Exercício Avaliativo - Recursos Computacionais

Parte I (ggplot2)

Antonio Mendes Magalhães Junior

9 de Abril de 2019

Parte I

Resolva os seguintes exercícios propostos em Wickham e Grolemund (2016):

- Seção 3.2.4: Ex. 4 e 5
- Seção 3.3.1: Ex. 3, 4, 5 e 6
- Seção 3.5.1: Ex. 3,4,5 e 6
- Seção 3.6.1: Ex. 5 e 6
- Seção 3.7.1: Ex. 1, 2 e 5
- Seção 3.8.1: Ex. 1,2 e 3
- Seção 3.9.1: Ex. 1
- Seção 28.2.1: Ex. 1
- Seção 28.3.1: Ex. 3
- Seção 28.4.4: Ex. 2 e 4

Prazo máximo de entrega: **13h do dia 15/04/2019**

OBS: O relatório com a resolução dos exercícios deverão ser feito em Rmarkdown, formato pdf e deverão ser enviado por e-mail (izabela.oliveira@ufla.br).

Seção 3.2.4

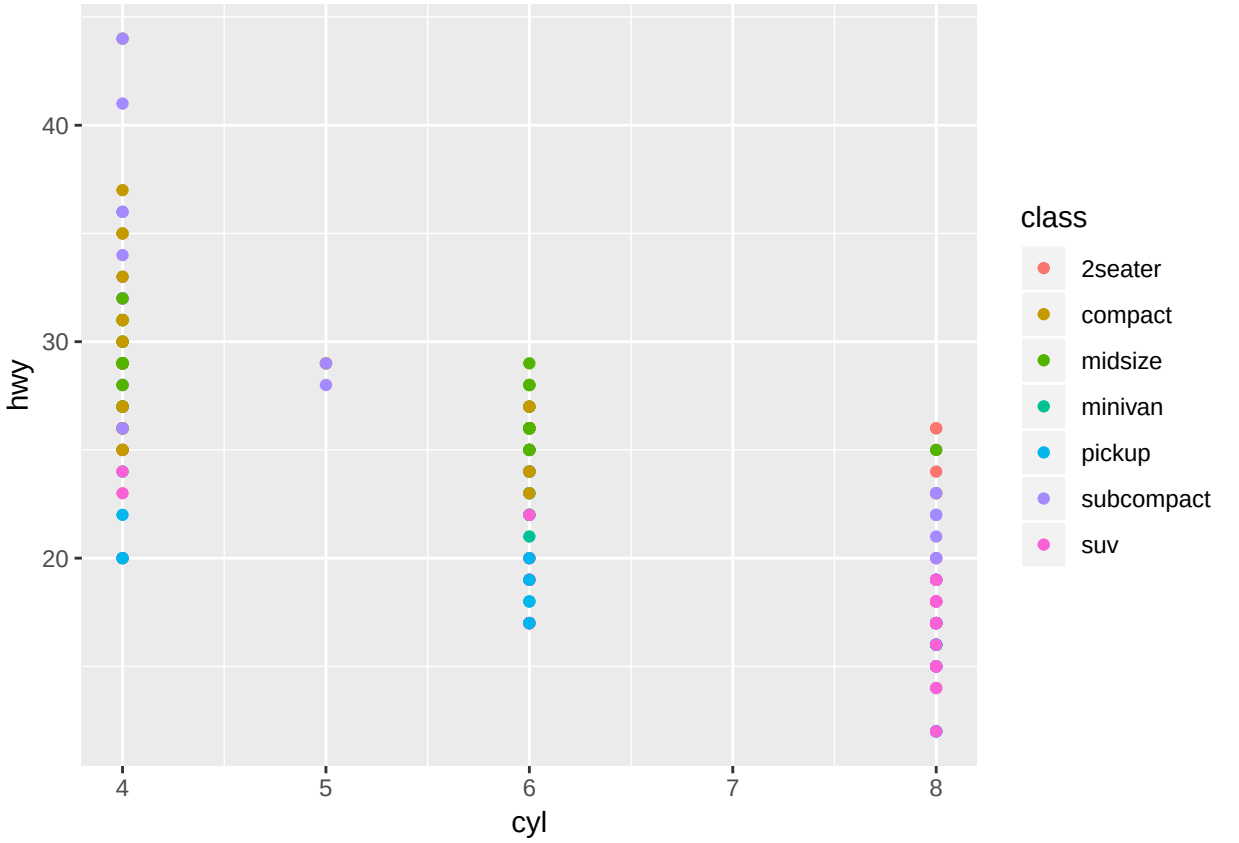
Exercício 4

P: Faça um gráfico de dispersão de `hwy` vs `cyl`.

R:

```
library(tidyverse);
```

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

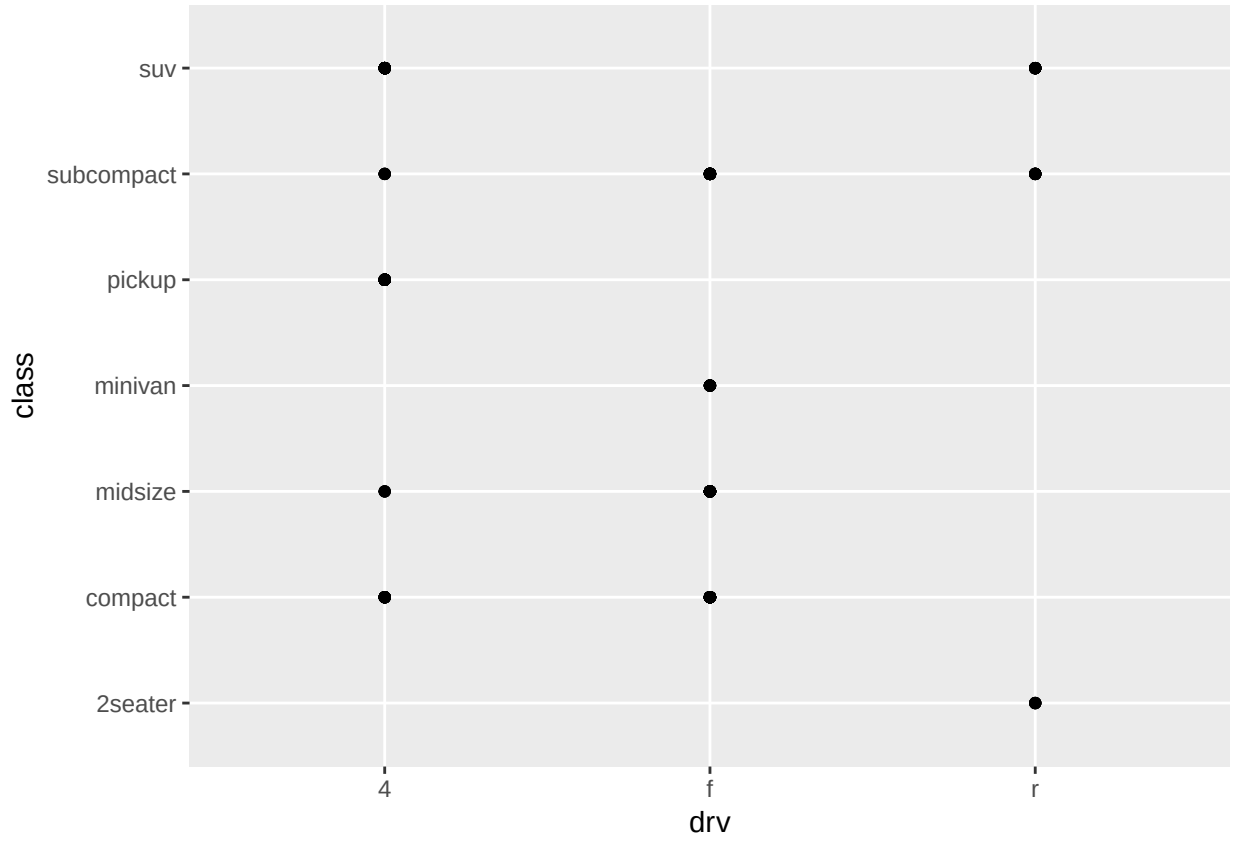


Exercício 5

P: O que acontece se plotar um gráfico de dispersão de `class` vs `drv`? Por que esse gráfico não é útil?

R: Um grafico de dispersão não é útil neste caso porque `class` e `drv` são variáveis categóricas. O gráfico mostra apenas as combinações existentes entre essas variáveis nesse conjunto de dados, como por exemplo a existência de SUV's com tração nas quatro rodas.

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


suv
subcompact
pickup
minivan
midsize
compact
2seater
class
4
f
r
drv

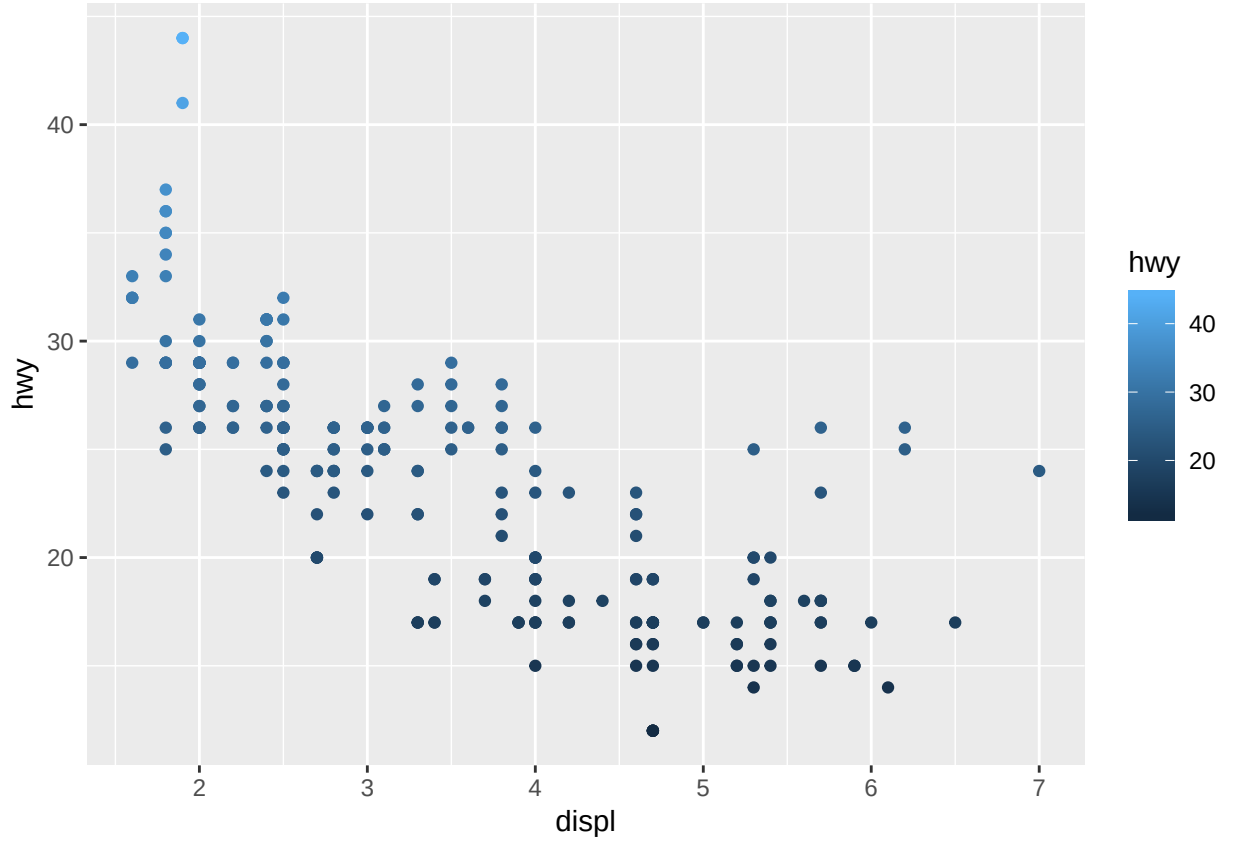
Seção 3.3.1

Exercício 3

P: Mapeie uma variável contínua para cor, tamanho e forma. Como essas estíticas se comportam de maneira diferente para variáveis categóricas e variáveis contínuas?

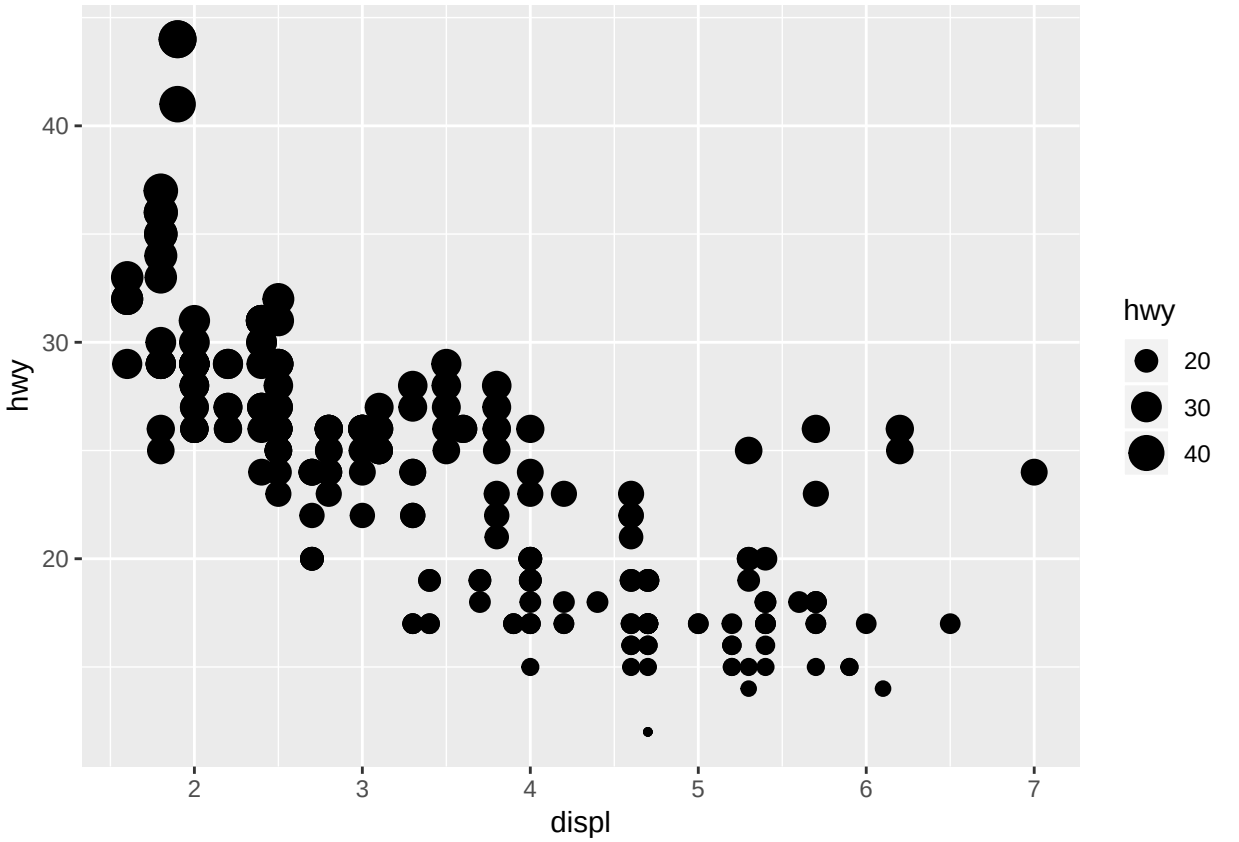
R: Quando se usa `color` para mapear os valores da variável contínua `hwy`, o `ggplot2` cria uma escala gradiente de cores para representar a variável, indo de uma cor escura (menores valores) até uma cor clara (maiores valores).

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



Isso é semelhante quando se mapeia por `size`, onde o `ggplot2` cria círculos de diferentes tamanhos para representar os valores, em que os círculos menores representam os menores valores e os círculos maiores representam os maiores valores.

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



Já o mapeamento por forma (`shape`) não é possível para variáveis contínuas, assim o `ggplot2` retorna uma mensagem de erro.

