

Análise de Dados - UFPE/2019 - Lista 6

Maria Eduarda R. N. Lessa

21 de maio de 2019

Questão 1:

letra a)

Uma variável dependente é aquela que apresenta valores que dependem, em parte, daqueles assumidos por uma ou mais variáveis independentes.

letra b)

Uma variável independente é aquela que provoca variações nos valores de uma variável dependente.

letra c)

Como mencionado nos tópicos anteriores, as variáveis independentes são “responsáveis” por parte ou mesmo por toda a variação nos escores de uma variável dependente.

Questão 2:

Esta equação é chamada de modelo de regressão amostral.

Questão 3:

Y_i - representa um valor específico da variável dependente Y.

\hat{a} - representa o parâmetro do intercepto estimado - o ponto no qual a reta de regressão encontra o eixo Y ($x = 0$).

\hat{b} - representa o coeficiente de variação / parâmetro de inclinação estimado - a variação da média de Y quando aumentamos uma unidade na variável X.

X_i - representa um valor específico da variável independente X.

\hat{u} - representa o componente estocástico / aleatório / resíduo / erro amostral - é a distância entre um valor Y_i para o seu valor representado na reta de regressão.

Questão 4:

O componente sistemático da equação é \hat{Y}_i . Um componente sistemático é aquele que não é aleatório, ou seja, a sua evolução é previsível. \hat{Y}_i corresponderá a uma observação de Y para um valor específico de X ; o \hat{a} e o \hat{b} , que são constantes, determinarão o Y_i para cada unidade adicionada em X .

Questão 5:

O componente estocástico da equação é \hat{u} . Um componente estocástico é aquele que “evolui ao longo do tempo de maneira aleatória e imprevisível”, é isto que acontece como valor que o \hat{u} representa (que é a diferença entre o valor observado Y_i e o valor correspondente na linha de regressão).

Referência: http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0611740_08_cap_03.pdf

Questão 6:

O Y_i é o valor observado da variável Y para um determinado valor de X , enquanto o \hat{Y}_i é a representação deste ponto na reta de regressão; por isso, $Y_i = \hat{Y}_i + \hat{u}$.

Questão 7:

O método do *Ordinary Least Squares* (OLS) busca encontrar a menor soma dos quadrados da diferença entre \hat{Y}_i e Y_i (esta diferença é chamada de resíduo), para que a distância entre o conjunto dos pontos da reta de regressão e os seus respectivos valores observados seja a menor possível. É utilizado para ajustar um modelo aos dados observados, capaz de apresentar um padrão de relação entre as variáveis.

Questão 8:

letra a)

Para nomear os arquivos, é recomendável que o nome escolhido faça referência aos dados analisados, que as diferentes palavras no nome do arquivo sejam separadas por “**_” e que estes sejam salvos na extensão “.R”**.

letra b)

Para identificar as variáveis, recomenda-se que nos casos em que há mais de uma palavra, estas sejam escritas em letras minúsculas e separadas por “.” (ex: uma.variavel). No caso de funções, é recomendado que as diferentes palavras sejam identificadas com a primeira letra maiúscula (ex: UmaFuncao). Para constantes, a recomendação é a mesma das funções, porém, a letra “k” é adicionada ao início do nome (ex: kUmaConstante).

letra c)

Para quebrar (ou dividir) um código, recomenda-se utilizar dois espaços na linha seguinte à quebra, uma exceção é quando esta quebra ocorre dentro de parênteses; recomenda-se, então, que a linha de baixo esteja alinhada como primeiro caracter dentro dos parênteses da linha acima.

letra d)

É recomendado adicionar espaços antes e depois de operadores binários, tais como “=”, “+”, “-”, “<-”. No caso da vírgula, deve-se adicionar espaço apenas após o sinal. Não deve-se utilizar espaços para funções entre colchetes ou parenteses, com exceção dos espaços após avírgula, como anteriormente mencionado.

letra e)

Para atribuir valores a um identificador, nunca utilizar “=”, mas sim “<-”.

letra f)

Para adicionar comentários ao código, utilizar “#” seguido por um espaço.

letra g)

Recomenda-se, primeiramente, listar um argumento “vazio” e, depois, atribuir a este argumento uma função com seus respectivos valores. É possível adicionar mais de um argumento por linha, no entanto, é necessário atentar para as quebras de linha, que devem ser feitas apenas entre os diferentes argumentos (e não dividi-los ao meio).

