i 139,494 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

d. Partiram em julho, agosto e setembro.

```
filter(flights, month %in% c(7, 8, 9))
```

```
## # A tibble: 86,326 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     7     1        1           2029       212      236           2359
##  2  2013     7     1        2           2359         3      344            344
##  3  2013     7     1       29           2245       104      151              1
##  4  2013     7     1       43           2130       193      322             14
##  5  2013     7     1       44           2150       174      300            100
##  6  2013     7     1       46           2051       235      304           2358
##  7  2013     7     1       48           2001       287      308           2305
##  8  2013     7     1       58           2155       183      335             43
##  9  2013     7     1      100           2146       194      327             30
## 10  2013     7     1      100           2245       135      337            135
## # i 86,316 more rows
```

```
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

e. Chegaram com mais de duas horas de atraso, mas não saíram atrasados.

```
filter(flights, dep_delay <= 0, arr_delay > 120)
```

```
## # A tibble: 29 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1    27     1419           1420        -1     1754           1550
##  2  2013    10     7     1350           1350         0     1736           1526
##  3  2013    10     7     1357           1359        -2     1858           1654
##  4  2013    10    16      657            700        -3     1258           1056
##  5  2013    11     1      658            700        -2     1329           1015
##  6  2013     3    18     1844           1847        -3       39           2219
##  7  2013     4    17     1635           1640        -5     2049           1845
##  8  2013     4    18      558            600        -2     1149            850
##  9  2013     4    18      655            700        -5     1213            950
## 10  2013     5    22     1827           1830        -3     2217           2010
## # i 19 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

f. Atrasaram pelo menos uma hora, mas compensaram mais de 30 minutos durante o trajeto.

```
filter(flights, dep_delay >= 60 & dep_delay - arr_delay >= 30)
```

```
## # A tibble: 2,074 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1     1716           1545        91     2140           2039
##  2  2013     1     1     2205           1720       285       46           2040
##  3  2013     1     1     2326           2130       116      131             18
##  4  2013     1     3     1503           1221       162     1803           1555
##  5  2013     1     3     1821           1530       171     2131           1910
```

```
##  6  2013     1     3     1839           1700        99     2056           1950
##  7  2013     1     3     1850           1745        65     2148           2120
##  8  2013     1     3     1923           1815        68     2036           1958
##  9  2013     1     3     1941           1759       102     2246           2139
## 10  2013     1     3     1950           1845        65     2228           2227
## # i 2,064 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

g. Saíram entre meia-noite e 6h (incluindo esses horários).

```
filter(flights, dep_time >= 0, dep_time <= 600)
```

```
## # A tibble: 9,344 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 9,334 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Exercício 3.3.2

Outro ajudante da filtragem do **dplyr** é `between()`. O que ele faz? Você consegue utilizá-lo para simplificar o código necessário para responder os desafios anteriores?

Solução. O `between` recebe três parâmetros e verifica se o primeiro está entre o segundo e o terceiro.

```
filter(flights, between(dep_time, 0, 600))
```

```
## # A tibble: 9,344 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 9,334 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Exercício 3.3.3

Quantos voos têm um `dep_time` faltante? Que outras variáveis estão faltando? O que essas linhas podem representar?

Solução.

```
count(flights, is.na(dep_time))
```

```
## # A tibble: 2 x 2
##   `is.na(dep_time)`      n
##   <lgl>              <int>
## 1 FALSE             328521
## 2 TRUE                8255
```

```
summary(is.na(flights))
```

```
##     year            month             day            dep_time
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
```