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

## Error: A continuous variable can not be mapped to shape
```

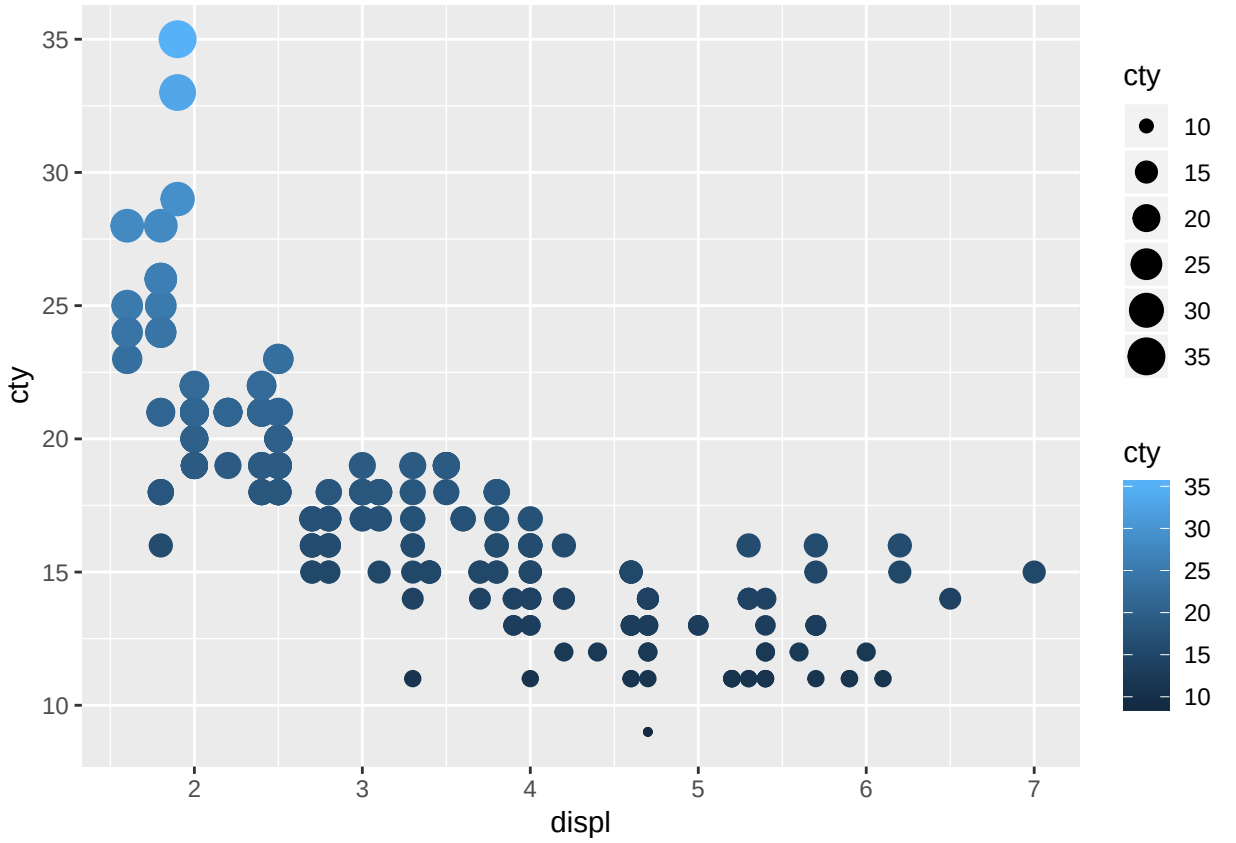
Exercício 4

P: O que acontece se você mapear a mesma variável para múltiplas estéticas?

R: É possível mapear a mesma variável com múltiplas estéticas desde que elas sejam aplicáveis ao tipo de variável usada (contínua ou categórica).

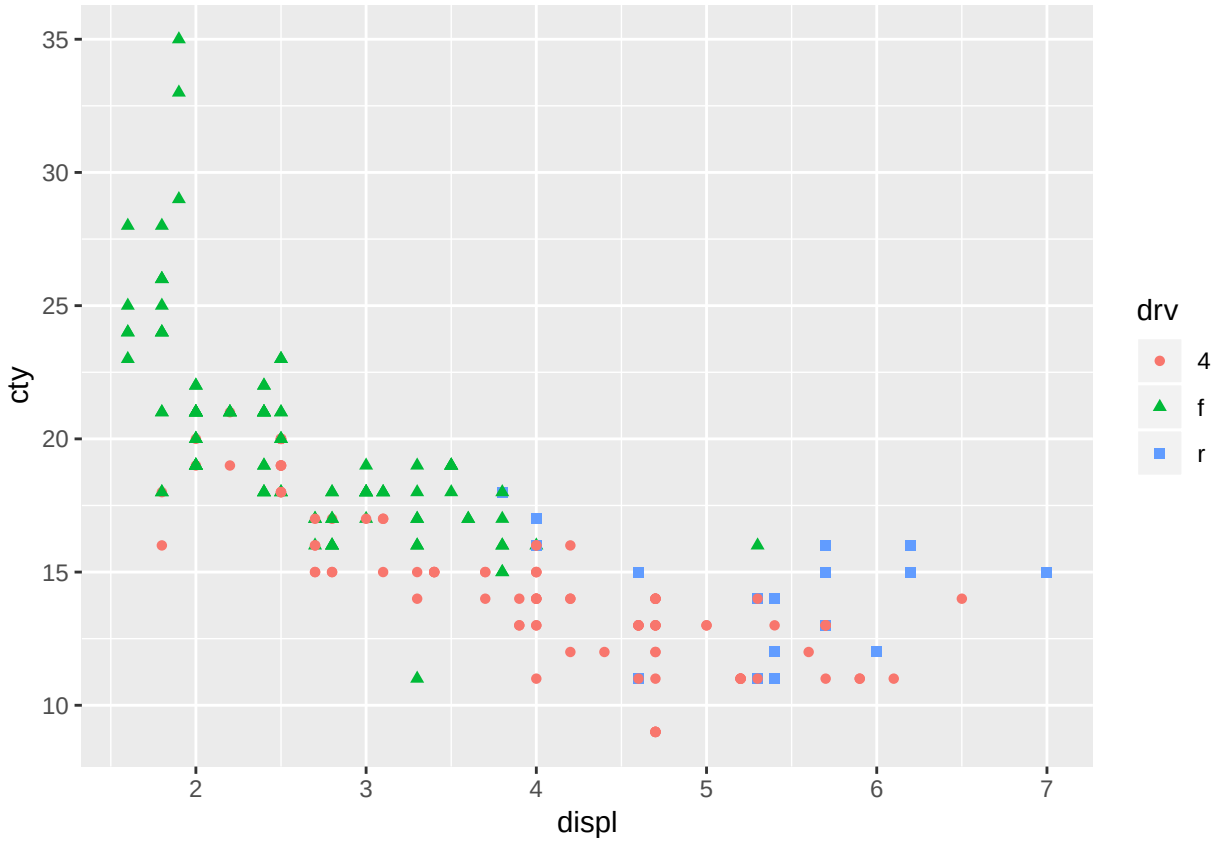
Exemplo para variável contínua:

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



Exemplo para variável categórica:

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = cty, shape = drv, color = drv))
```

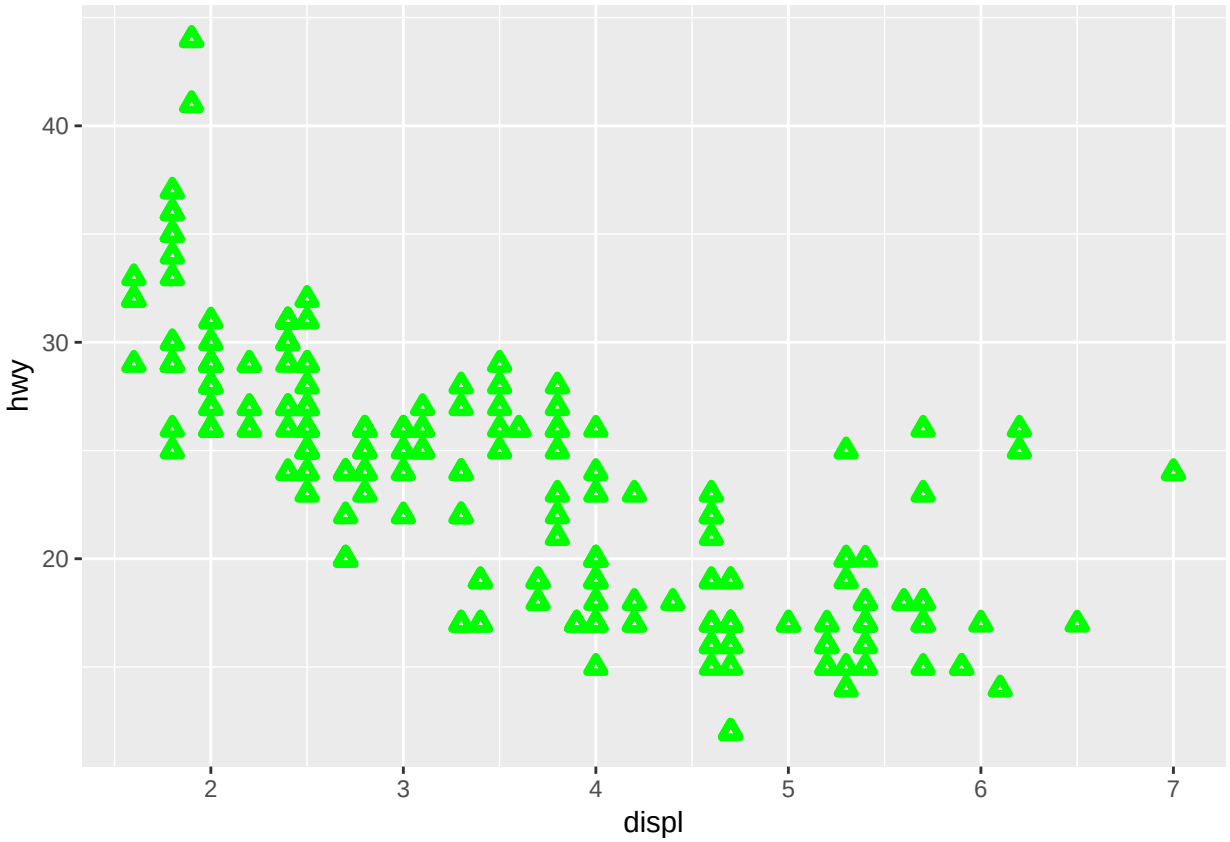


Exercício 5

P: O que faz a estética `stroke`? Com quais formas ele funciona? (Dica: use `?Geom_point`)

R: A estética `stroke` é utilizada para modificar a espessura da borda em formas que possuem tais bordas (21 a 25). Por exemplo:

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), shape = 24,
             fill = 'white', size = 1, stroke = 2, color = 'green')
```

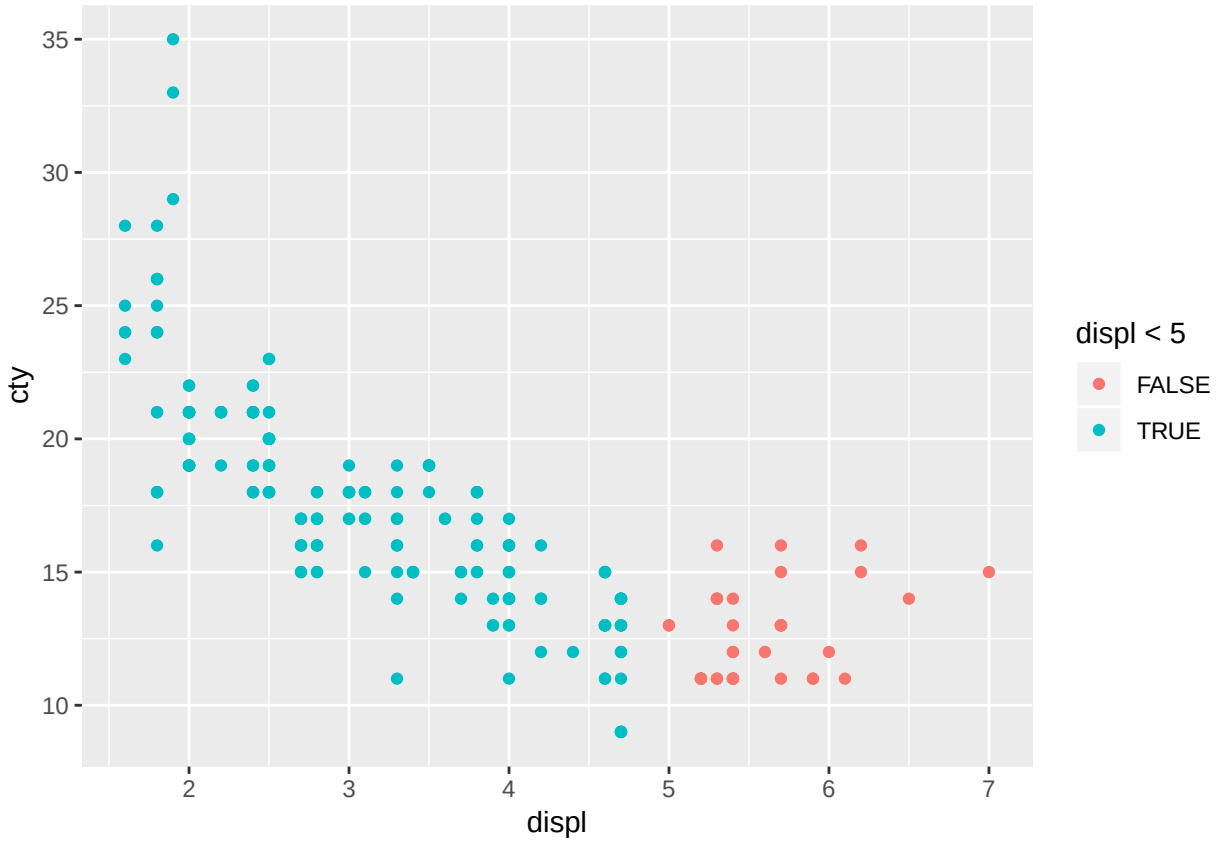


Exercício 6

P: O que acontece se você mapear uma estética para algo diferente de um nome de variável, como `aes (color = displ <5)`? (Você também precisará especificar x e y).

R: O `ggplot2` avalia o resultado da expressção e cria uma variável booleana temporária para mostrar, com cores diferentes, quais elementos satisfazem e quais não satisfazem a expressão.

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


35
30
25
20
15
10
cty
2
3
4
5
6
7
displ
displ < 5
FALSE
TRUE

Seção 3.5.1

Exercício 3

P: Quais gráficos o código a seguir faz? O que . faz?

Código 1:

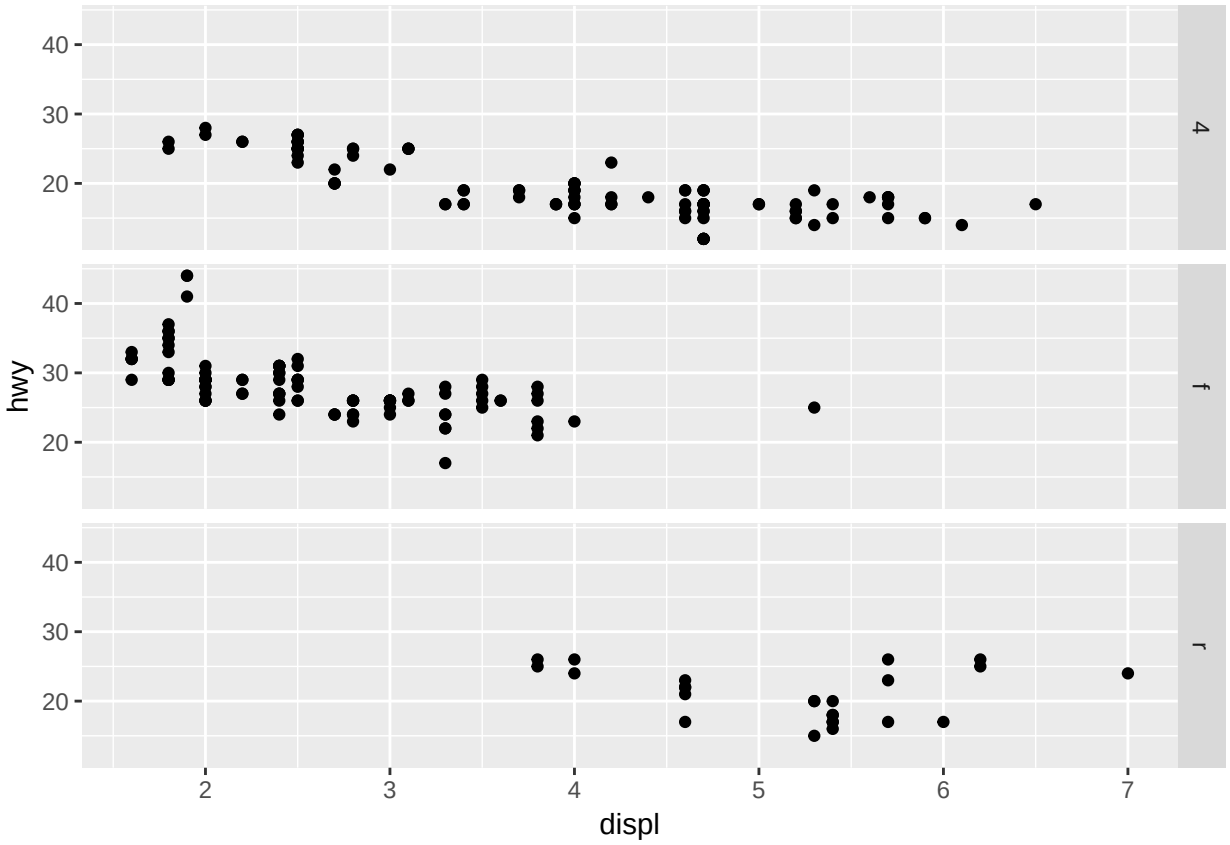
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ .)
```

Código 2:

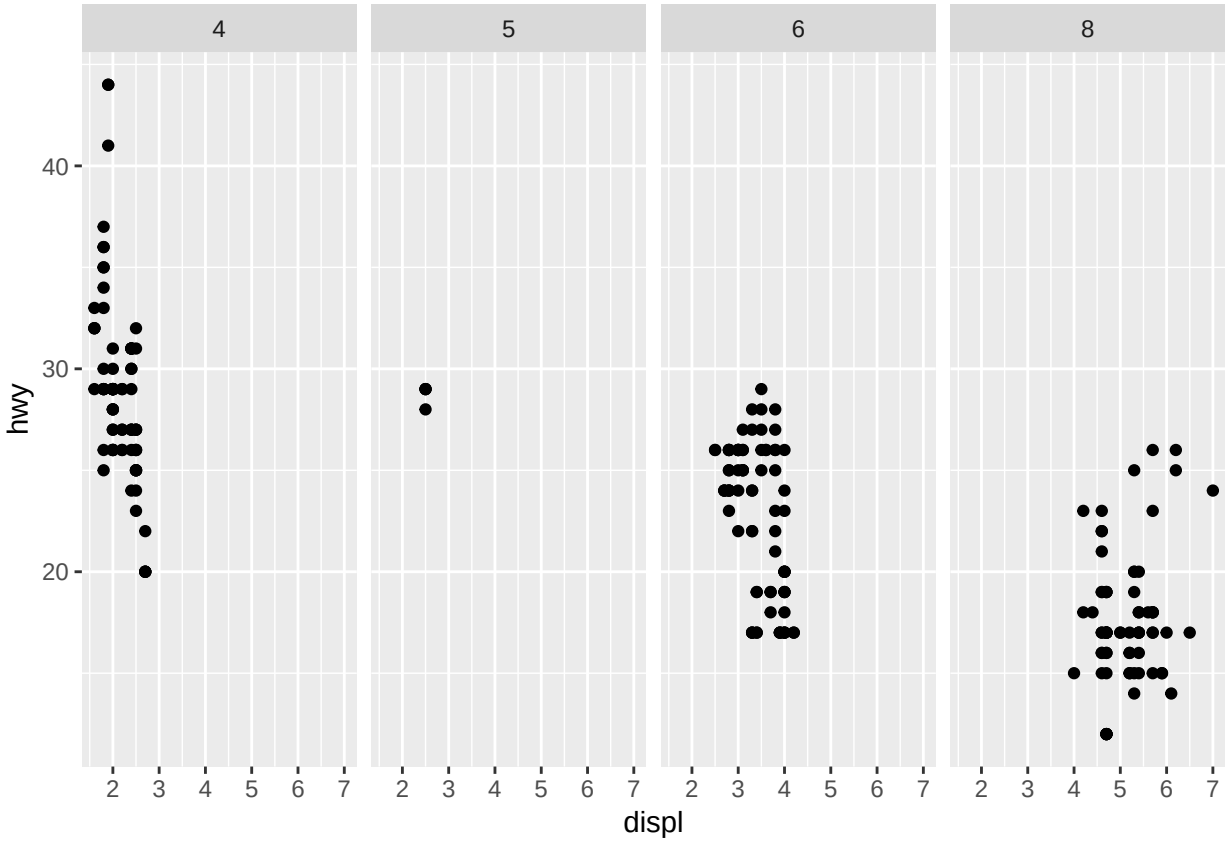
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(. ~ cyl)
```

R: Utilizando `facet_grid ()`, as linhas e colunas são plotadas pela variável do lado esquerdo e do lado direito de `~`, respectivamente. Forma um gride definido por variáveis de linha e coluna. É mais útil quando você tem duas variáveis discretas e todas as combinações das variáveis existem nos dados. O . significa que não haverá faceta na dimensão.

Código 1: Cria um grid com a variável `drv` no eixo x e sem variável no eixo y.



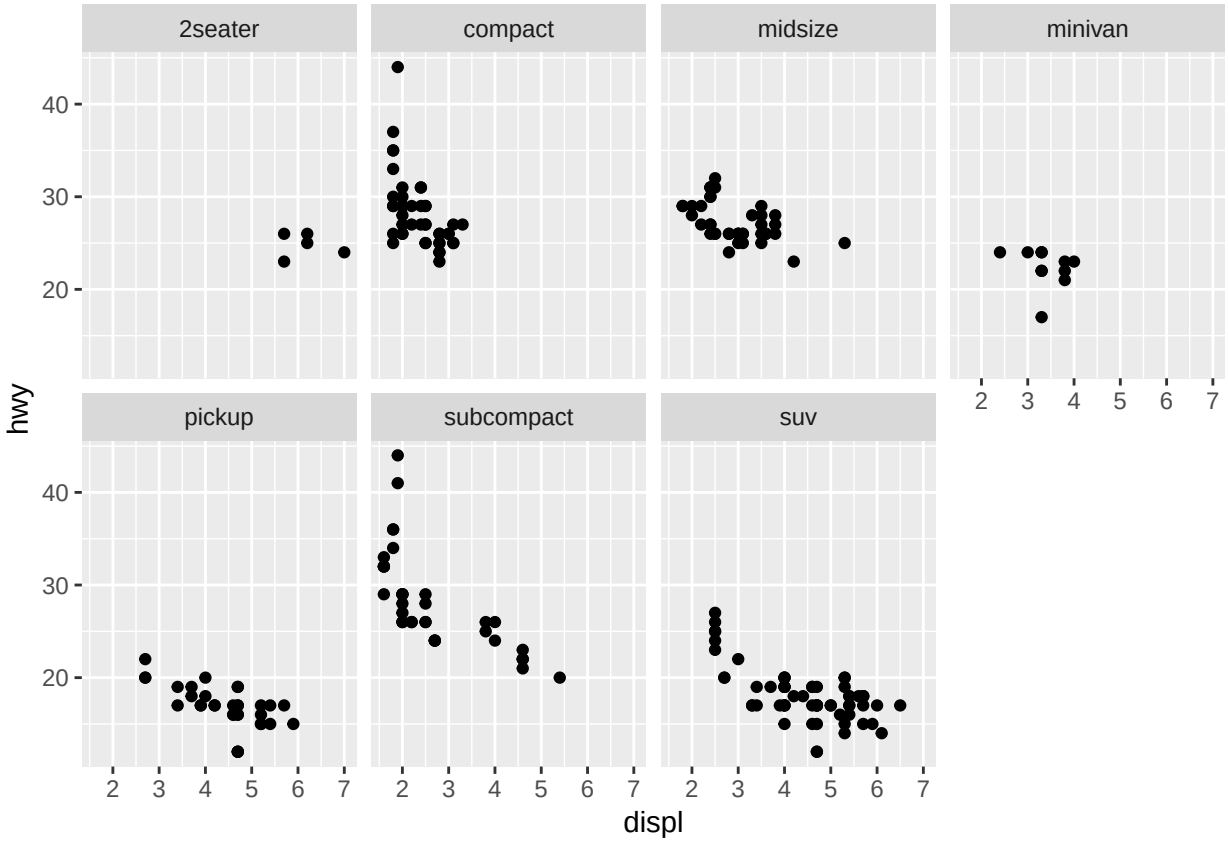
Código 2: Cria um grid com a variável `cyl` no eixo y e sem variável no eixo x.



Exercício 4

P: Pegue o primeiro gráfico facetado nesta seção:

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```



Quais são as vantagens de usar o **faceting** em vez da estética de cores? Quais são as desvantagens? Como a comparação poderia mudar se você tivesse um conjunto de dados maior?

R: As vantagens de se utilizar facetas em vez da estética de cores residem na visualização dos dados quando há muitas categorias. Utilizando a estética de cores a visualização dos dados pode ficar confusa quando há muitas cores envolvidas. Já as desvantagens são em razão dos pontos estarem em gráficos separados, o que pode dificultar comparações diretas. Para um conjunto de dados grandes pode haver sobreposição de pontos. Além disso, se houverem muitas categorias, a diferença entre as cores que as representam será menor, tornando visualmente difícil distinguir as categorias.

Exercício 5

P: Leia `?facet_wrap`. O que `nrow` faz? O que `ncol` faz? Quais outras opções controlam o layout dos painéis individuais? Por que `facet_grid ()` não tem argumentos `nrow` e `ncol`?

R: Os argumentos `nrow` e `ncol` definem o número de linhas e colunas dos painéis. Em `facet_grid()` não há os argumentos `nrow` e `ncol`, pois o número de valores exclusivos das variáveis especificadas na função determina o número de linhas e colunas.

Exercício 6

P: Ao usar `facet_grid()`, você normalmente deve colocar a variável com mais níveis exclusivos nas colunas. Por quê?

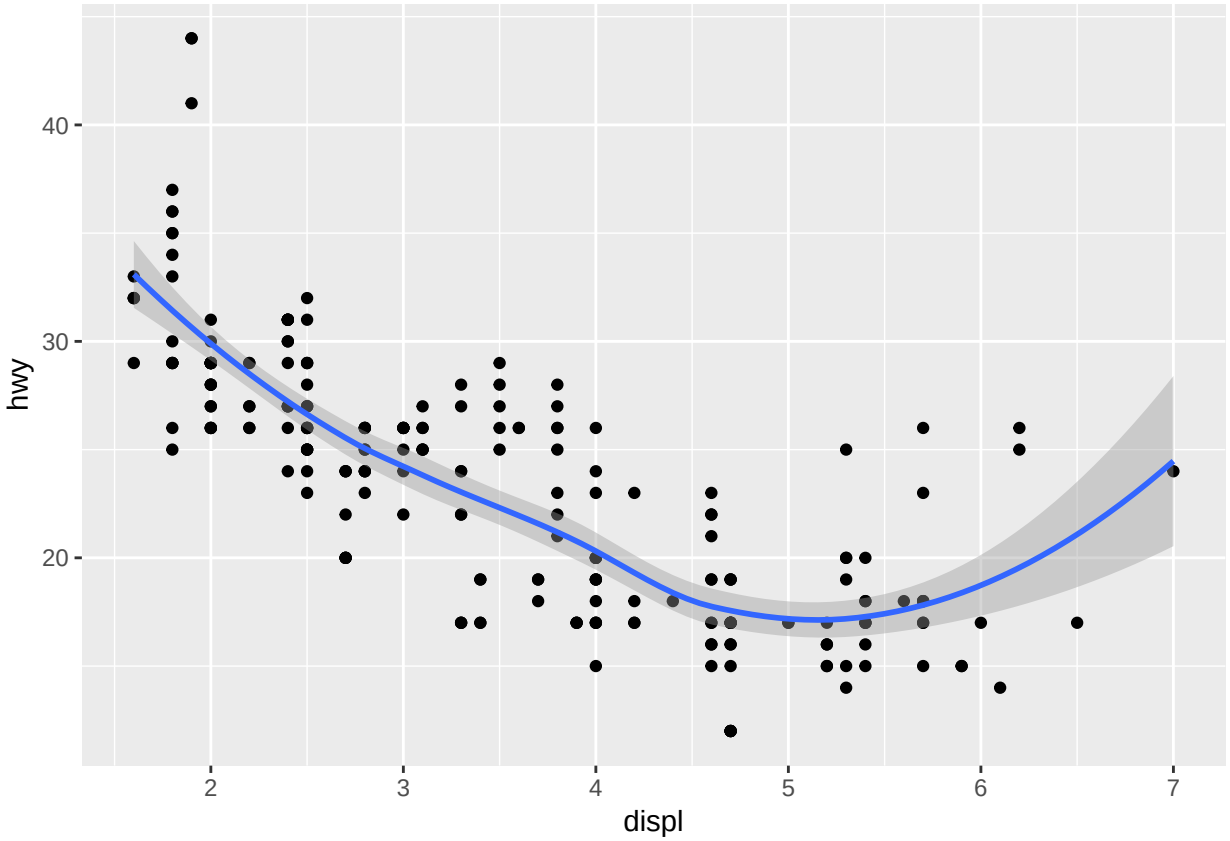
R: Porque dessa forma se terá espaço para colunas, desde que o gráfico esteja disposto horizontalmente.

Seção 3.6.1

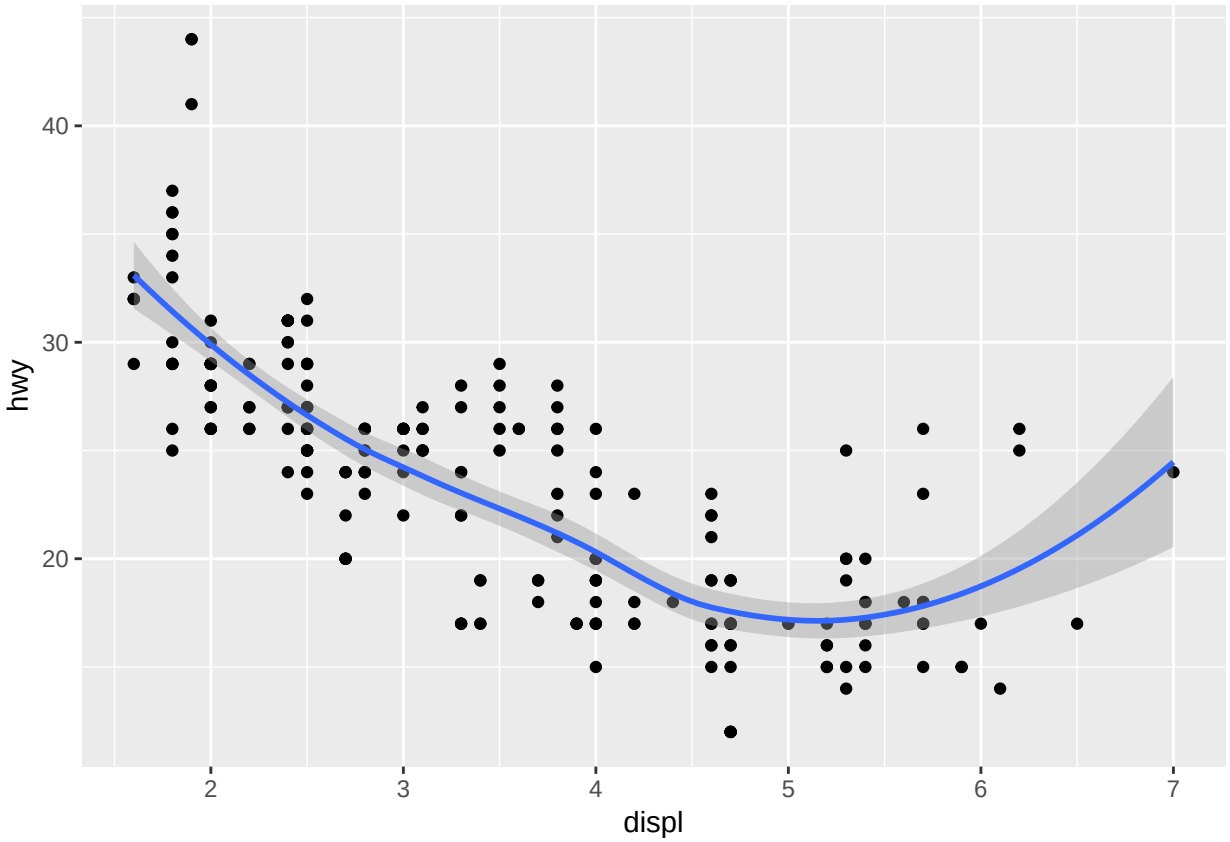
Exercício 5

P: Esses dois gráficos parecem diferentes? Por quê?

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




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



R: Os gráficos não são diferentes. Tanto `geom_point()` como `geom_smooth()` usaram os mesmos dados e mapeamentos em ambos os gráficos. Na primeira parte do código (primeiro gráfico), `geom_point()` e `geom_smooth()` irão herdar do objeto `ggplot()` os dados e o mapeamento, não sendo necessário que sejam especificados novamente. Já na segunda parte do código (segundo gráfico) os dados e mapeamentos são especificados individualmente para `geom_point()` e `geom_smooth()`.

Exercício 6

P: Recrie o código R necessário para gerar os gráficos a seguir.

R: Gráfico 1:

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

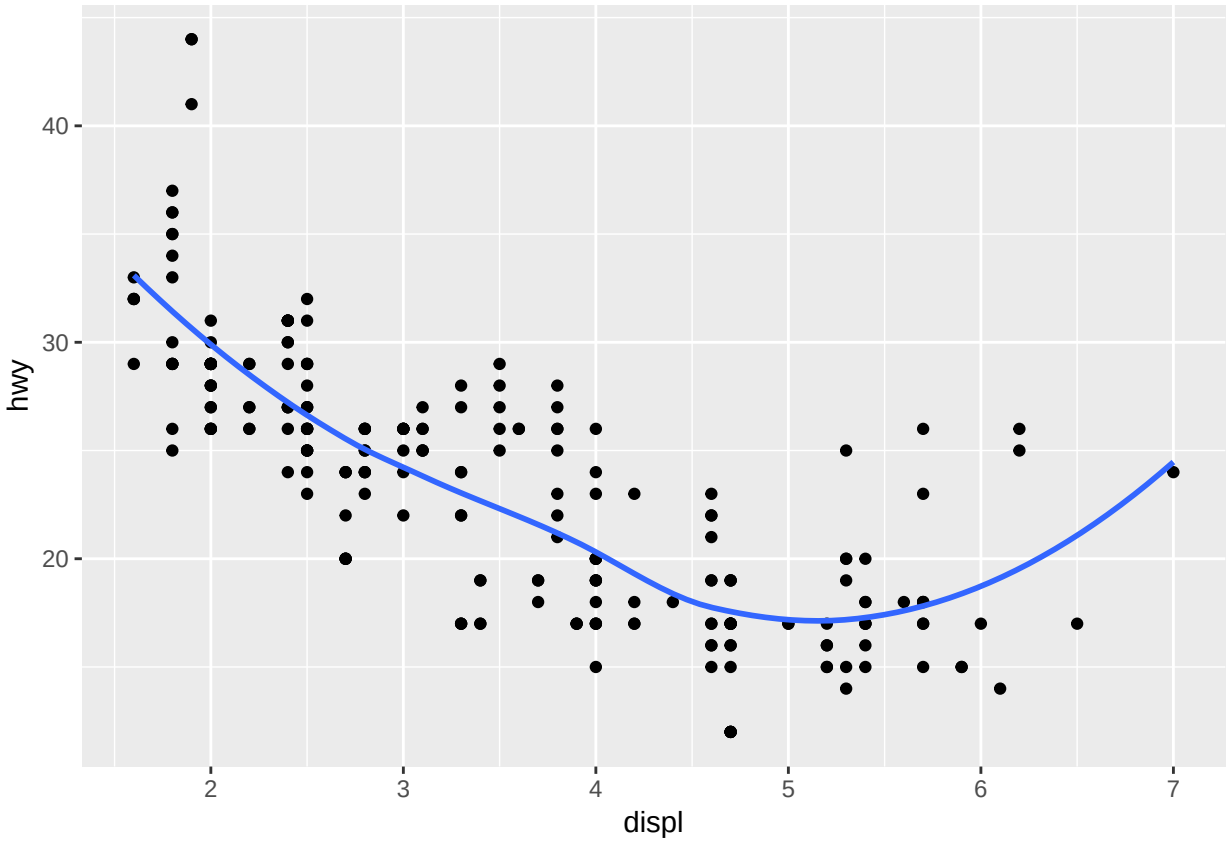


Gráfico 2:

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

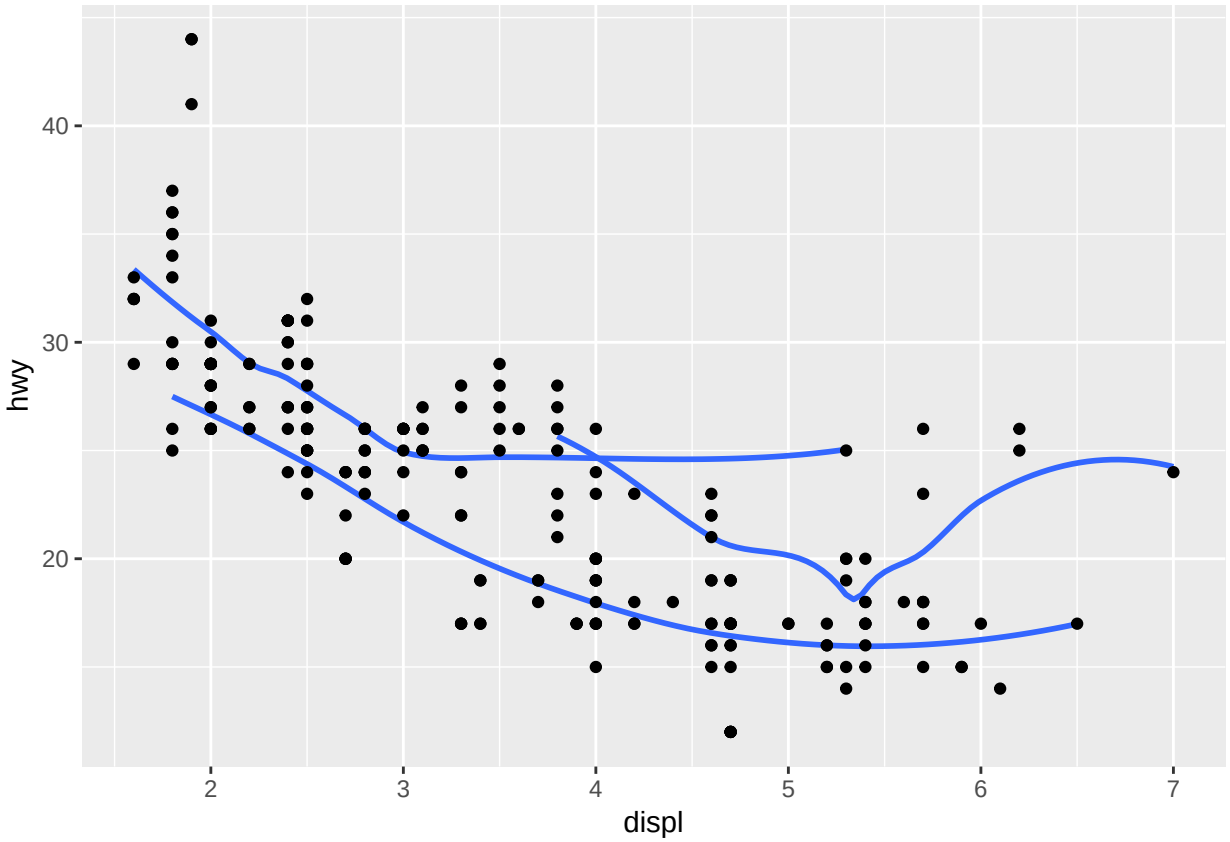


Gráfico 3:

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

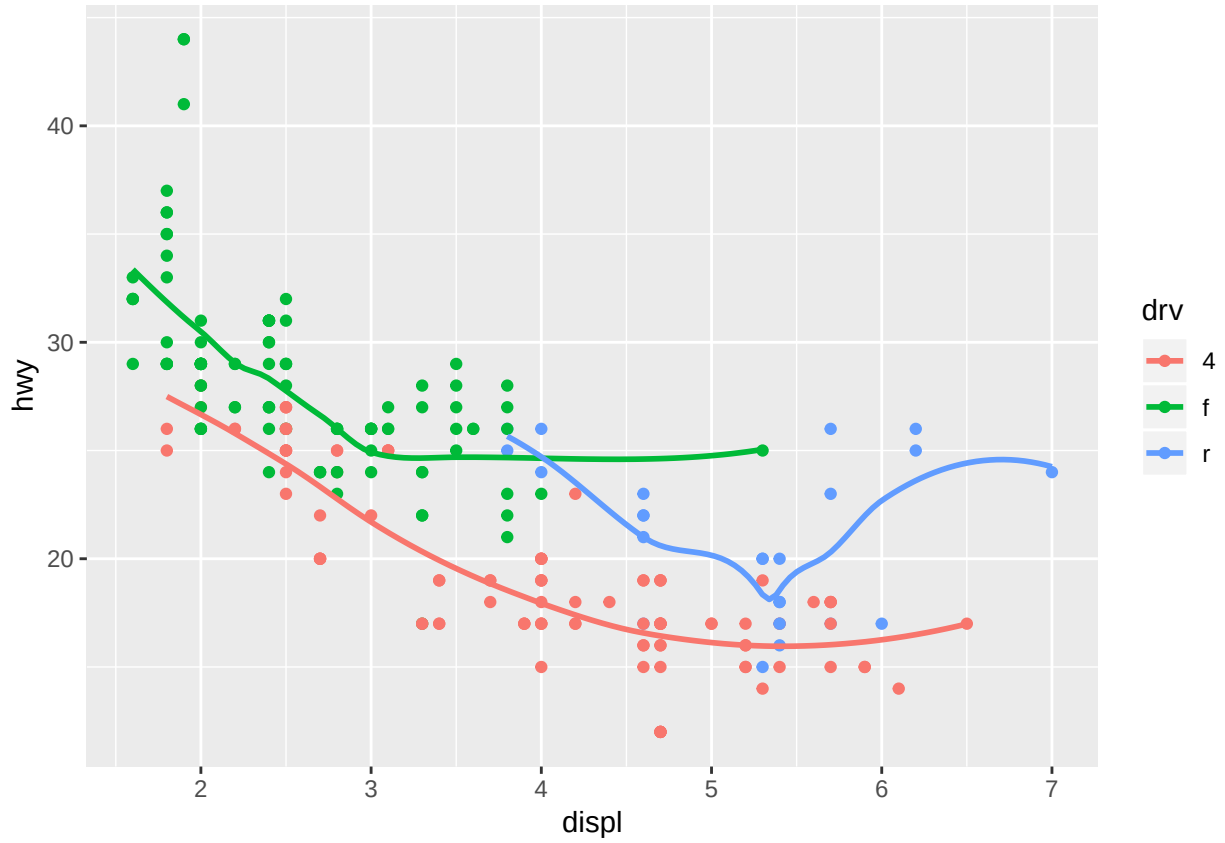


Gráfico 4:

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

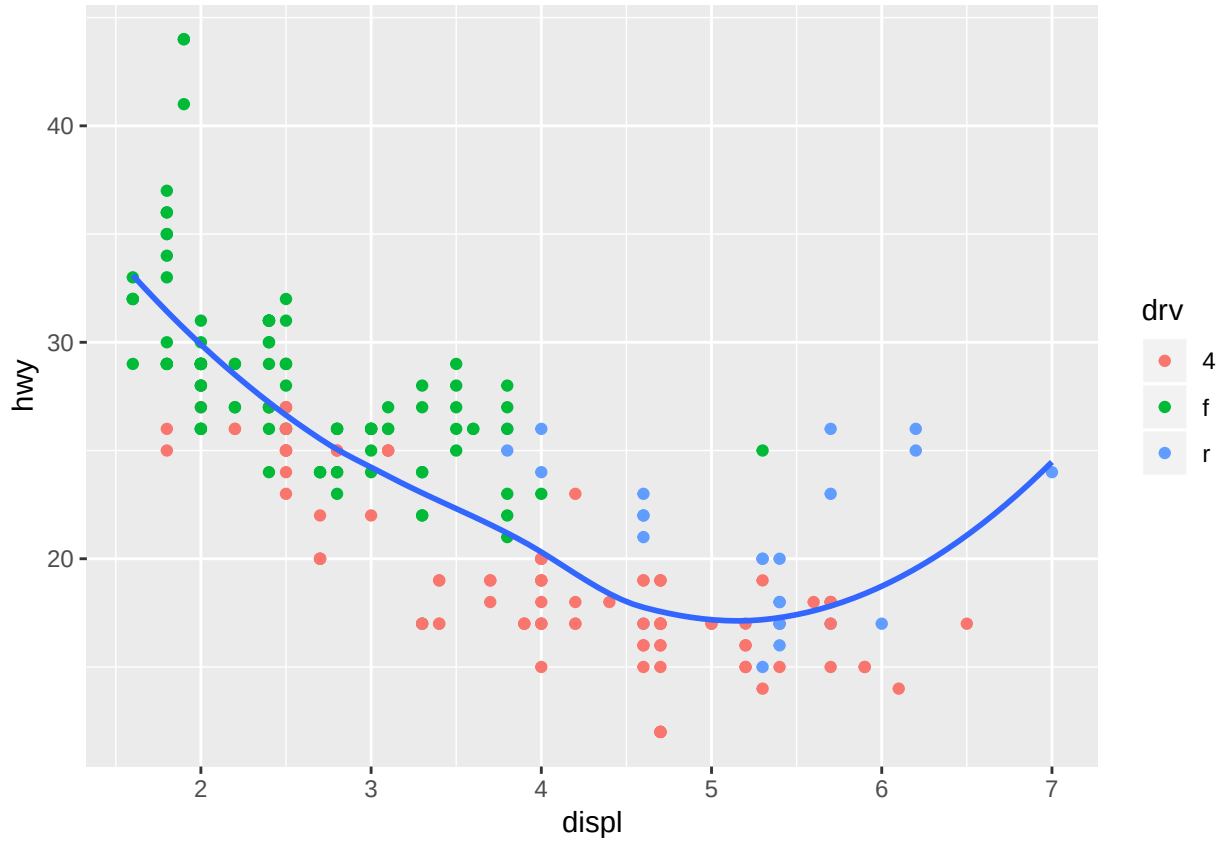


Gráfico 5:

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

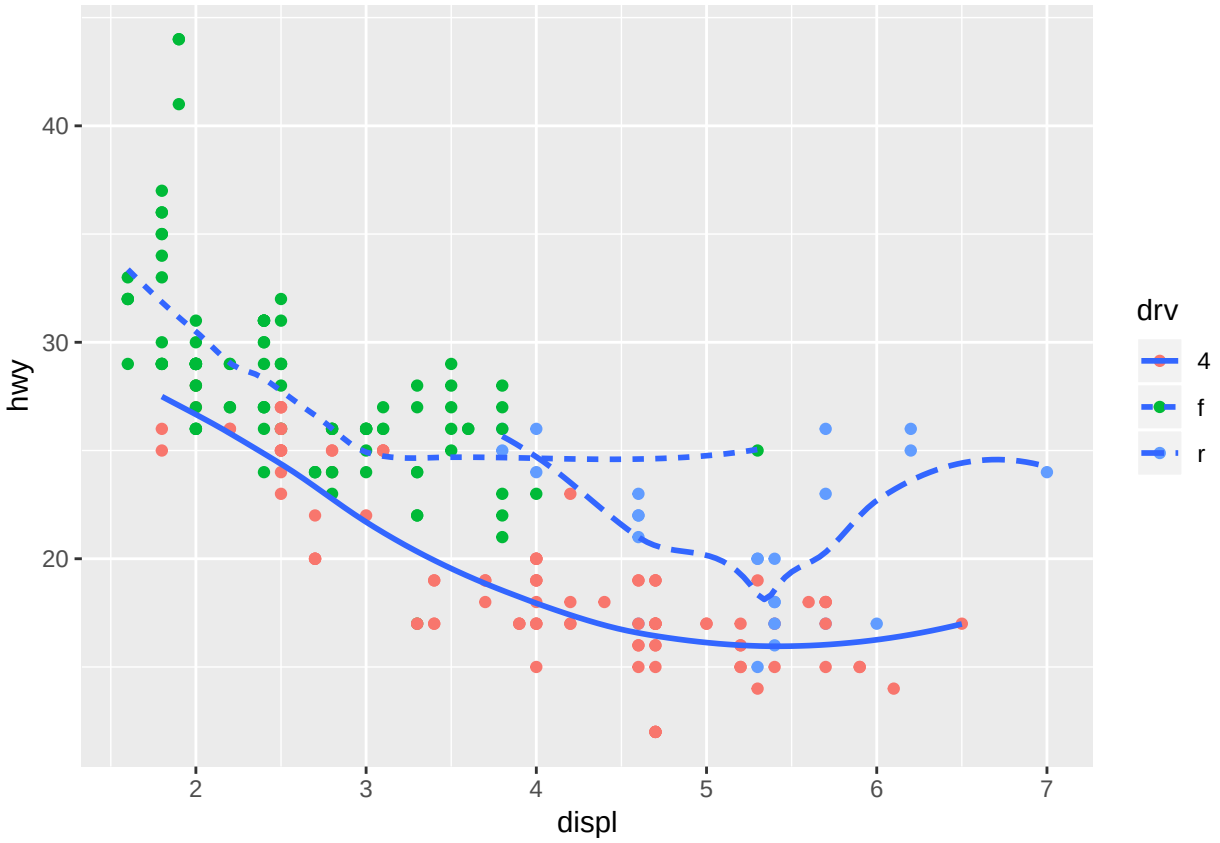
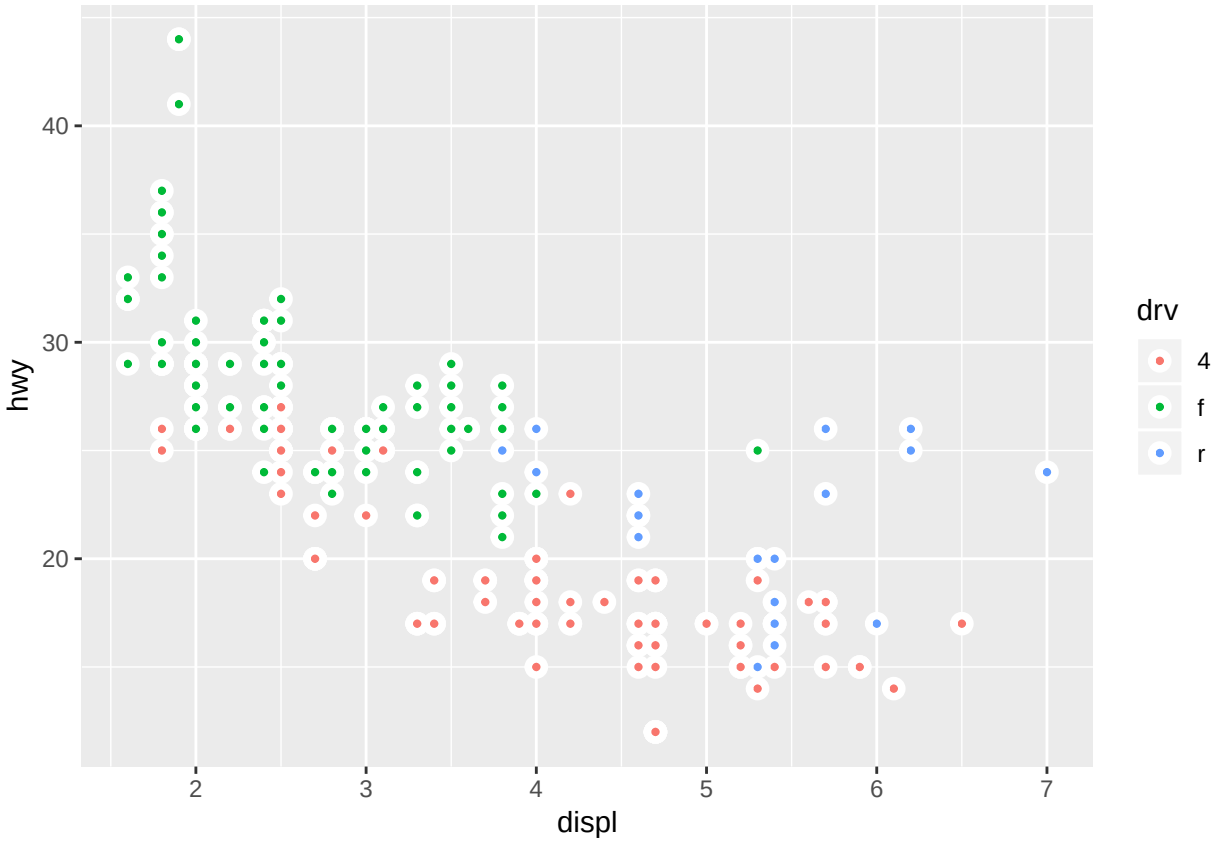


Gráfico 6:

```
ggplot(data = mpg, mapping = aes(y = hwy, x = displ)) +
  geom_point(mapping = aes(fill = drv), color = 'white', stroke = 2, shape = 21)
```


hwy
40
30
20
2
3
4
5
6
7
displ
drv
4
f
r

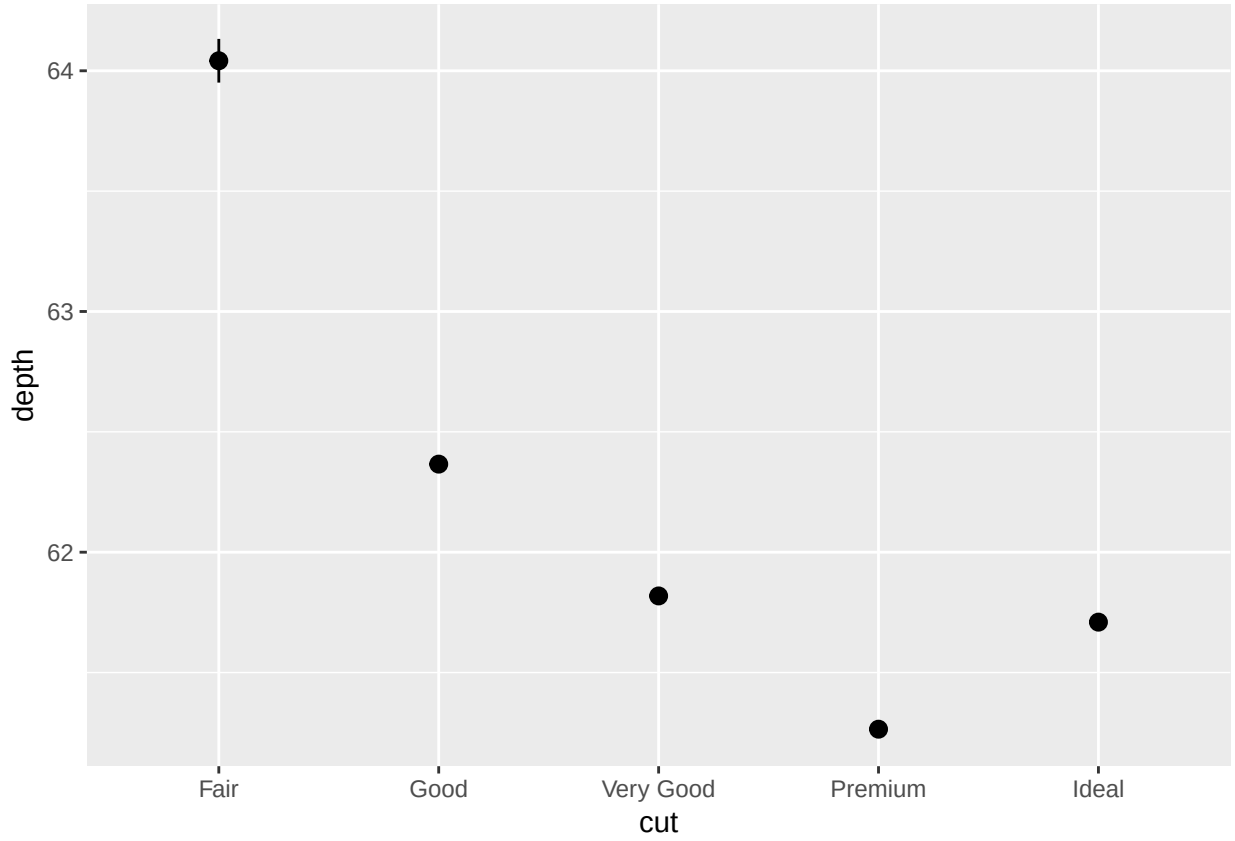
Seção 3.7.1

Exercício 1

P: Qual é o padrão associado a `stat_summary()`? Como você poderia plotar o gráfico anterior usando essa função geom em vez da função stat?

R: O padrão para `stat_summary()` é `geom_pointrange()` e a estatística padrão para `geom_pointrage()` é `stat_identity()`, significando que as medianas, mínimos e máximos devem ser computados primeiro. Porém é possível adicionar o argumento `stat = "summary"` para se usar `stat_summary()` em vez de `stat_identity()`. Isso pode ser feito com o seguinte código:

```
ggplot(data = diamonds) +
  geom_pointrange(
    mapping = aes(x = cut, y = depth),
    stat = "summary"
  )
```

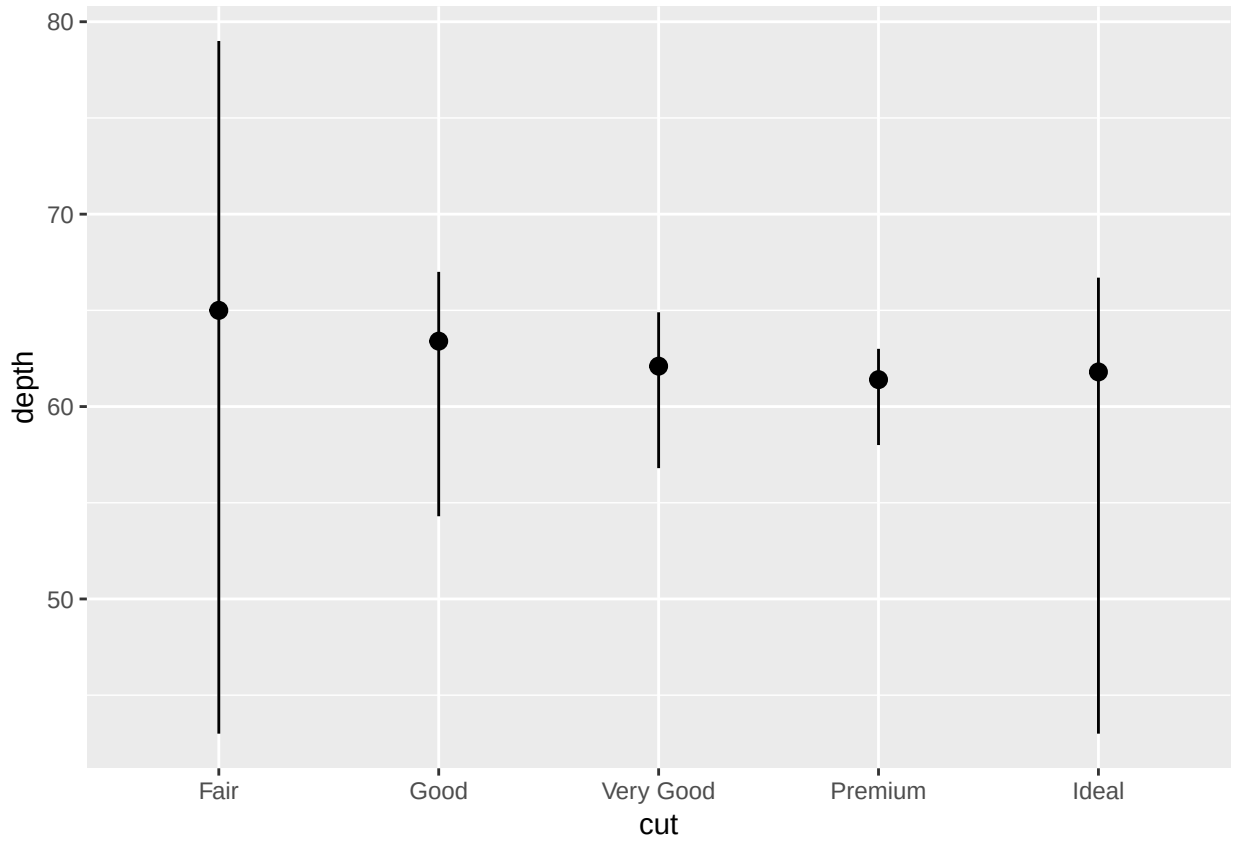



```
    ## > No summary function supplied, defaulting to `mean_se()
```

Entretanto é retornada uma mensagem informando que `stat_summary()` utiliza a média e o desvio padrão para calcular os pontos médios e os pontos finais das linhas no gráfico. Já no gráfico original são utilizados os pontos mínimos e máximos para plotar os pontos finais das linhas e portanto para se recriar o gráfico original é preciso que se repita isso. Dessa forma, o código fica assim:

```
ggplot(data = diamonds) +
  geom_pointrange(
    mapping = aes(x = cut, y = depth),
    stat = "summary",
```

```
    fun.ymin = min,
    fun.ymax = max,
    fun.y = median)
```

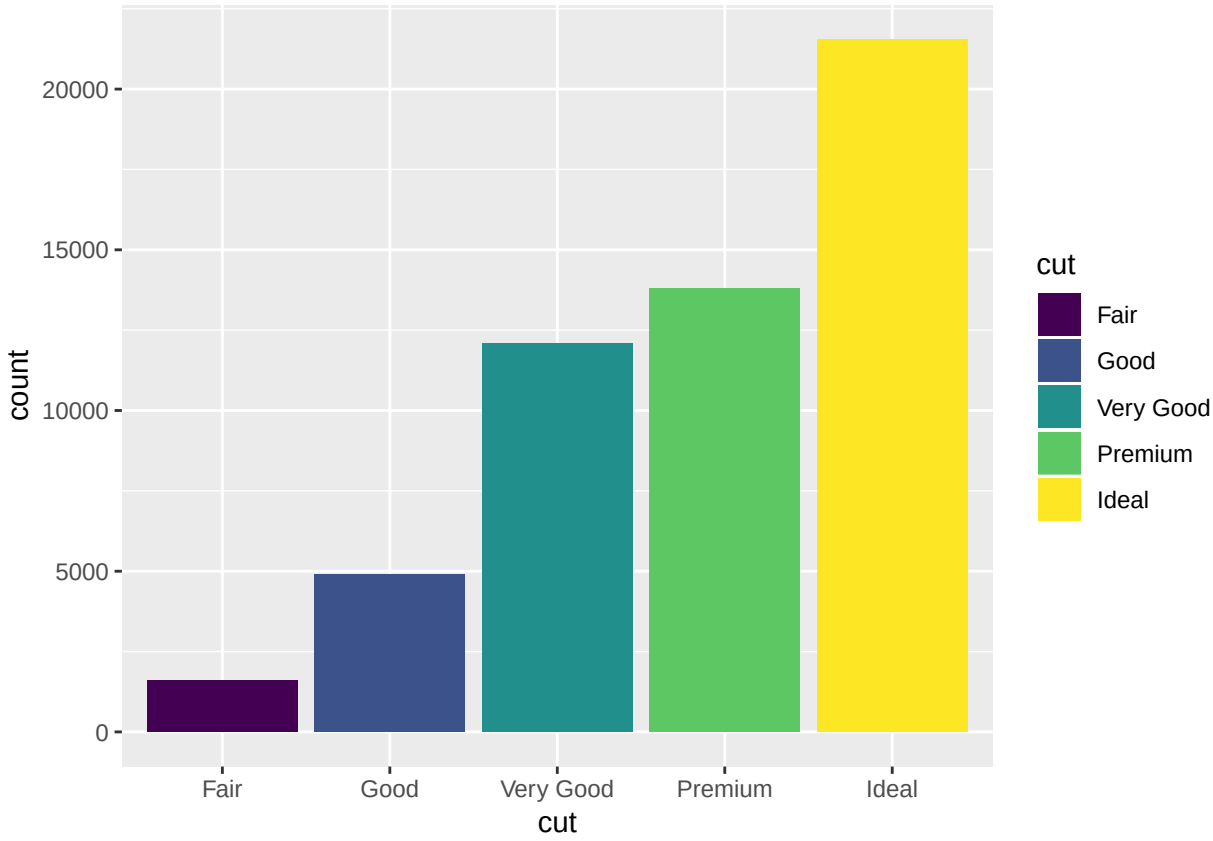


Exercício 2

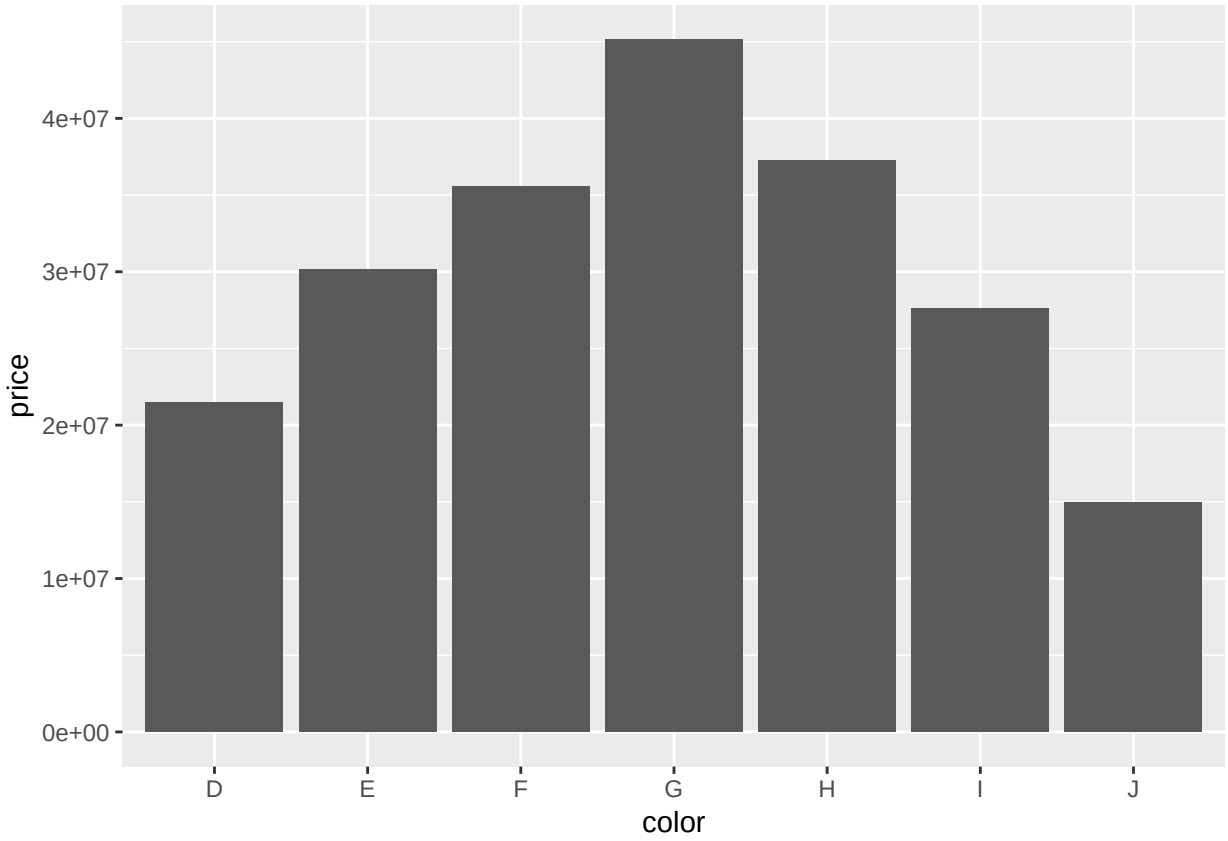
P: O que o `geom_col()` faz? Como isso é diferente de `geom_bar()`?

R: Ambos criam gráficos de barras. Porém `geom_bar()` torna a altura da barra proporcional ao número de casos em cada grupo. Já se o que se deseja for que as alturas das barras representem os valores dos dados, deve se usar `geom_col()`. O `geom_bar()` usa `stat_count()` por padrão, em que se conta o número de casos em cada posição x e `geom_col()` usa `stat_identity()`, em que se deixa os dados como estão. Por exemplo:

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