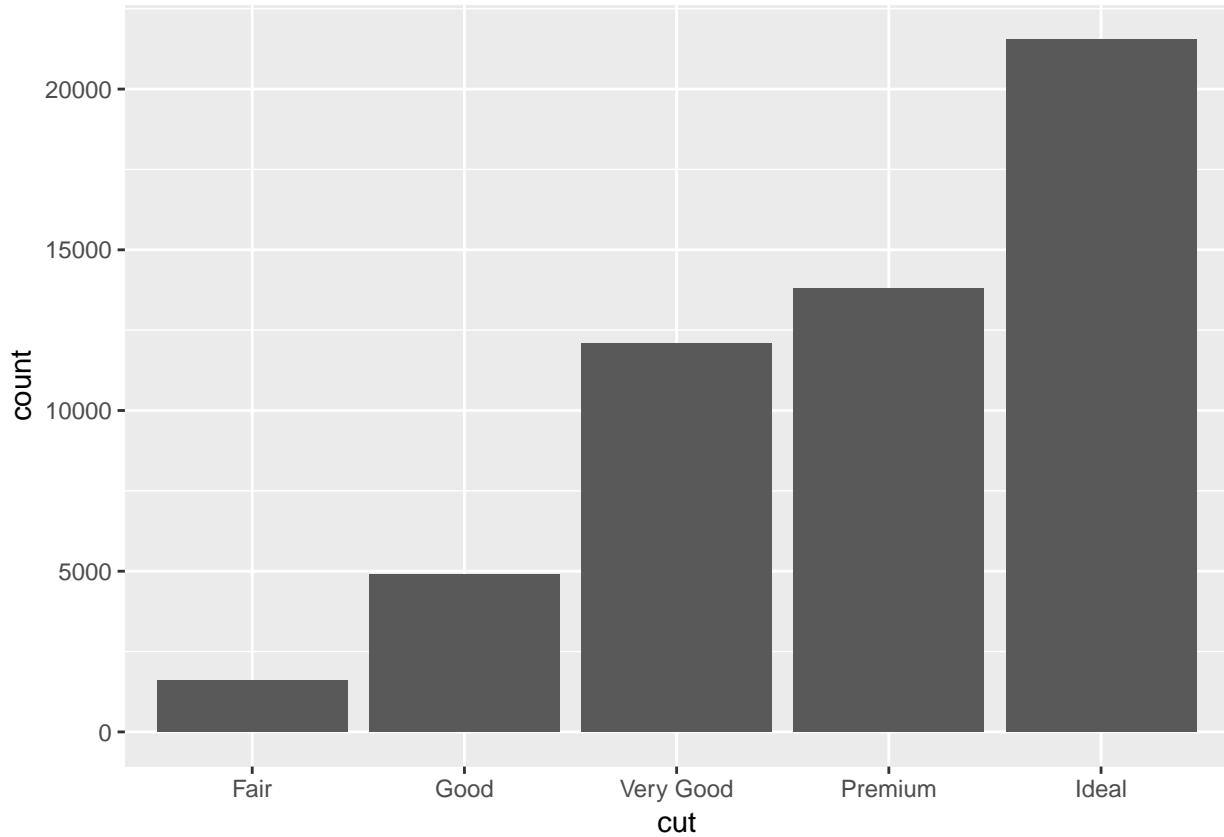
letra h)

Recomenda-se que uma seção de comentários sobre uma função seja adicionada na linha subsequente ao comando desta. Nos comentários, deve haver uma lista com os argumentos da função (introduzidos pela notação “Args:”) e a definição dos valores retornados que são esperados (“Returns:”).

Questão 9:

Exemplo 7.3.1:

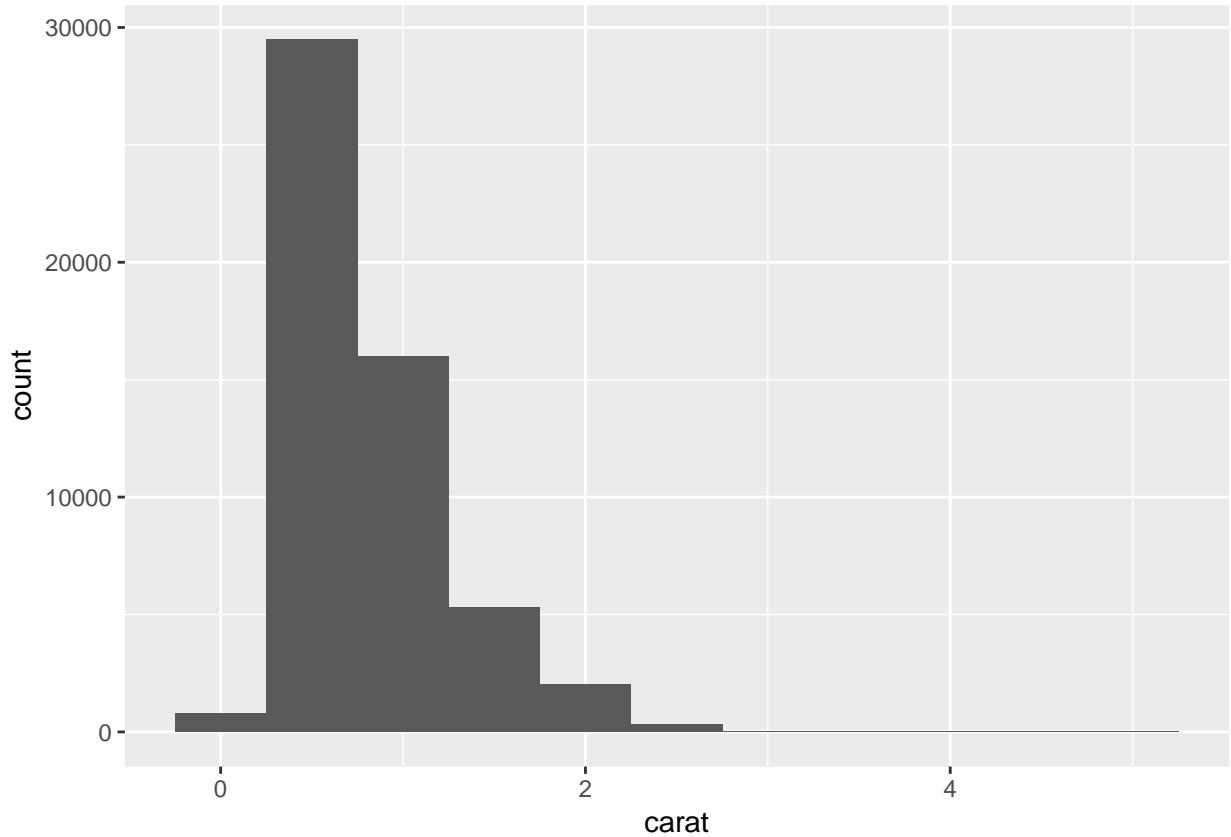
```
# Requerer funções do pacote "tidyverse":  
library(tidyverse)  
  
## Warning: package 'tidyverse' was built under R version 3.5.3  
  
## -- Attaching packages ----- tidyverse 1.2.1 --  
  
## v ggplot2 3.1.1      v purrr   0.3.2  
## v tibble  2.1.1      v dplyr    0.8.0.1  
## v tidyr   0.8.3      v stringr  1.4.0  
## v readr   1.3.1      vforcats  0.4.0  
  
## Warning: package 'ggplot2' was built under R version 3.5.3  
  
## Warning: package 'tibble' was built under R version 3.5.3  
  
## Warning: package 'tidyr' was built under R version 3.5.3  
  
## Warning: package 'readr' was built under R version 3.5.3  
  
## Warning: package 'purrr' was built under R version 3.5.3  
  
## Warning: package 'dplyr' was built under R version 3.5.3  
  
## Warning: package 'forcats' was built under R version 3.5.3  
  
## -- Conflicts ----- tidyverse_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag()   masks stats::lag()  
  
# Criar gráfico de barras da variável categórica "cut" - da base "diamonds"  
# com a função "ggplot" - para analisar a sua distribuição:  
ggplot(data = diamonds) +  
  geom_bar(mapping = aes(x = cut))
```



```
# Verificar a distribuição da variável "cut":  
diamonds %>%  
  count(cut)
```

```
## # A tibble: 5 x 2  
##   cut      n  
##   <ord>    <int>  
## 1 Fair     1610  
## 2 Good     4906  
## 3 Very Good 12082  
## 4 Premium   13791  
## 5 Ideal    21551
```

```
# Criar histograma da variável contínua "carat" - da base "diamonds"  
# com a função "ggplot" - para analisar a sua distribuição:  
ggplot(data = diamonds) +  
  geom_histogram(mapping = aes(x = carat), binwidth = 0.5)
```