```
##  FALSE:336776    FALSE:336776    FALSE:336776    FALSE:328521
##                                                  TRUE :8255
##  sched_dep_time  dep_delay        arr_time       sched_arr_time
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:336776    FALSE:328521    FALSE:328063    FALSE:336776
##                  TRUE :8255      TRUE :8713
##  arr_delay        carrier          flight         tailnum
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:327346    FALSE:336776    FALSE:336776    FALSE:334264
##  TRUE :9430                                      TRUE :2512
##    origin           dest          air_time        distance
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:336776    FALSE:336776    FALSE:327346    FALSE:336776
##                                  TRUE :9430
##     hour            minute       time_hour
##  Mode :logical   Mode :logical   Mode :logical
##  FALSE:336776    FALSE:336776    FALSE:336776
##
```

São 8255 voos com `dep_time` faltante, o que pode indicar voos cancelados. As seguintes colunas também possuem dados faltantes: `dep_delay`, `arr_time`, `arr_delay`, `tailnum` e `air_time`.

Exercício 3.3.4

Por que `NA ^ 0` não é um valor faltante? Por que `NA | TRUE` não é um valor faltante? Por que `FALSE & NA` não é um valor faltante? Você consegue descobrir a regra geral? (`NA * 0` é um contraexemplo complicado!)

Solução. `NA ^ 0` resulta em um, pois qualquer número real satisfaz essa mesma condição. A regra geral parece ser que, ao avaliar a expressão, sempre que o valor que `NA` representaria for indiferente para o resultado da expressão, então será retornado um valor diferente de `NA`.

3.4 Ordenar linhas com `arrange()`

Exercício 3.4.1

Como você poderia usar `arrange()` para classificar todos os valores faltantes no começo? (dica: use `is.na()`.)

Solução.

```
arrange(
  flights,
  !is.na(year),
  !is.na(month),
  !is.na(day),
  !is.na(dep_time),
  !is.na(sched_dep_time),
  !is.na(dep_delay),
  !is.na(arr_time),
  !is.na(sched_arr_time),
  !is.na(arr_delay),
  !is.na(carrier),
  !is.na(flight),
  !is.na(tailnum),
  !is.na(origin),
  !is.na(dest),
  !is.na(air_time),
  !is.na(distance),
  !is.na(hour),
  !is.na(minute),
  !is.na(time_hour)
)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     2       NA           1545        NA       NA           1910
##  2  2013     1     2       NA           1601        NA       NA           1735
##  3  2013     1     3       NA            857        NA       NA           1209
##  4  2013     1     3       NA            645        NA       NA            952
##  5  2013     1     4       NA            845        NA       NA           1015
##  6  2013     1     4       NA           1830        NA       NA           2044
##  7  2013     1     5       NA            840        NA       NA           1001
##  8  2013     1     7       NA            820        NA       NA            958
##  9  2013     1     8       NA           1645        NA       NA           1838
## 10  2013     1     9       NA            755        NA       NA           1012
## # i 336,766 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Observação. Deve haver uma solução muito mais elegante para este problema.

Exercício 3.4.2

Ordene `flights` para encontrar os voos mais atrasados. Encontre os voos que saíram mais cedo.

Solução. Voos mais atrasados:

```
arrange(flights, desc(dep_delay))
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     9      641            900      1301     1242           1530
##  2  2013     6    15     1432           1935      1137     1607           2120
##  3  2013     1    10     1121           1635      1126     1239           1810
##  4  2013     9    20     1139           1845      1014     1457           2210
##  5  2013     7    22      845           1600      1005     1044           1815
##  6  2013     4    10     1100           1900       960     1342           2211
##  7  2013     3    17     2321            810       911      135           1020
##  8  2013     6    27      959           1900       899     1236           2226
##  9  2013     7    22     2257            759       898      121           1026
## 10  2013    12     5      756           1700       896     1058           2020
## # i 336,766 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Voos que saíram mais cedo:

```
arrange(flights, dep_time)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1    13        1           2249        72      108           2357
##  2  2013     1    31        1           2100       181      124           2225
##  3  2013    11    13        1           2359         2      442            440
##  4  2013    12    16        1           2359         2      447            437
##  5  2013    12    20        1           2359         2      430            440
##  6  2013    12    26        1           2359         2      437            440
##  7  2013    12    30        1           2359         2      441            437
##  8  2013     2    11        1           2100       181      111           2225
```

```
##  9  2013     2    24        1           2245        76      121           2354
## 10  2013     3     8        1           2355         6      431            440
## # i 336,766 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Exercício 3.4.3

Ordene `flights` para encontrar os voos mais rápidos.