```
ggplot(data = diamonds) +
  geom_col(mapping = aes(x = color, y= price))
```

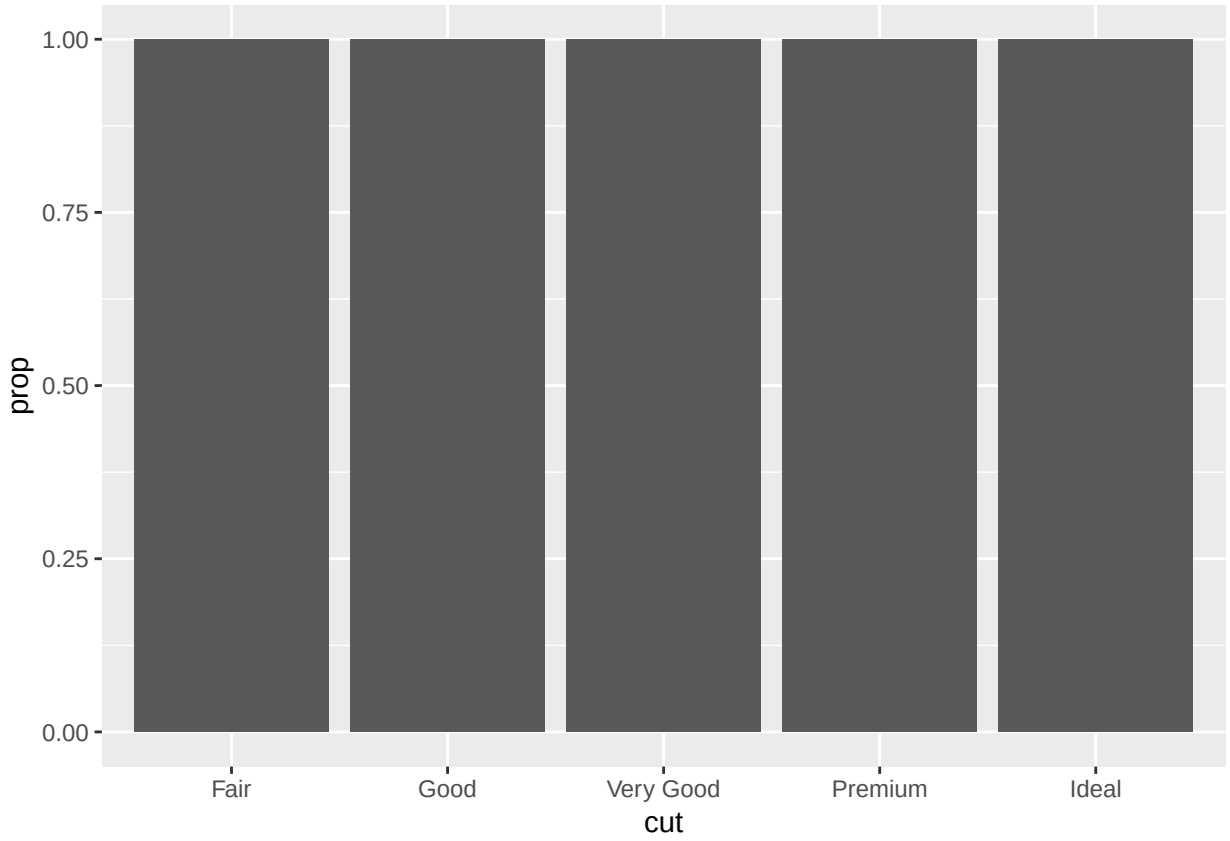


Exercício 5

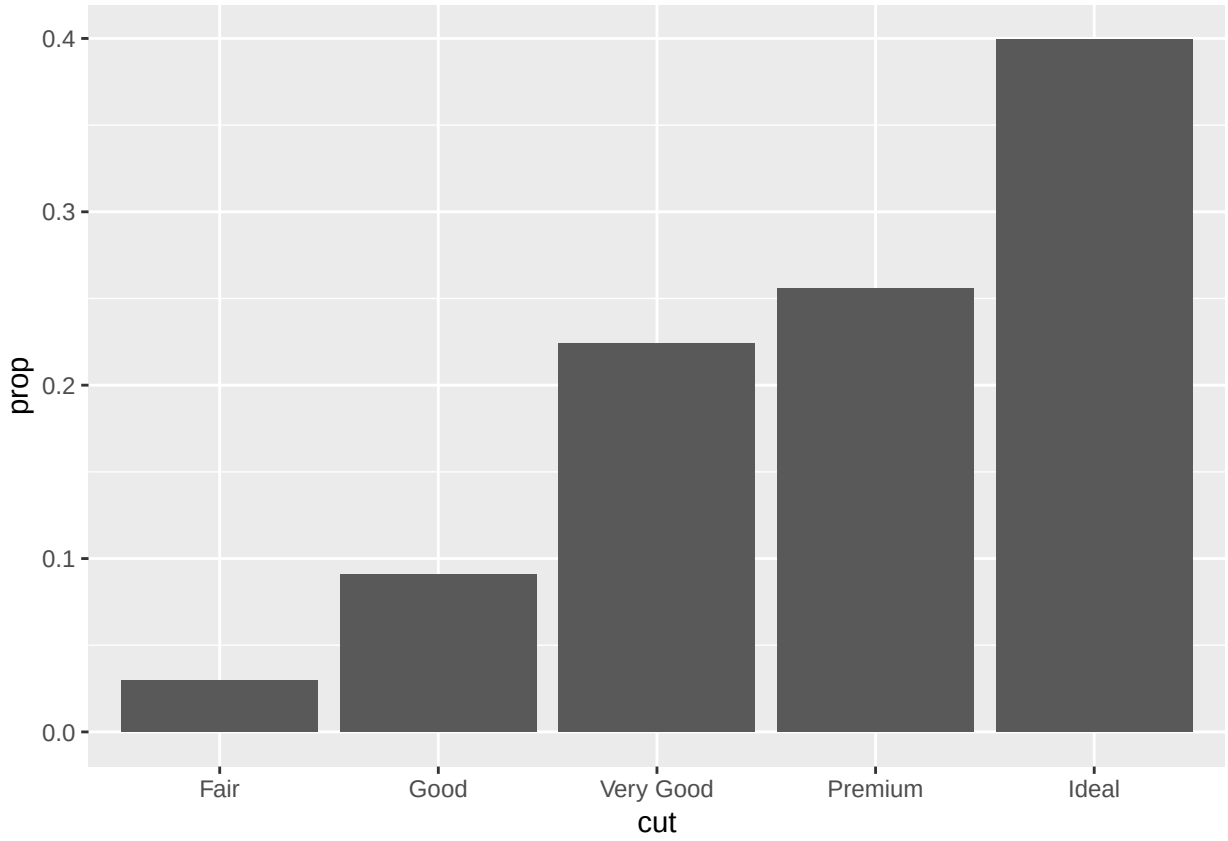
P: Em nosso gráfico de barras de proporção, precisamos definir o `group = 1`. Por quê? Em outras palavras, qual é o problema desses dois gráficos?

R: Porque se o `group = 1` não for incluído, todas as barras terão mesma altura na plotagem (altura 1). Esse problema ocorre porque, por padrão, `geom_bar()` conta o número de ocorrências em cada nível da variável e se quisermos exibir as proporções em vez de contagens, `geom_bar()` tratará os grupos da variável separadamente. Como, por exemplo, todos os diamantes em *"Fair"* são *"Fair"* e todos os diamantes em *"Good"* são *"Good"*, as proporções sempre serão iguais a 1 (ou 100%) para cada grupo.

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = ..prop..))
```




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

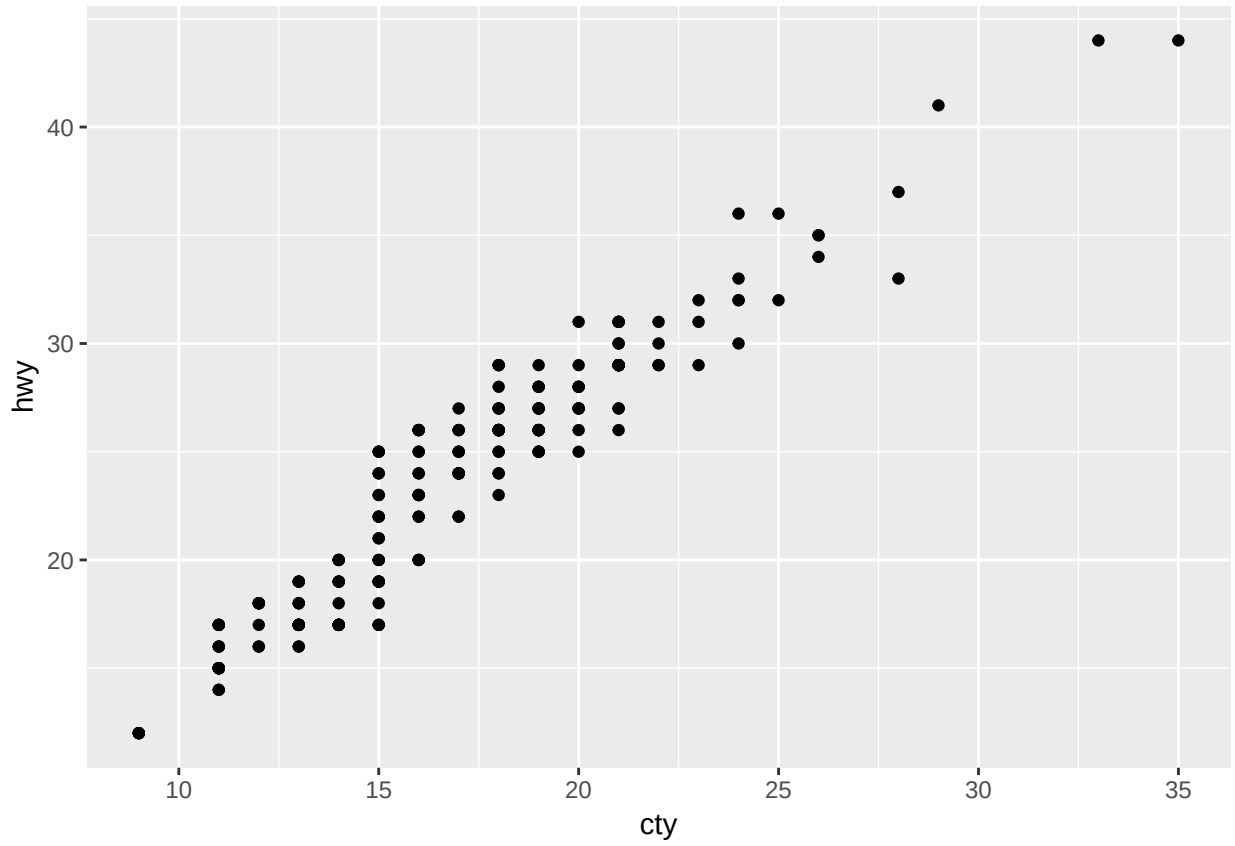

0.4
0.3
0.2
0.1
0.0
prop
Fair
Good
Very Good
Premium
Ideal
cut

Seção 3.8.1

Exercício 1

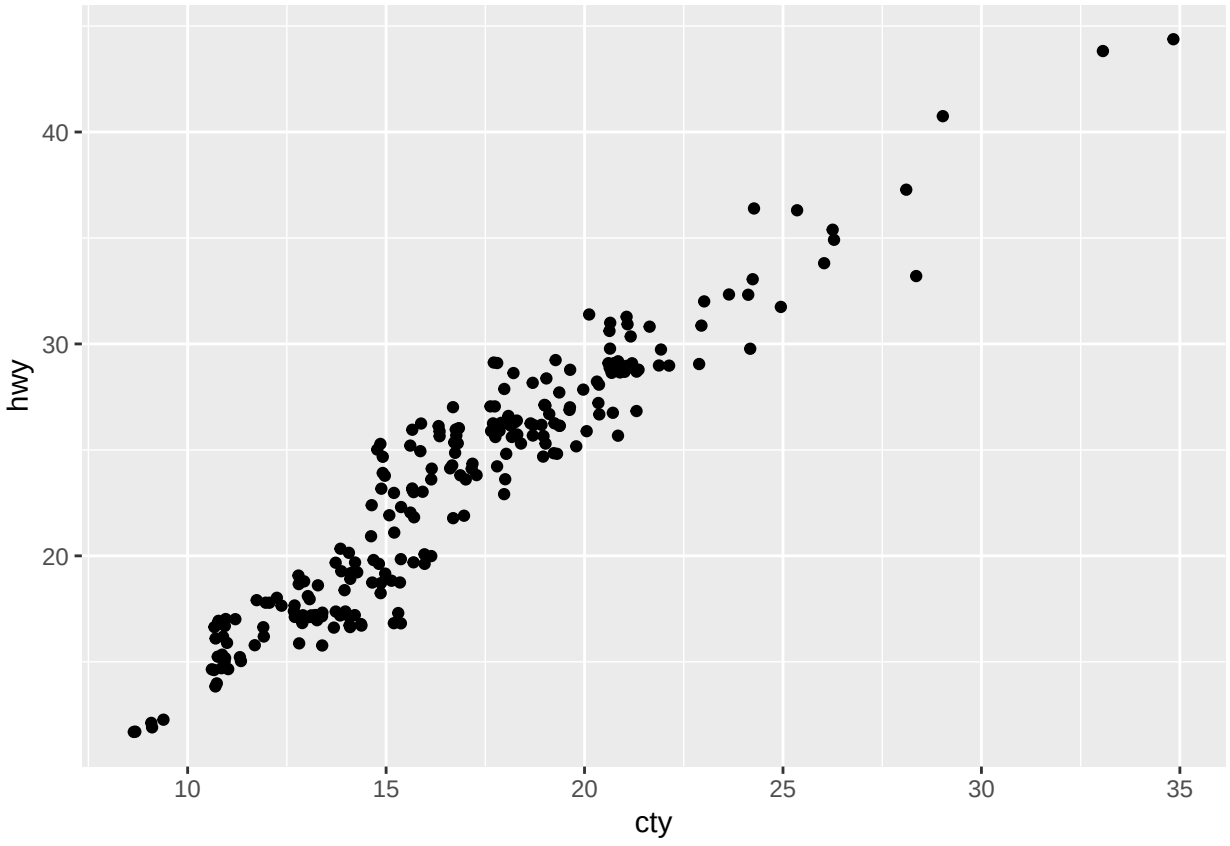
P: Qual é o problema com esse gráfico? Como você poderia melhorar isso?

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



R: O problema é que há sobreposição gráfica das observações. Há várias observações para cada combinação de valores `cty` e `hwy`, o que faz com que alguns pontos se sobreponham e não seja possível ver onde está o maior volume de dados. Uma forma de contornar este problema é usando o ajuste de posiçãoo *jitter* para diminuir a sobreposição, `geom_jitter()` adiciona uma pequena quantidade de variação aleatória à localização de cada ponto.

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



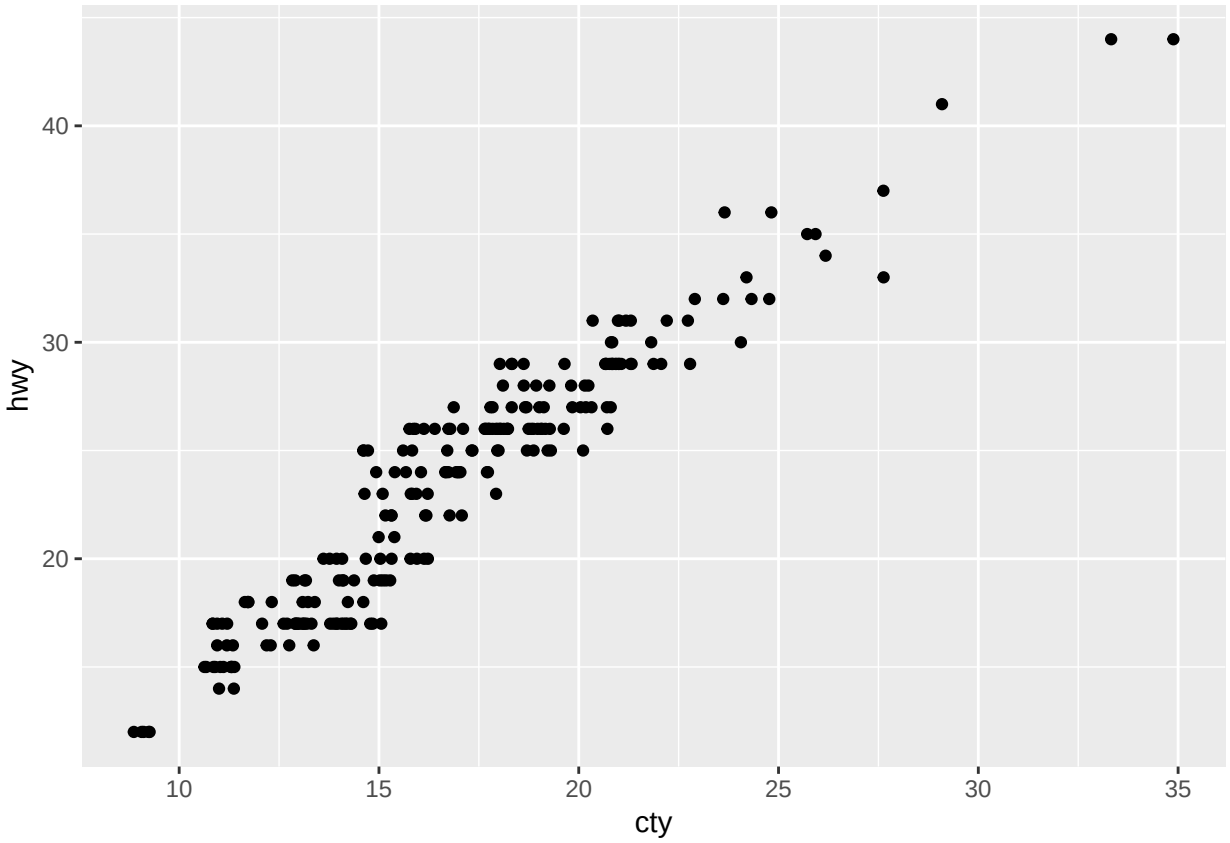
Exercício 2

P: Quais parâmetros para `geom_jitter()` controlam a quantidade de "variação" inserida?

R: Há dois parâmetros para isso: *width*, que é a quantidade de variação horizontal e *height*, que é a quantidade de variação vertical. Por *default*, `geom_jitter()` adiciona variação em ambas as direções.