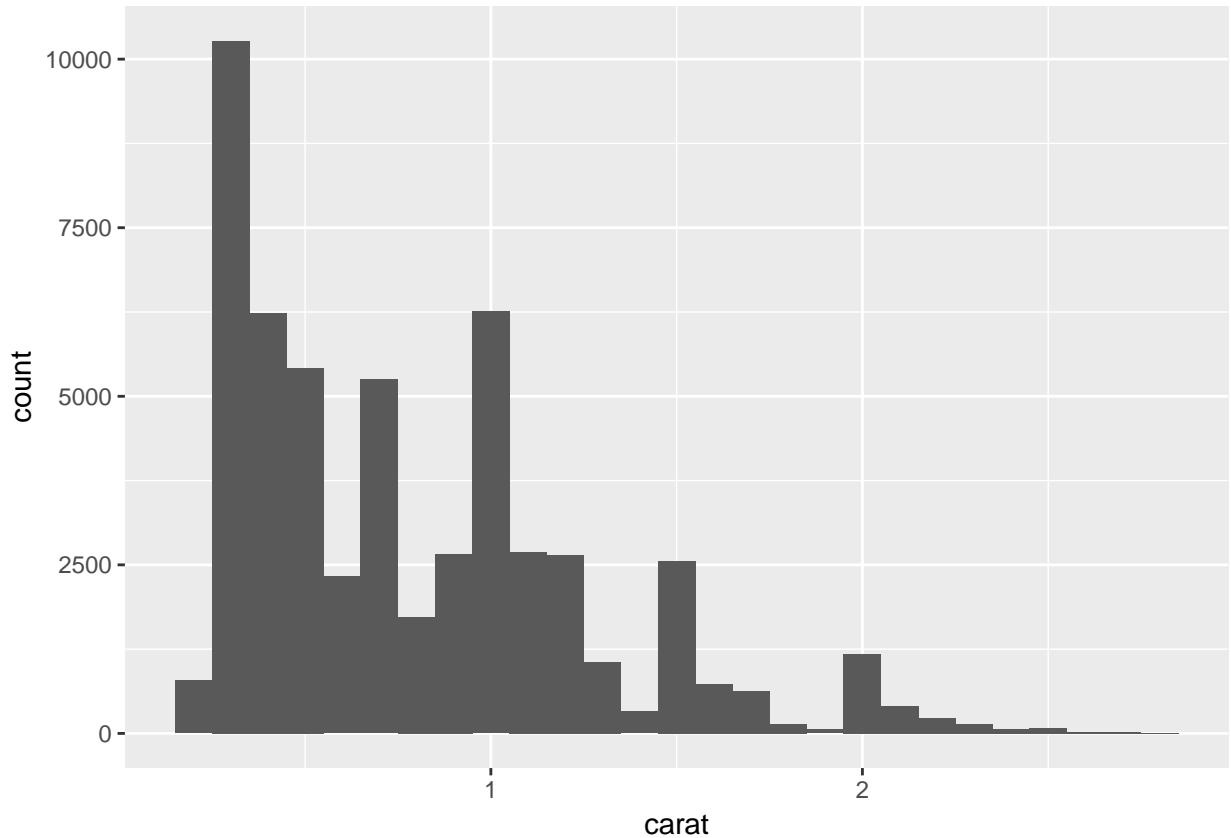
```
# Verificar a distribuição da variável "carat":  
diamonds %>%  
  count(cut_width(carat, 0.5))
```

```
## # A tibble: 11 x 2  
##   `cut_width(carat, 0.5)`     n  
##   <fct>                  <int>  
## 1 [-0.25,0.25]            785  
## 2 (0.25,0.75]           29498  
## 3 (0.75,1.25]           15977  
## 4 (1.25,1.75]            5313  
## 5 (1.75,2.25]            2002  
## 6 (2.25,2.75]             322  
## 7 (2.75,3.25]              32  
## 8 (3.25,3.75]                5  
## 9 (3.75,4.25]                4  
## 10 (4.25,4.75]               1  
## 11 (4.75,5.25]               1
```

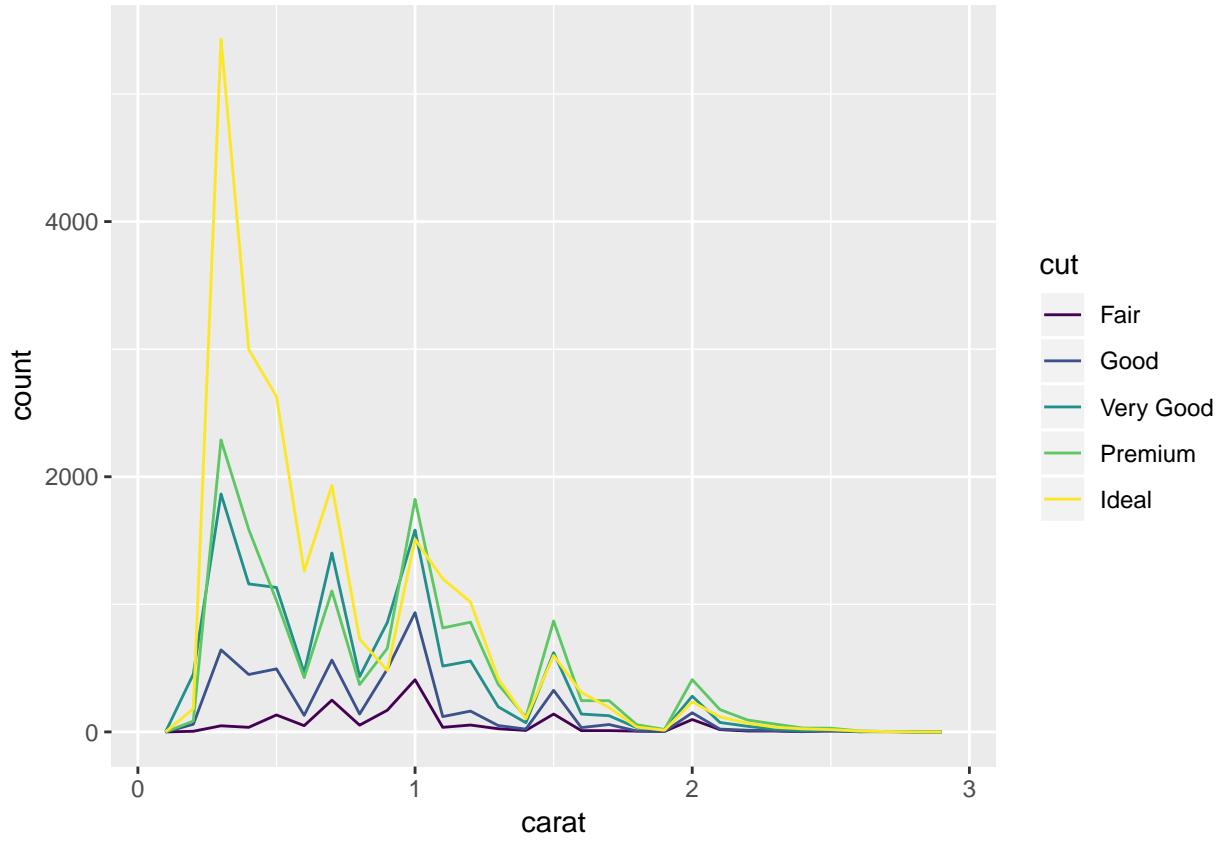
```
# Aplicar filtro para analisar apenas diamantes menores do que três quilates:  
smaller <- diamonds %>%  
  filter(carat < 3)
```

```
# Diminuir intervalos do histograma e gerar gráfico da base filtrada  
# com "ggplot":
```

```
ggplot(data = smaller, mapping = aes(x = carat)) +  
  geom_histogram(binwidth = 0.1)
```