Solução.

```
arrange(flights, air_time)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1    16     1355           1315        40     1442           1411
##  2  2013     4    13      537            527        10      622            628
##  3  2013    12     6      922            851        31     1021            954
##  4  2013     2     3     2153           2129        24     2247           2224
##  5  2013     2     5     1303           1315       -12     1342           1411
##  6  2013     2    12     2123           2130        -7     2211           2225
##  7  2013     3     2     1450           1500       -10     1547           1608
##  8  2013     3     8     2026           1935        51     2131           2056
##  9  2013     3    18     1456           1329        87     1533           1426
## 10  2013     3    19     2226           2145        41     2305           2246
## # i 336,766 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Exercício 3.4.4

Quais voos viajaram por mais tempo? Quais viajaram por menos tempo?

Solução. Voos que viajaram por mais tempo:

```
arrange(flights, desc(air_time))
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     3    17     1337           1335         2     1937           1836
##  2  2013     2     6      853            900        -7     1542           1540
##  3  2013     3    15     1001           1000         1     1551           1530
##  4  2013     3    17     1006           1000         6     1607           1530
##  5  2013     3    16     1001           1000         1     1544           1530
##  6  2013     2     5      900            900         0     1555           1540
##  7  2013    11    12      936            930         6     1630           1530
##  8  2013     3    14      958           1000        -2     1542           1530
##  9  2013    11    20     1006           1000         6     1639           1555
## 10  2013     3    15     1342           1335         7     1924           1836
## # i 336,766 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Voos que viajaram por menos tempo:

```
arrange(flights, air_time)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1    16     1355           1315        40     1442           1411
##  2  2013     4    13      537            527        10      622            628
##  3  2013    12     6      922            851        31     1021            954
##  4  2013     2     3     2153           2129        24     2247           2224
##  5  2013     2     5     1303           1315       -12     1342           1411
##  6  2013     2    12     2123           2130        -7     2211           2225
##  7  2013     3     2     1450           1500       -10     1547           1608
##  8  2013     3     8     2026           1935        51     2131           2056
##  9  2013     3    18     1456           1329        87     1533           1426
## 10  2013     3    19     2226           2145        41     2305           2246
## # i 336,766 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

3.5 Selecionar colunas com `select()`

Exercício 3.5.1

Faça um *brainstorm* da maior quantidade possível de maneiras de selecionar `dep_time`, `dep_delay`, `arr_time` e `air_delay` de `flights`.

Solução. x

Exercício 3.5.2

O que acontece se você incluir o nome de uma variável varias vezes em uma chamada `select()`?

Solução.

```
select(flights, arr_time, arr_time, arr_time)
```

```
## # A tibble: 336,776 x 1
##    arr_time
##       <int>
##  1      830
##  2      850
##  3      923
##  4     1004
##  5      812
##  6      740
##  7      913
##  8      709
##  9      838
## 10      753
## # i 336,766 more rows
```

A variável em questão é selecionada apenas uma vez.

Exercício 3.5.3

O que a função `one_of()` faz? Por que poderia ser útil em conjunção com este vetor?

```
vars <- c("year", "month", "day", "dep_delay", "arr_delay")
```

Solução.

```
vars <- c("year", "month", "day", "dep_delay", "arr_delay")
select(flights, one_of(vars)) # superseded in favor of `any_of()`
```

```
## # A tibble: 336,776 x 5
##     year month   day dep_delay arr_delay
##    <int> <int> <int>     <dbl>     <dbl>
##  1  2013     1     1         2        11
##  2  2013     1     1         4        20
##  3  2013     1     1         2        33
##  4  2013     1     1        -1       -18
##  5  2013     1     1        -6       -25
##  6  2013     1     1        -4        12
##  7  2013     1     1        -5        19
##  8  2013     1     1        -3       -14
##  9  2013     1     1        -3        -8
## 10  2013     1     1        -2         8
## # i 336,766 more rows
```

A função `one_of()`, substituída por `any_of()` serve para indicar que devem ser selecionadas todas as colunas cujos nomes estejam no *array*.

Exercício 3.5.4

O resultado ao executar o código a seguir lhe surpreende? Como as funções auxiliares lidam com o caso por padrão? Como você pode mudar esse padrão?

```
select(flights, contains("TIME"))
```

Solução.