Exemplo sem variação vertical:

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_jitter(height = 0)
```



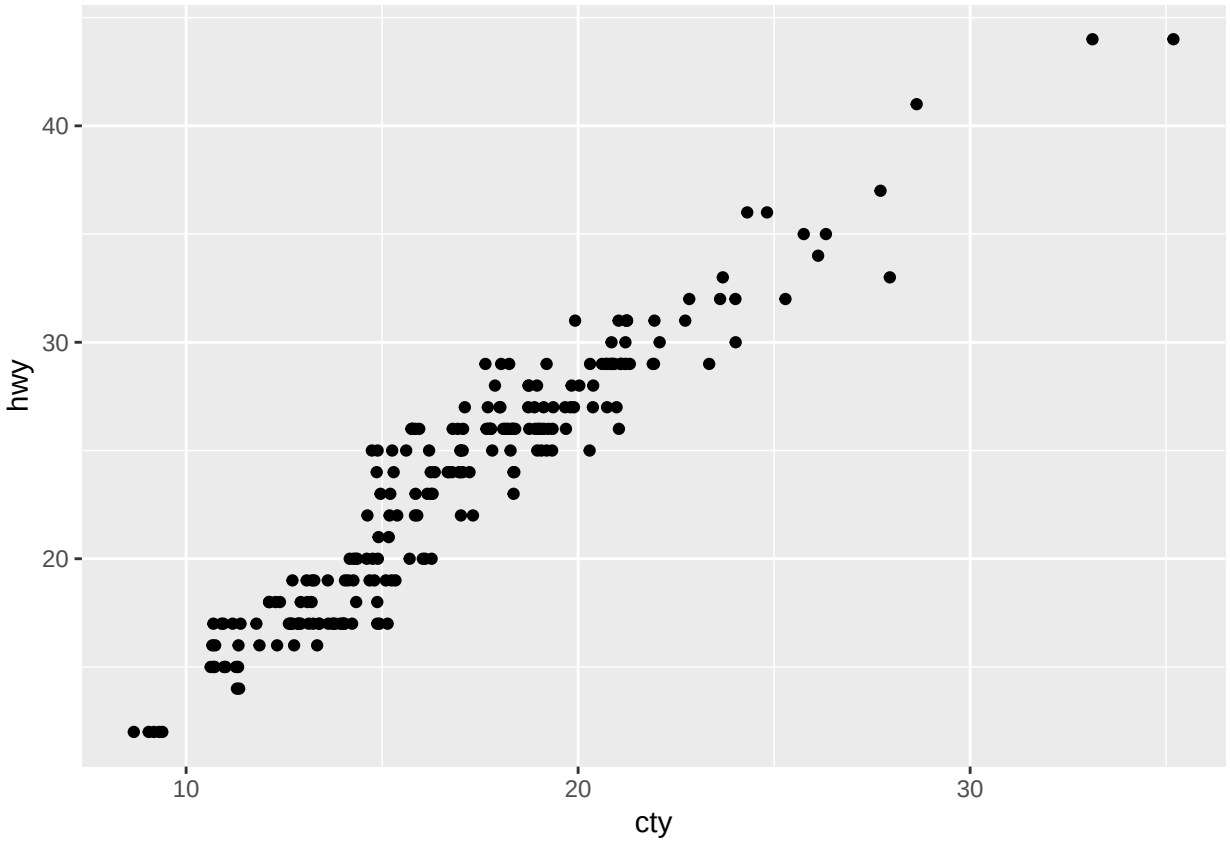
Exercício 3

P: Compare `geom_jitter()` com `geom_count()`.

R: Ambos podem representar melhor os dados quando há muitos pontos sobrepostos. O `geom_jitter()` adiciona uma pequena variação aleatória aos pontos no gráfico. Isso reduz a sobreposição, pois é improvável que dois pontos com a mesma localização tenham a mesma variação aleatória. Já `geom_count()` redimensiona os pontos em relação ao número de observações, as combinações de valores (x, y) com mais observações serão maiores do que aquelas com menos observações.

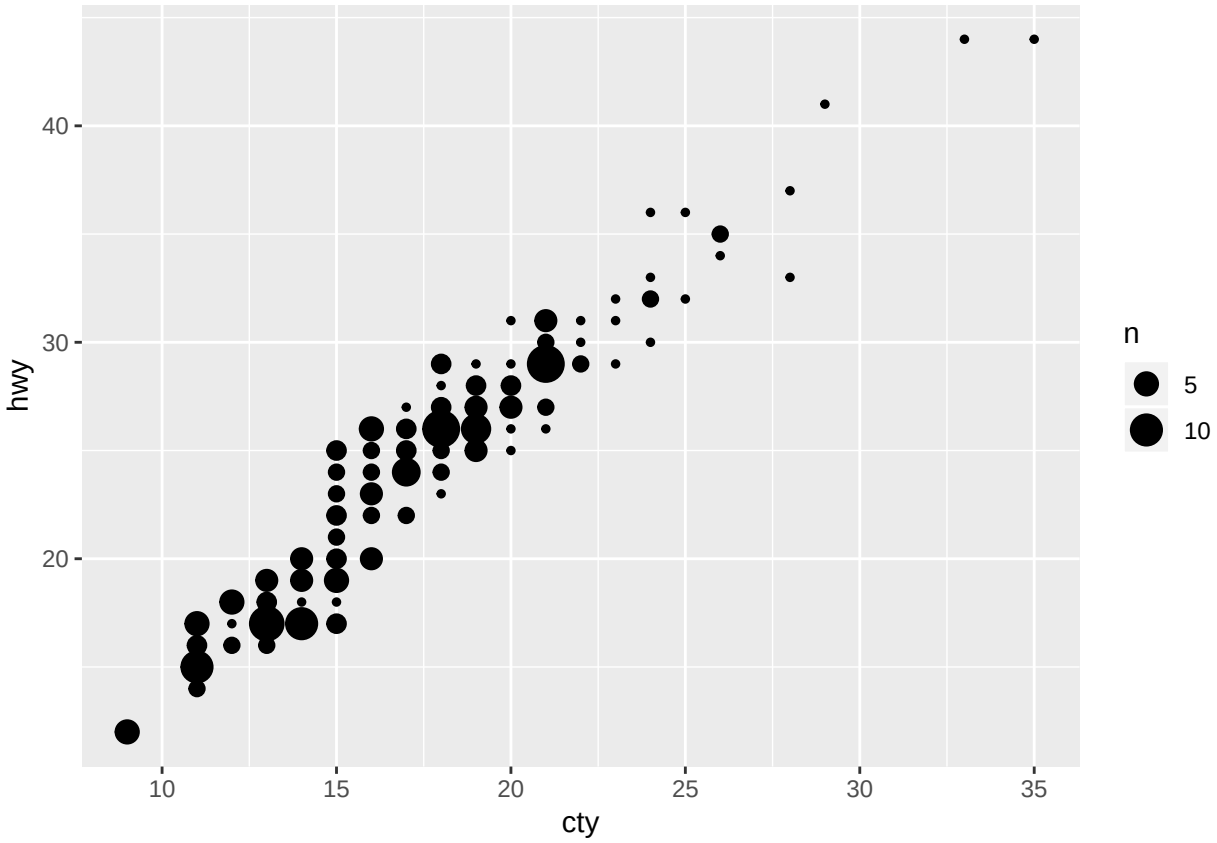
Exemplo com `geom_jitter()`:

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_jitter(height = 0)
```



Exemplo com `geom_count()`:

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
  geom_count(height = 0)
```


hwy
40
30
20
10
15
20
25
30
35
cty
n
5
10

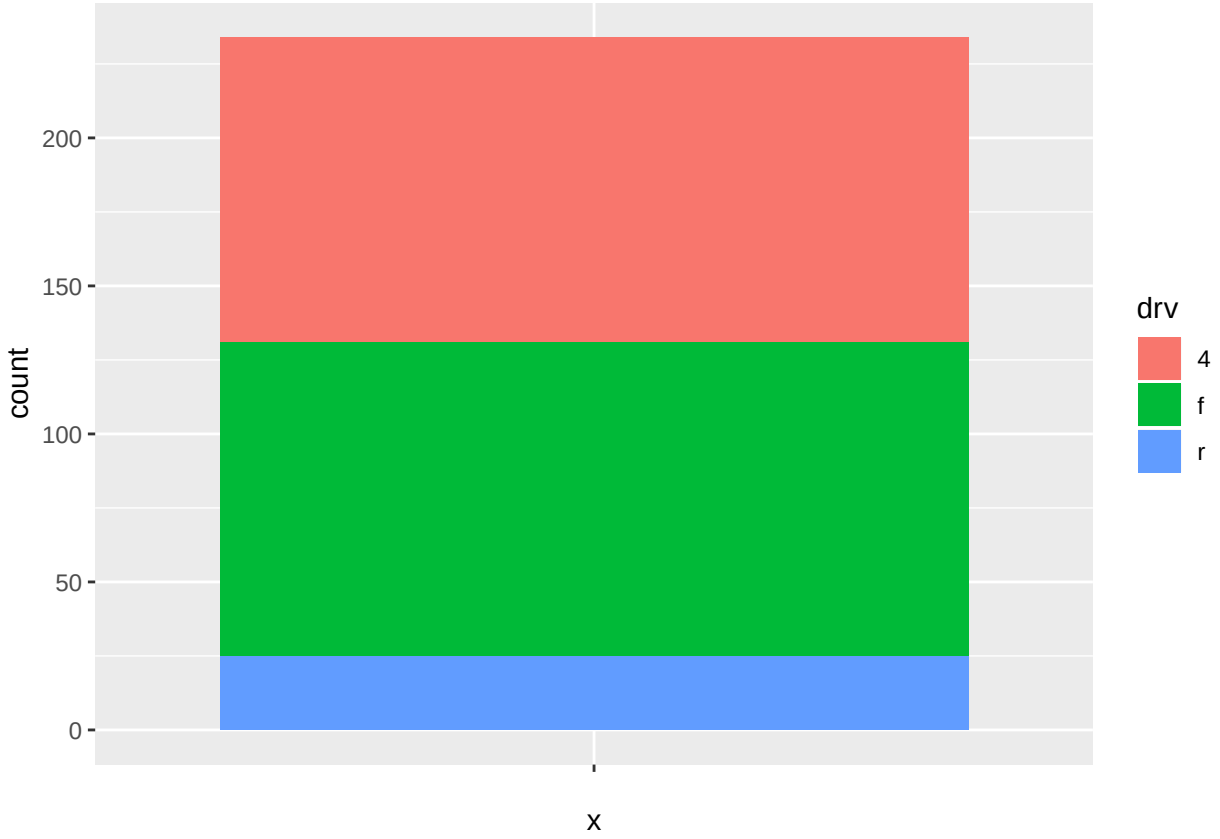
Seção 3.9.1

Exercício 1

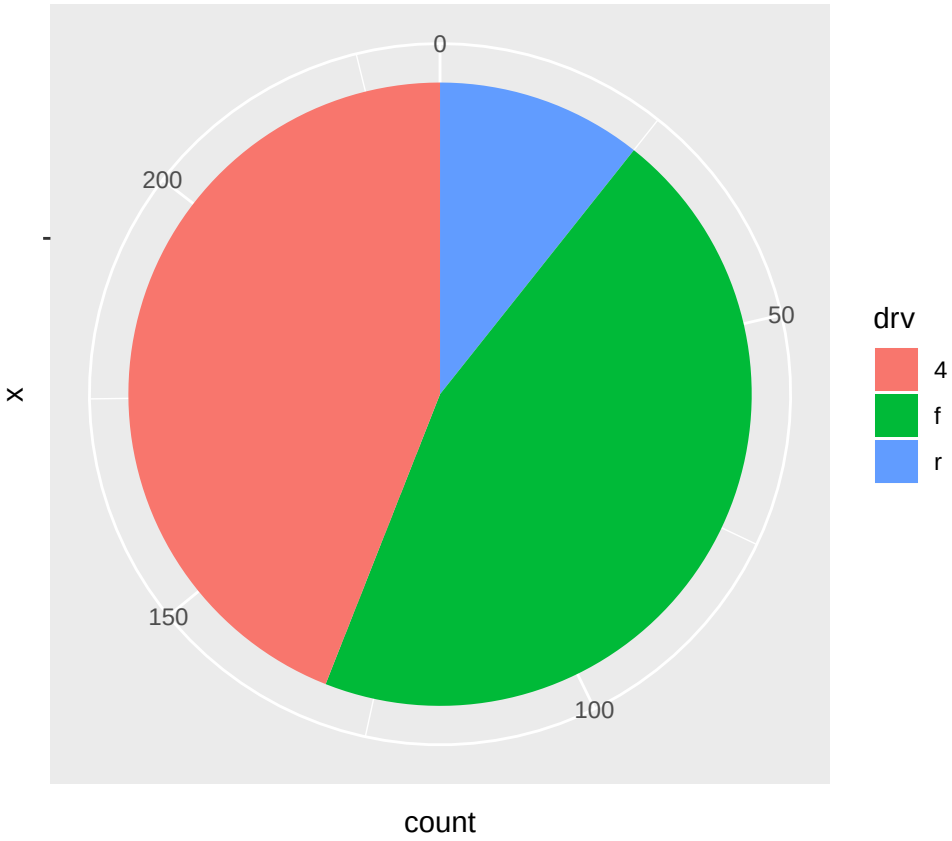
P: Transforme um gráfico de barras empilhadas em um gráfico de pizza usando `coord_polar()`.

R:

```
ggplot(data = mpg, aes(x = "", fill = drv)) +
  geom_bar()
```




```
ggplot(data = mpg, aes(x = "", fill = drv)) +
  geom_bar(width = 1) +
  coord_polar(theta = "y")
```


0
50
100
150
200
x
count
drv
4
f
r

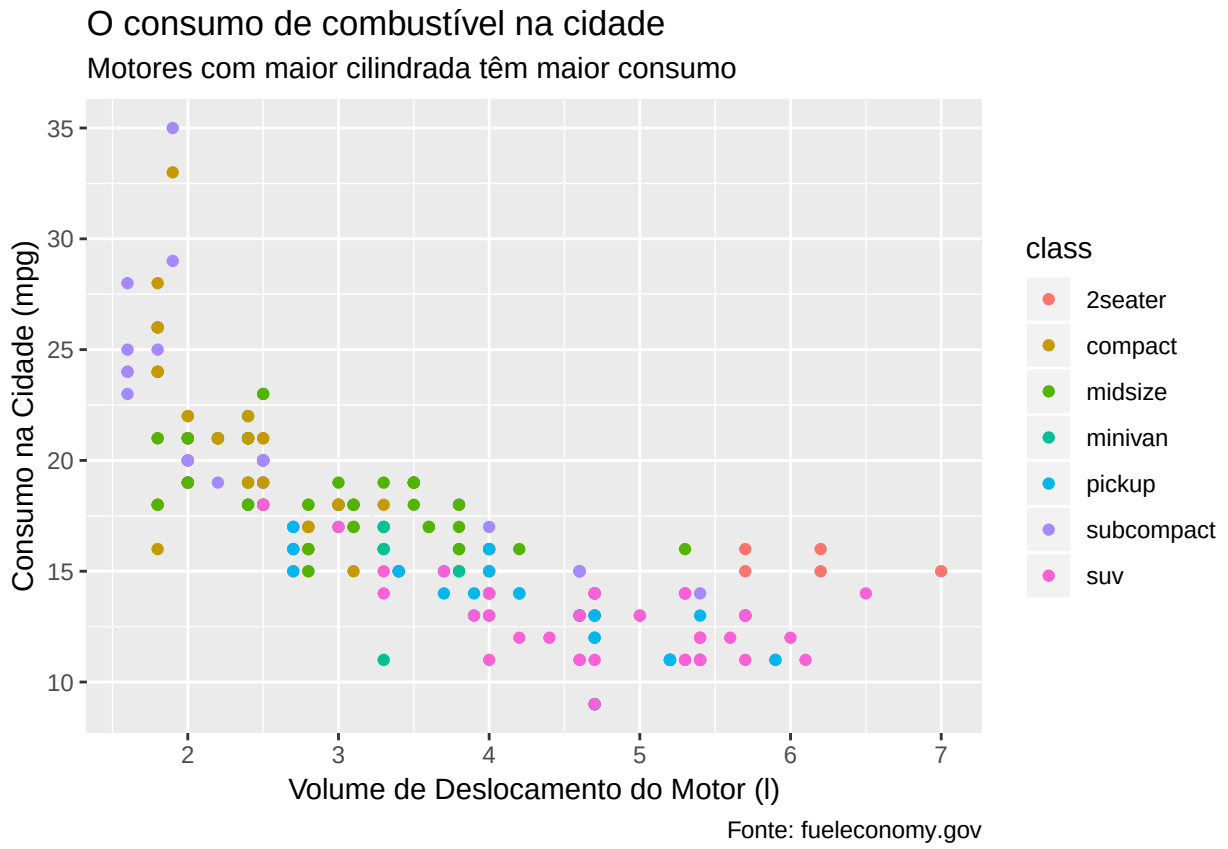
Seção 28.2.1

Exercício 1

P: Crie um gráfico com dados de economia de combustível com os parâmetros título, subtítulo, legenda, eixos x e y e cor personalizados.

R:

```
ggplot(data = mpg, mapping = aes(x = displ, y = cty, color = class)) +
  geom_point() +
  labs(
    title = "O consumo de combustível na cidade",
    subtitle = "Motores com maior cilindrada têm maior consumo",
    caption = "Fonte: fueleconomy.gov",
    x = "Volume de Deslocamento do Motor (l)",
    y = "Consumo na Cidade (mpg)")
```



Seção 28.3.1

Exercício 3

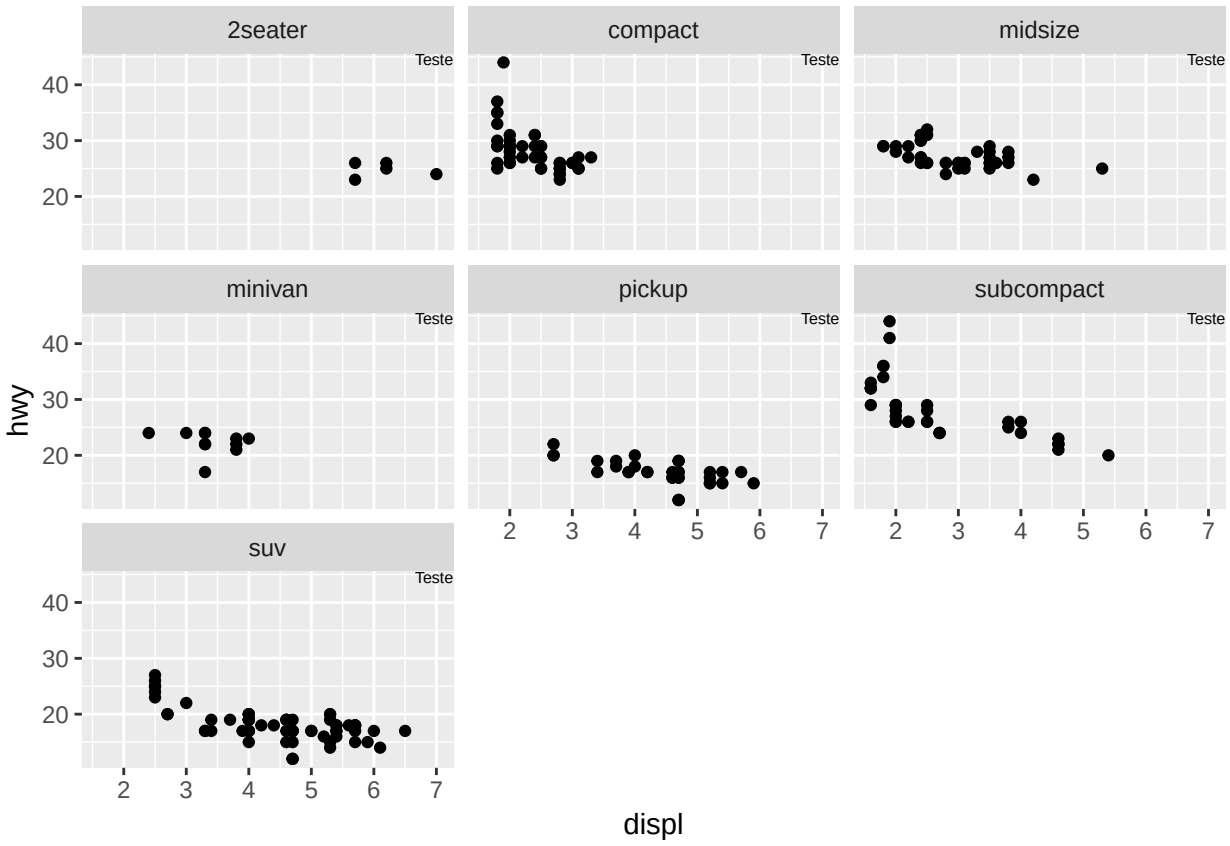
P: Como rótulos com `geom_text()` interagem com *faceting*? Como você pode adicionar um rótulo a uma única faceta? Como você pode colocar um rótulo diferente em cada faceta? (Dica: pense nos dados subjacentes.)

R: Os rótulos devem ser especificados e pode ser definida sua localização da faceta. Se a variável não for especificada, o texto é escrito em todas as facetas. Para colocar em uma única faceta, deve-se especificar qual será nos argumentos de `geo_text()`. Já para adicionar um rótulo diferente para cada faceta basta ter as variáveis e indicar os textos.

Exemplos:

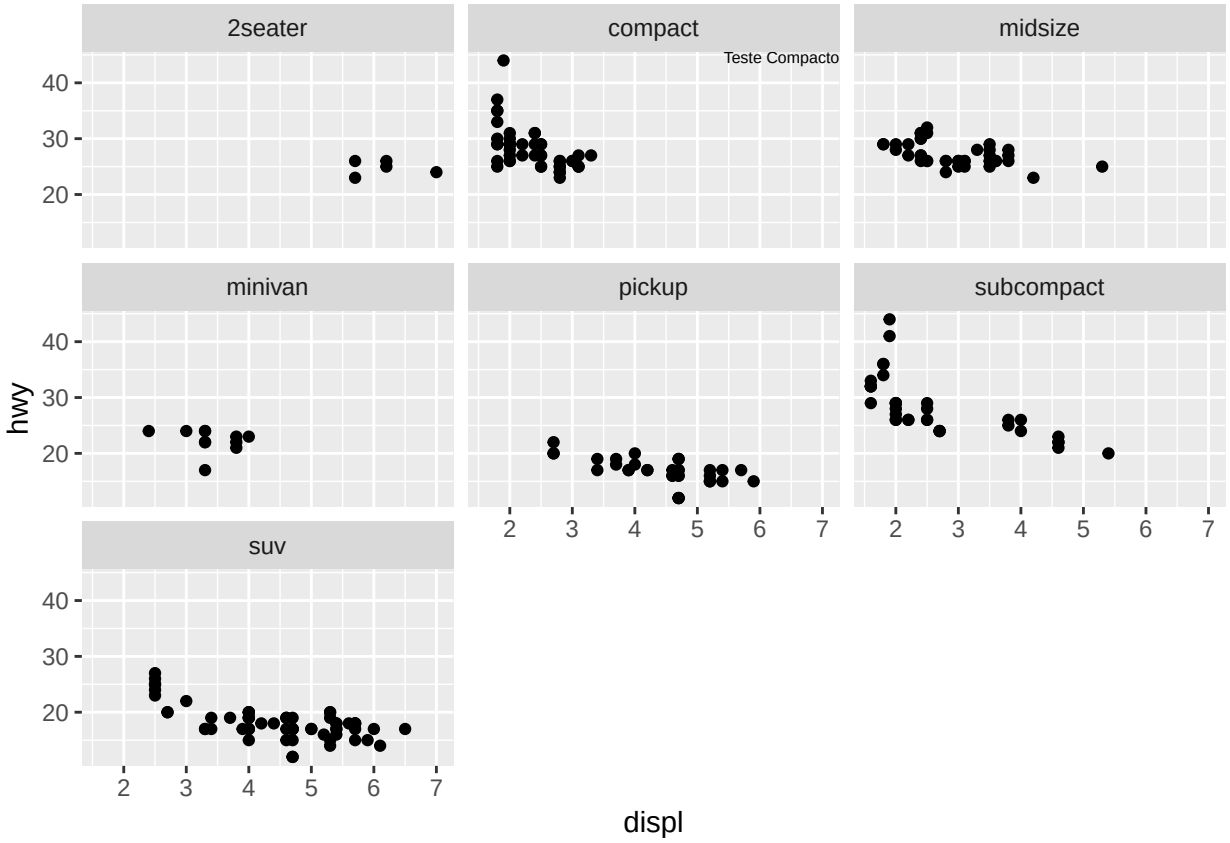
Rótulo em todas as facetas.

```
label <- tibble(
  displ = Inf,
  hwy = Inf,
  label = "Teste")

ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_text(aes(label = label),
    data = label, vjust = "top", hjust = "right", size = 2) +
  facet_wrap(~class)
```



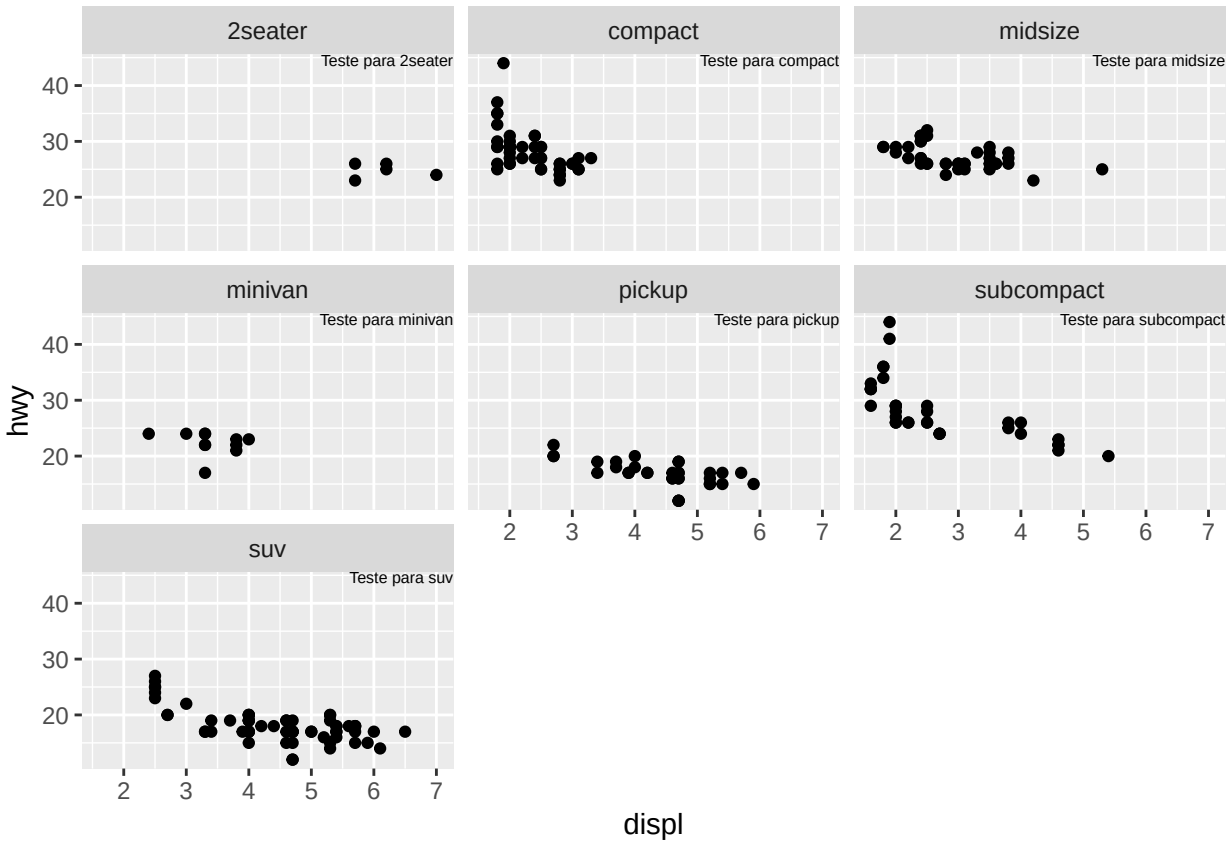
Rótulo para apenas uma faceta.

```
label <- tibble(
  displ = Inf,
  hwy = Inf,
  class = "compact",
  label = "Teste Compacto")

ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_text(aes(label = label), data = label, vjust = "top", hjust = "right", size = 2) +
  facet_wrap(~class)
```



Um rótulo para cada faceta.

```
label <- tibble(
  displ = Inf,
  hwy = Inf,
  class = unique(mpg$class),
  label = paste0("Teste para ", class))

ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_text(aes(label = label), data = label, vjust = "top", hjust = "right", size = 2) +
  facet_wrap(~class)
```


2seater
Teste para 2seater
compact
Teste para compact
midsize
Teste para midsize
minivan
Teste para minivan
pickup
Teste para pickup
subcompact
Teste para subcompact
suv
Teste para suv
hwy
displ
40
30
20
2
3
4
5
6
7

Seção 28.4.4

Exercício 2

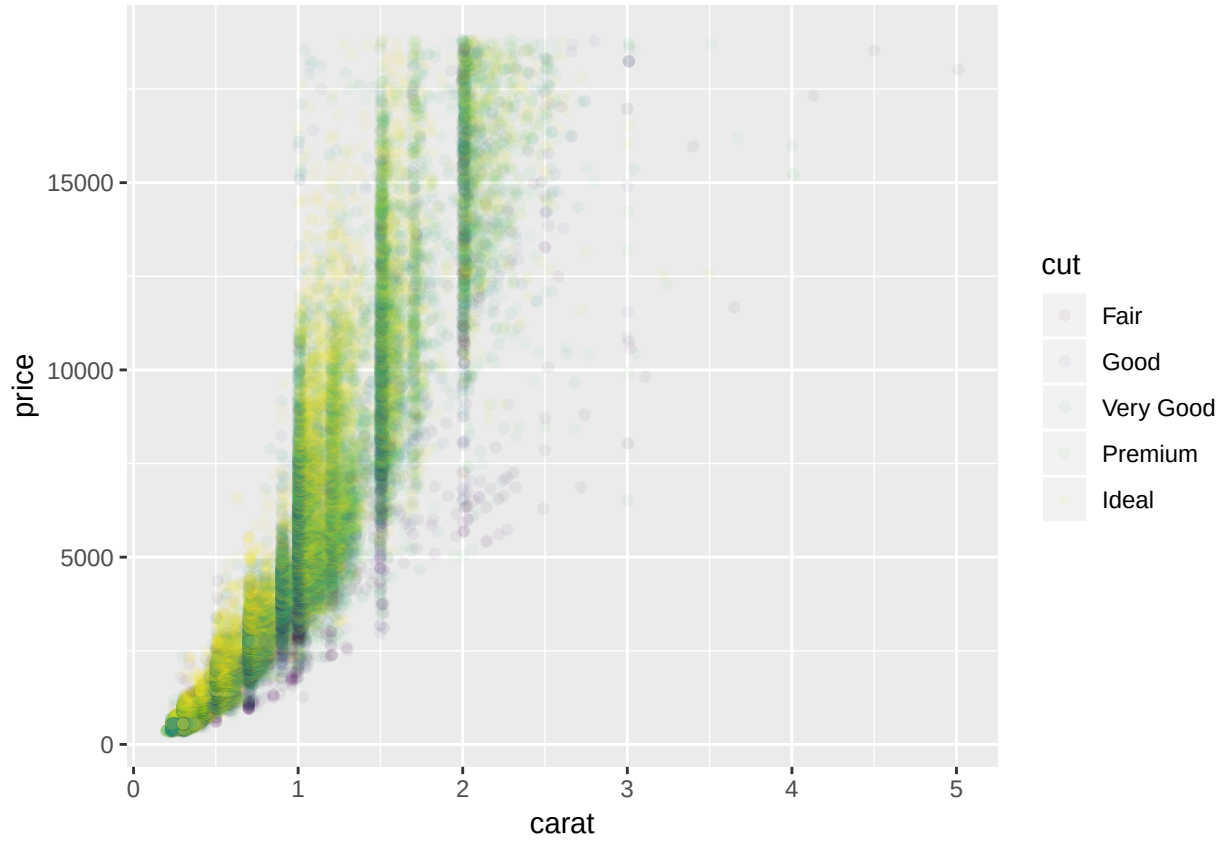
P: Qual é o primeiro argumento para todas as escalas? Como isso se compara aos `labs()`?

R: O primeiro argumento para cada escala é o rótulo da escala. É equivalente a usar a função `labs()`.

Exercício 4

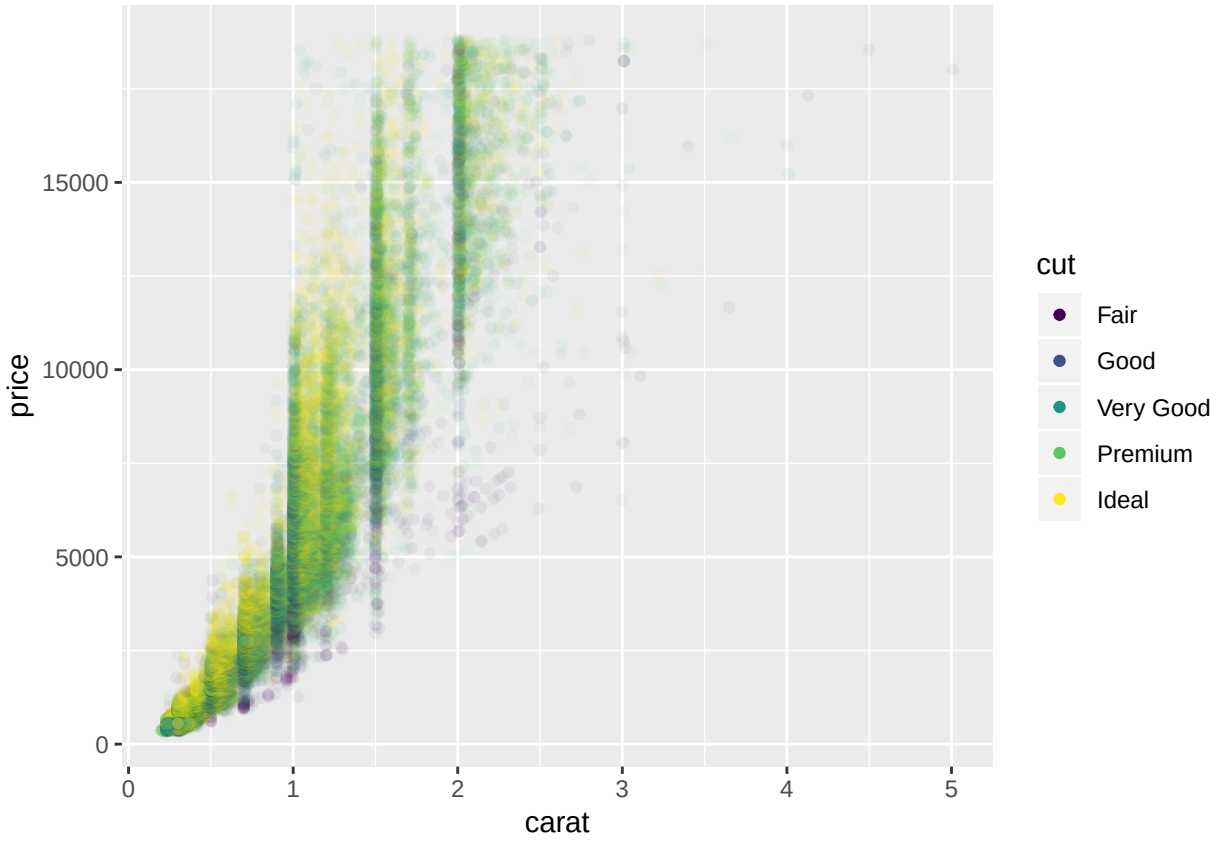
P: Use `override.aes` para facilitar a visualização da legenda da próxima plotagem.

```
ggplot(diamonds, aes(carat, price)) +
  geom_point(aes(color = cut), alpha = 1/20)
```



R:

```
ggplot(diamonds, aes(carat, price)) +
  geom_point(aes(color = cut), alpha = 1/20) +
  guides(color = guide_legend(nrow = 5, override.aes = list(alpha = 1)))
```


price
15000
10000
5000
0
0
1
2
3
4
5
carat
cut
Fair
Good
Very Good
Premium
Ideal