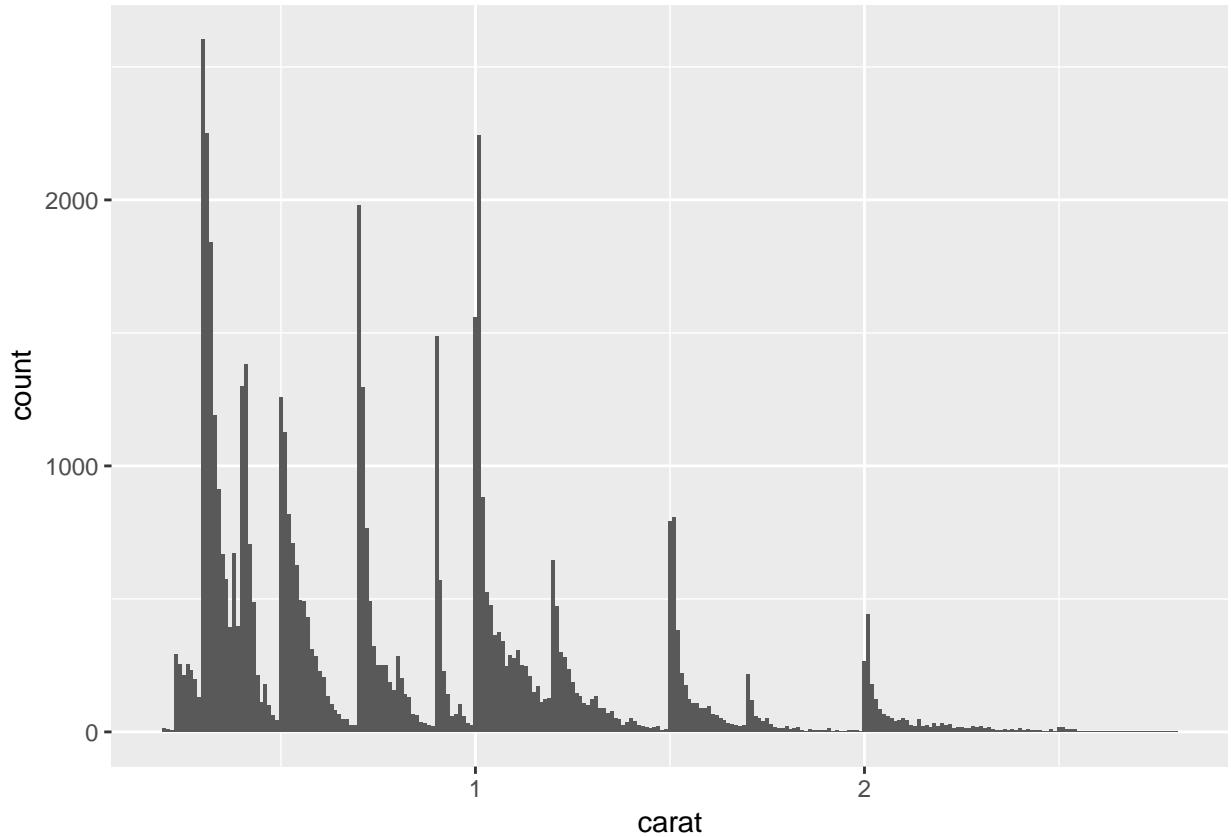


```
# Sobrepor histogramas dos diferentes quilates (na forma de linhas) e atribuir  
# diferentes cores aos valores da variável "cut":  
ggplot(data = smaller, mapping = aes(x = carat, colour = cut)) +  
  geom_freqpoly(binwidth = 0.1)
```