```
select(flights, contains("TIME"))
```

```
## # A tibble: 336,776 x 6
##    dep_time sched_dep_time arr_time sched_arr_time air_time time_hour
##       <int>          <int>    <int>          <int>    <dbl> <dttm>
##  1      517            515      830            819      227 2013-01-01 05:00:00
##  2      533            529      850            830      227 2013-01-01 05:00:00
##  3      542            540      923            850      160 2013-01-01 05:00:00
```

```
##  4      544           545     1004           1022      183 2013-01-01 05:00:00
##  5      554           600      812            837      116 2013-01-01 06:00:00
##  6      554           558      740            728      150 2013-01-01 05:00:00
##  7      555           600      913            854      158 2013-01-01 06:00:00
##  8      557           600      709            723       53 2013-01-01 06:00:00
##  9      557           600      838            846      140 2013-01-01 06:00:00
## 10      558           600      753            745      138 2013-01-01 06:00:00
## # i 336,766 more rows
```

O caso não surpreende. São retornadas todas as colunas que possuem "TIME" em seus nomes, não diferenciando maíusculas e minúsculas. O comportamento pode ser alterado da seguinte forma:

```
select(flights, contains("TIME", ignore.case = FALSE))
```

```
## # A tibble: 336,776 x 0
```

3.6 Adicionar novas variáveis com `mutate()`

Exercício 3.6.1

Atualmente, `dep_time` e `sched_dep_time` são convenientes para observar, mas difíceis de usar para calcular, porque não são realmente números contínuos. Converta-os para uma representação mais apropriada do número de minutos desde a meia-noite.

Solução.

```
(flights_min <- mutate(
                    flights,
                    dep_time_minutes = 60 * (dep_time %/% 100) + (dep_time %% 100),
                    sched_dep_time_minutes = 60 * (sched_dep_time %/% 100) + (sched_dep_time %% 100),
                    arr_time_minutes = 60 * (arr_time %/% 100) + (arr_time %% 100)
                ))
```

```
## # A tibble: 336,776 x 22
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
```

```
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 336,766 more rows
## # i 14 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>, dep_time_minutes <dbl>,
## #   sched_dep_time_minutes <dbl>, arr_time_minutes <dbl>
```

Exercício 3.6.2

Compare `air_time` e `arr_time - dep_time`. O que você espera ver? O que você vê? O que você precisa fazer para corrigir isso?

Solução.

```
transmute(flights_min, air_time, arr_time_minutes - dep_time_minutes)
```

```
## # A tibble: 336,776 x 2
##    air_time `arr_time_minutes - dep_time_minutes`
##       <dbl>                                 <dbl>
##  1      227                                   193
##  2      227                                   197
##  3      160                                   221
##  4      183                                   260
##  5      116                                   138
##  6      150                                   106
##  7      158                                   198
##  8       53                                    72
##  9      140                                   161
## 10      138                                   115
## # i 336,766 more rows
```

Como os valores `arr_time` e `dep_time` não são números de fato, a diferença não faz sentido e assim o cálculo gera uma diferença muito grande. Para corrigir isso, primeiro teremos que converter os valores dessas duas variáveis para o número de minutos

desde a meia noite e, depois, efetuar a diferença. Ainda assim, pode haver divergência entre esse valor e `air_time`, que pode ser explicada por chegada antecipada, saída atrasada ou porque um vôo chegou ao seu destino após a meia-noite.

Exercício 3.6.3

Compare `dep_time`, `sched_dep_time` e `dep_delay`. Como você espera que esses números estejam relacionados?

Solução.

```
select(flights_min, "dep_time", "sched_dep_time", dep_delay)
```

```
## # A tibble: 336,776 x 3
##    dep_time sched_dep_time dep_delay
##       <int>          <int>     <dbl>
##  1      517            515         2
##  2      533            529         4
##  3      542            540         2
##  4      544            545        -1
##  5      554            600        -6
##  6      554            558        -4
##  7      555            600        -5
##  8      557            600        -3
##  9      557            600        -3
## 10      558            600        -2
## # i 336,766 more rows
```

É esperado que `dep_time = sched_dep_time + dep_delay`.

Exercício 3.6.4

Encontre os 10 voos mais atrasados usando uma função de classificação. Como você quer lidar com empates? Leia cuidadosamente a documentação de `min_rank()`.

Solução.

```
filter(
    flights,
    between(rank(desc(flights$dep_delay), ties.method = "min"), 1, 10)
)
```

```
## # A tibble: 10 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     9      641            900      1301     1242           1530
##  2  2013     1    10     1121           1635      1126     1239           1810
##  3  2013    12     5      756           1700       896     1058           2020
##  4  2013     3    17     2321            810       911      135           1020
##  5  2013     4    10     1100           1900       960     1342           2211
##  6  2013     6    15     1432           1935      1137     1607           2120
##  7  2013     6    27      959           1900       899     1236           2226
##  8  2013     7    22      845           1600      1005     1044           1815
##  9  2013     7    22     2257            759       898      121           1026
## 10  2013     9    20     1139           1845      1014     1457           2210
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Usei a função `rank` e os empates foram tratados com o parâmetro `ties.method` setado como `min`.