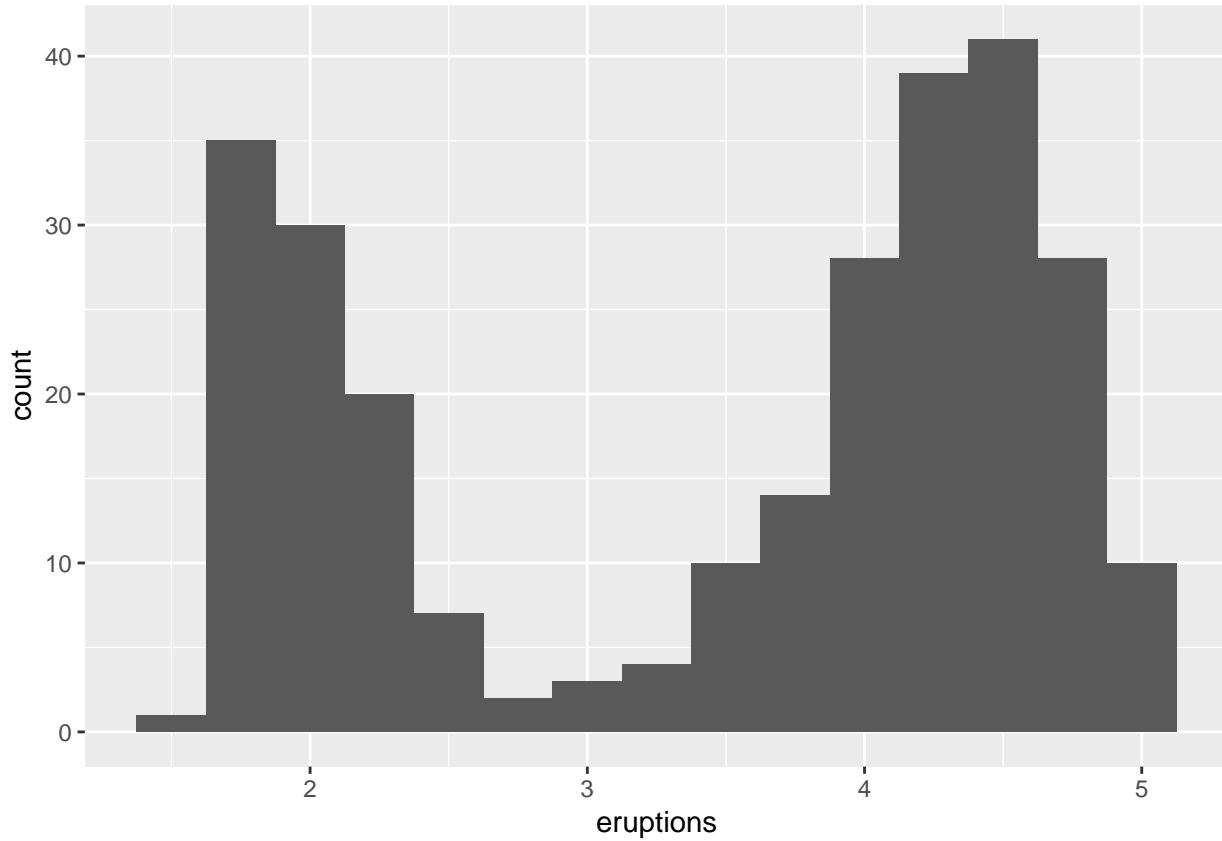


Exemplo 7.3.2:

```
# Diminuir intervalos do histograma e gerar gráfico da base
# filtrada com "ggplot":
ggplot(data = smaller, mapping = aes(x = carat)) +
  geom_histogram(binwidth = 0.01)
```

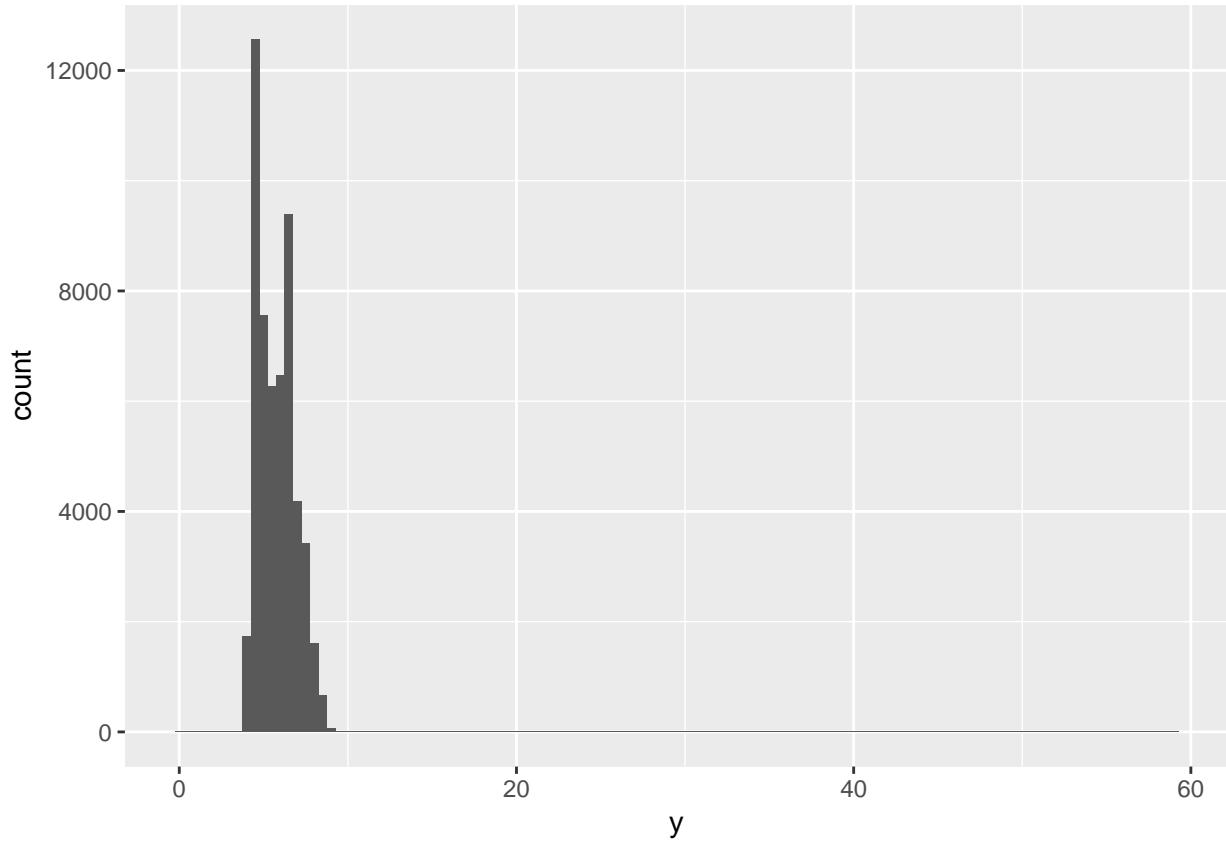


```
# Gerar histograma com "ggplot" para a base que contém as observações  
# acerca das erupções do geyser "Old Faithful", nomeada "faithful":  
ggplot(data = faithful, mapping = aes(x = eruptions)) +  
  geom_histogram(binwidth = 0.25)
```

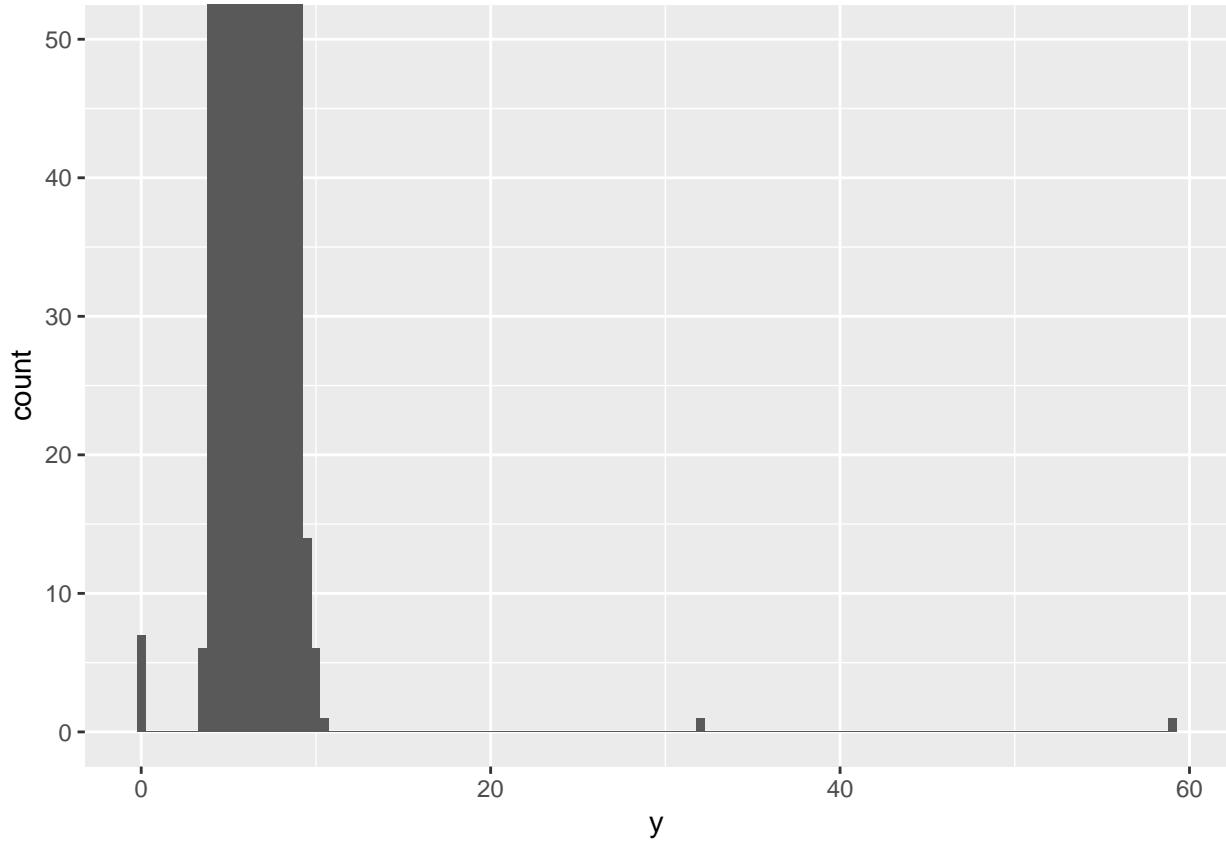


Exemplo 7.3.3:

```
# Gerar histograma com "ggplot" para a base "diamonds", a fim de
# identificar possíveis outliers:
ggplot(diamonds) +
  geom_histogram(mapping = aes(x = y), binwidth = 0.5)
```



```
# Identificar valores pequenos (pouco frequentes) no eixo Y:  
ggplot(diamonds) +  
  geom_histogram(mapping = aes(x = y), binwidth = 0.5) +  
  coord_cartesian(ylim = c(0, 50))
```



```
# Checar outliers (foi possível notar 3 valores considerados outliers
# a partir do histograma anterior, 0, ~30 e ~60).
# Filtrar base para detectar as observações com valores extremos,
# selecionar apenas variáveis "price", "x", "y" e "z" e ordenar por
# valores de "y", nomear base "unusual":
unusual <- diamonds %>%