Exercício 3.6.5

O que `1:3 + 1:10` retorna? Por quê?

Solução.

```
1:3 + 1:10
```

```
## Warning in 1:3 + 1:10: comprimento do objeto maior não é múltiplo do
## comprimento do objeto menor
```

```
##  [1]  2  4  6  5  7  9  8 10 12 11
```

Como os vetores têm tamanhos diferentes, a soma vai ser executada entre as posições e, quando o menor dos vetores tiver sido completamente consumido, será tomado novamente o primeiro elemento (como em um movimento circular).

Exercício 3.6.6

Quais funções trigonométricas o R fornece?

Solução. Utilizamos o comando `?cos` para chegar até a documentação do pacote `Trig`, um dos componentes da base do R.

O R fornece as funções `cos(x)`, `sin(x)`, `tan(x)`, `acos(x)`, `asin(x)`, `atan(x)`, `atan2(y, x)` (arco tangente entre dois vetores), `cospi(x)`, `sinpi(x)` e `tanpi(x)`.

3.7 Resumos agrupados com `summarize()`

Exercício 3.7.1

Faça um *brainstorming* de pelo menos cinco maneiras diferentes de avaliar as características do atraso típico de um grupo de voos. Considere os seguintes cenários:

- Um voo está 15 minutos adiantado em 50% do tempo e 15 minutos atrasado em 50% do tempo.
- Um voo está sempre 10 min atrasado.
- Um voo está 30 minutos adiantado em 50% do tempo e 30 minutos atrasado em 50% do tempo.
- Em 99% do tempo um voo está no horário. Em 1% do tempo, está 2 horas atrasado.

O que é mais importante: atrsado na chegada ou atraso na partida?

Solução. x

Exercício 3.7.2

Crie outra abordagem que lhe dará o mesmo resultado que `not_cancelled %>% count(dest)` e `not_cancelled %>% count(tailnum, wt = distance)` (sem usar `count()`).

Solução.

```
not_cancelled %>%
    group_by(dest) %>%
    summarise(n = n())
```

```
## # A tibble: 104 x 2
##    dest      n
##    <chr> <int>
##  1 ABQ     254
##  2 ACK     264
```

```
##  3 ALB     418
##  4 ANC       8
##  5 ATL   16837
##  6 AUS    2411
##  7 AVL     261
##  8 BDL     412
##  9 BGR     358
## 10 BHM     269
## # i 94 more rows
```

```
not_cancelled %>%
    group_by(tailnum) %>%
    summarise(n = sum(distance))
```

```
## # A tibble: 4,037 x 2
##    tailnum      n
##    <chr>    <dbl>
##  1 D942DN    3418
##  2 N0EGMQ  239143
##  3 N10156  109664
##  4 N102UW   25722
##  5 N103US   24619
##  6 N104UW   24616
##  7 N10575  139903
##  8 N105UW   23618
##  9 N107US   21677
## 10 N108UW   32070
## # i 4,027 more rows
```

Exercício 3.7.3

Nossa definição de voos cancelados (`is.na(dep_delay) | is.na(arr_delay)`) é ligeiramente insuficiente. Por quê? Qual é a coluna mais importante?

Solução. As váriáveis `dep_delay` e `arr_delay` se referem ao atraso na partida ou na chegada dos voos. Caso um voo tenha saído e chegado no horário exato, esses valores podem estar `NA`, ou seja, o voo não foi cancelado, apenas partiu e chegou no horário planejado. Nesse caso, o mais correto seria considerar como cancelados os voos `dep_time` é `NA`.

Exercício 3.7.4

Veja o número de voos cancelados por dia. Existe um padrão? A proporção de voos cancelados está relacionado ao atraso médio?

Solução.

```
cancelled_by_day <- flights %>%
                        group_by(year, month, day) %>%
                        summarise(
                            date = as.Date(paste(year, month, day, sep='-')),
                            count = n(),
                            count_cancelled = sum(is.na(dep_time)),
                            count_not_cancelled = sum(!is.na(dep_time)),
                            mean_dep_delay = mean(dep_delay, na.rm = TRUE),
                            mean_arr_delay = mean(arr_delay, na.rm = TRUE),
                        )
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

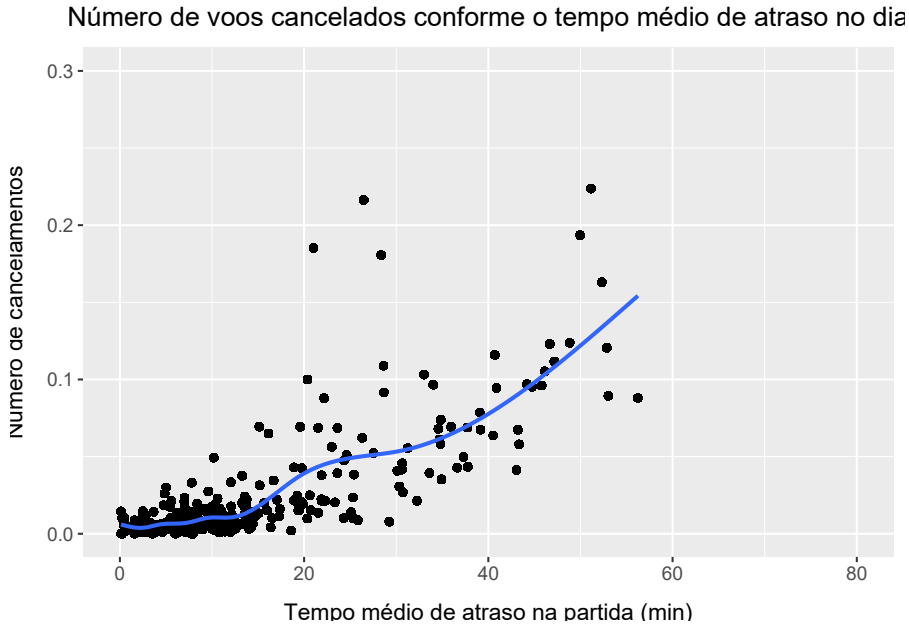
```
## `summarise()` has grouped output by 'year', 'month', 'day'. You can override
## using the `.groups` argument.
```

```
cancelled_by_day %>%
    ggplot(aes(mean_dep_delay, count_cancelled / count)) +
        geom_point() +
        geom_smooth(se = FALSE) +
        labs(
            title = "Número de voos cancelados conforme o tempo médio de atraso no dia",
            x = "Tempo médio de atraso na partida (min)",
            y = "Número de cancelamentos"
        ) +
        xlim(0, 80) +
        ylim(0, 0.3) +
        tema
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 12409 rows containing non-finite values (`stat_smooth()`).
```

```
## Warning: Removed 12409 rows containing missing values (`geom_point()`).
```



Parece existir uma relação entre o número de coos cancelados no dia e a média de atraso nos voos desse mesmo dia. Caso haja alguma condição desfavorável (tempo ruim, problemas na pista de decolagem/pouso, etc), o intervalo entre uma decolagem/pouso e outro pode aumentar significativamente gerando atrasos que se acumulam a ponto de alguns voos terem que ser cancelados (esse comportamento é real?).

Exercício 3.7.5

Qual companhia tem os piores atrasos? Desafio: você consegue desembaralhar o efeito dos aeroportus ruins *versus* companhiars ruins? Por quê/Por que não? (Dica: pense em `flights %>% group_by(cartier, dest) %>% summarize(n())`)

Solução. Para verificar qual companhia tem os piores atrasos, vamos calcular o atraso médio por companhia.

```
flights %>%
    group_by(carrier) %>%
    summarize(
        mean_delay = mean(arr_delay, na.rm = TRUE)
    ) %>%
    arrange(desc(mean_delay))
```

```
## # A tibble: 16 x 2
##    carrier mean_delay
##    <chr>        <dbl>
##  1 F9          21.9
##  2 FL          20.1
##  3 EV          15.8
##  4 YV          15.6
##  5 OO          11.9
##  6 MQ          10.8
##  7 WN           9.65
##  8 B6           9.46
##  9 9E           7.38
## 10 UA           3.56
## 11 US           2.13
## 12 VX           1.76
## 13 DL           1.64
## 14 AA           0.364
## 15 HA          -6.92
## 16 AS          -9.93
```

Podemos notar que a companhia com o maior atraso médio é a F9 (Frontier Airlines Inc).

Para tentar desembaralhar o efeito de aeroportos ruins e companhias ruins, vamos:

- filtrar apenas os voos com atraso;
- agrupar os voos conforme as rotas e companhias;
- calcular o atraso médio e o total de voos por companhia no trecho (`arr_delay` e `flights`);
- calcular o atraso médio e o total de voos do trecho de todas as companhias (`arr_delay_total` e `flights_total`);
- calcular o atraso médio por voo da companhia (`arr_delay_mean <- arr_delay / flights`);
- calcular o atraso "médio" das demais companhias (`arr_delay_others <- (arr_delay_total - arr_delay) / (flights_total - fligths)`);
- calcular a diferença entre o atraso médio da companhia e o atraso médio das outras companhias juntas (`arr_delay_diff <- arr_delay_mean - arr_delay_others`);

- remover valores cuja diferença não faça sentido (`is.finite(arr_delay_diff)`);
- agrupar por companhia;
- calcular a média das diferenças de atraso da companhia (`arr_delay_diff`);

```
(atrasos <- flights %>%
                filter(!is.na(arr_delay)) %>%
                group_by(origin, dest, carrier) %>%
                summarise(
                    arr_delay = sum(arr_delay),
                    flights = n()
                ) %>%
                group_by(origin, dest) %>%
                mutate(
                    arr_delay_total = sum(arr_delay),
                    flights_total = sum(flights)
                ) %>%
                ungroup() %>%
                mutate(
                    arr_delay_mean = arr_delay / flights, # atraso médio da companhia
                    arr_delay_others = (arr_delay_total - arr_delay) / (flights_total - flights), # atraso médio
                    arr_delay_diff = arr_delay_mean - arr_delay_others # diferença do atraso em relação às demai
                ) %>%
                filter(is.finite(arr_delay_diff)) %>%
                group_by(carrier) %>%
                summarise(
                    arr_delay_diff = mean(arr_delay_diff)
                ) %>%
                arrange(desc(arr_delay_diff)))
```

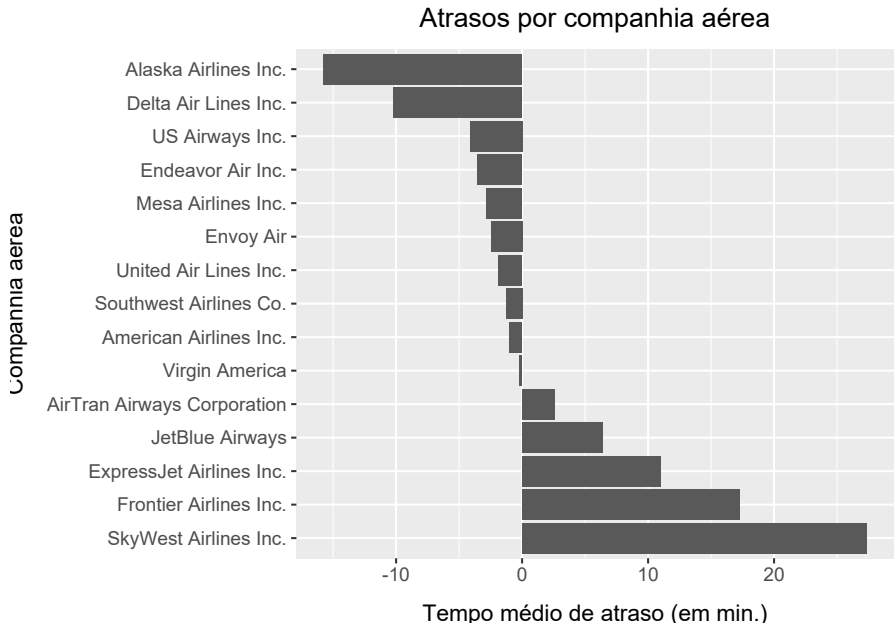
```
## `summarise()` has grouped output by 'origin', 'dest'. You can override using
## the `.groups` argument.
```

```
## # A tibble: 15 x 2
##    carrier arr_delay_diff
##    <chr>            <dbl>
##  1 OO              27.3
##  2 F9              17.3
##  3 EV              11.0
##  4 B6               6.41
##  5 FL               2.57
##  6 VX              -0.202
##  7 AA              -0.970
```

```
##  8 WN              -1.27
##  9 UA              -1.86
## 10 MQ              -2.48
## 11 YV              -2.81
## 12 9E              -3.54
## 13 US              -4.14
## 14 DL             -10.2
## 15 AS             -15.8
```

Desconsiderando o efeito de trechos e aeroportos ruins, a companhia com maior atraso é a OO (SkyWest Airlines Inc.).

```
atrasos %>%
    left_join(airlines, by = "carrier") %>%
    ggplot(aes(
        arr_delay_diff,
        reorder(name, desc(arr_delay_diff))
    )) +
        geom_col() +
        labs(
            title = "Atrasos por companhia aérea",
            y = "Companhia aérea",
            x = "Tempo médio de atraso (em min.)"
        ) +
        tema
```



Exercício 3.7.6

Para cada avião, conte o número de voos antes do primeiro atraso de mais de uma hora.

Solução. Utilizando a variável `flight` para identificar o voo e a variável `arr_delay` como parâmetro para determinar o tempo de atraso:

- ordenamos o data-frame conforme a hora agendada para decolagem;
- agrupamos pelo número do voo;
- utilizamos as funções `first()` e `which()` para buscar a posição do primeiro elemento que é `NA` ou o atraso é maior do que 60 min.

Obs.: `NA` indica que aquele voo não teve nenhum atraso superior a 60 min.

```
flights %>%
    arrange(time_hour) %>%
    group_by(flight) %>%
    summarise(
        first_delay_pos = first(which(is.na(arr_delay) | arr_delay > 60)) - 1
    )
```

```
## # A tibble: 3,844 x 2
##    flight first_delay_pos
##     <int>           <dbl>
##  1      1              47
##  2      2              NA
##  3      3               9
##  4      4              77
##  5      5              11
##  6      6              23
##  7      7              17
##  8      8              15
##  9      9              12
## 10     10              24
## # i 3,834 more rows
```

Exercício 3.7.7

O que o argumento `sort` para `count()` faz? Quando você pode usá-lo?

Solução. Utilizando o comando `?count`, identificamos que o argumento `sort` organiza a contagem em ordem decrescente.

3.8 Mudanças agrupadas (e filtros)

Exercício 3.8.1

x

Solução. x

Exercício 3.8.2

x

Solução. x

Exercício 3.8.3

x

Solução. x

Exercício 3.8.4

x

Solução. x

Exercício 3.8.5

x

Solução. x

Exercício 3.8.6

x

Solução. x

Exercício 3.8.7

x

Solução. x

4

Fluxo de trabalho: scripts

4.1 Executando códigos

4.2 Diagnósticos Rstudio

5

Análise exploratória de dados

6

Fluxo de trabalho: projetos

Parte II

Wrangle

7

Tibbles com `tibble`

7.1 Introdução

7.2 Criando tibbles

7.3 Tibbles *versus* `data.frame`

7.4 Interagindo com códigos mais antigos

8

Importando dados com `readr`

9

Arrumando dados com `tidyr`

10

Dados relacionais com `dplyr`

11

Strings com `stringr`

12

Fatores com `forcats`

13

Datas e horas com `lubridate`

Parte III

Programar

14

Pipes com `magrittr`

14.1 Introdução

14.2 Alternativas ao piping

14.3 Quando não usar o pipe

14.4 Outras ferramentas do `magrittr`

15

Funções

16

Vetores

17

Iteração com `purrr`

18

(PART) Modelar

19

O básico de modelos com `modelr`

20

Construção de modelos

21

Muitos modelos com `purrr` *e* `broom`

Parte IV

Comunicar

22

R Markdown

23

Gráficos para comunicação com `ggplot2`

24

Formatos R Markdown

24.1 Introdução

24.2 Opções de saída

24.3 Documentos

24.4 Notebooks

24.5 Apresentações

24.6 Dashboards

24.7 Interatividade

24.8 Sites

24.9 Outros formatos

24.10 Aprendendo mais

25

Fluxo de trabalho de R Markdown

Bibliografia

Hadley Wickham and Garrett Grolemund. *R para Data Science*. Alta Books, Rio de Janeiro, 2019.