  filter(y < 3 | y > 20) %>%
  select(price, x, y, z) %>%
  arrange(y)

# Checar valores:
unusual

## # A tibble: 9 x 4
##   price      x      y      z
##   <int> <dbl> <dbl> <dbl>
## 1 5139     0     0     0
## 2 6381     0     0     0
## 3 12800    0     0     0
## 4 15686    0     0     0
## 5 18034    0     0     0
## 6 2130     0     0     0
## 7 2130     0     0     0
## 8 2075    5.15  31.8  5.12
## 9 12210   8.09  58.9  8.06
```

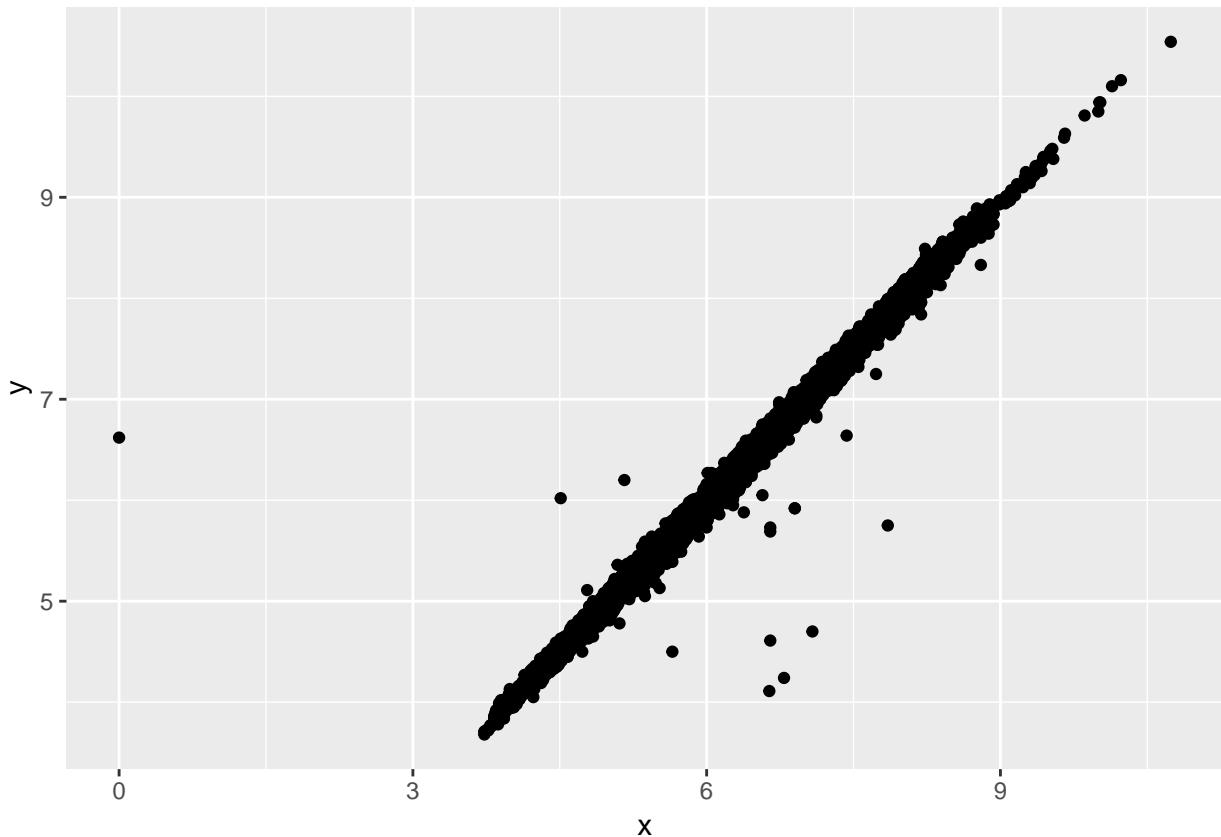
Exemplo 7.4:

```
# Filtrar base "diamonds", deixar na variável "y" apenas valores
# entre 3 e 20, nomear nova base "diamonds2":
diamonds2 <- diamonds %>%
  filter(between(y, 3, 20))

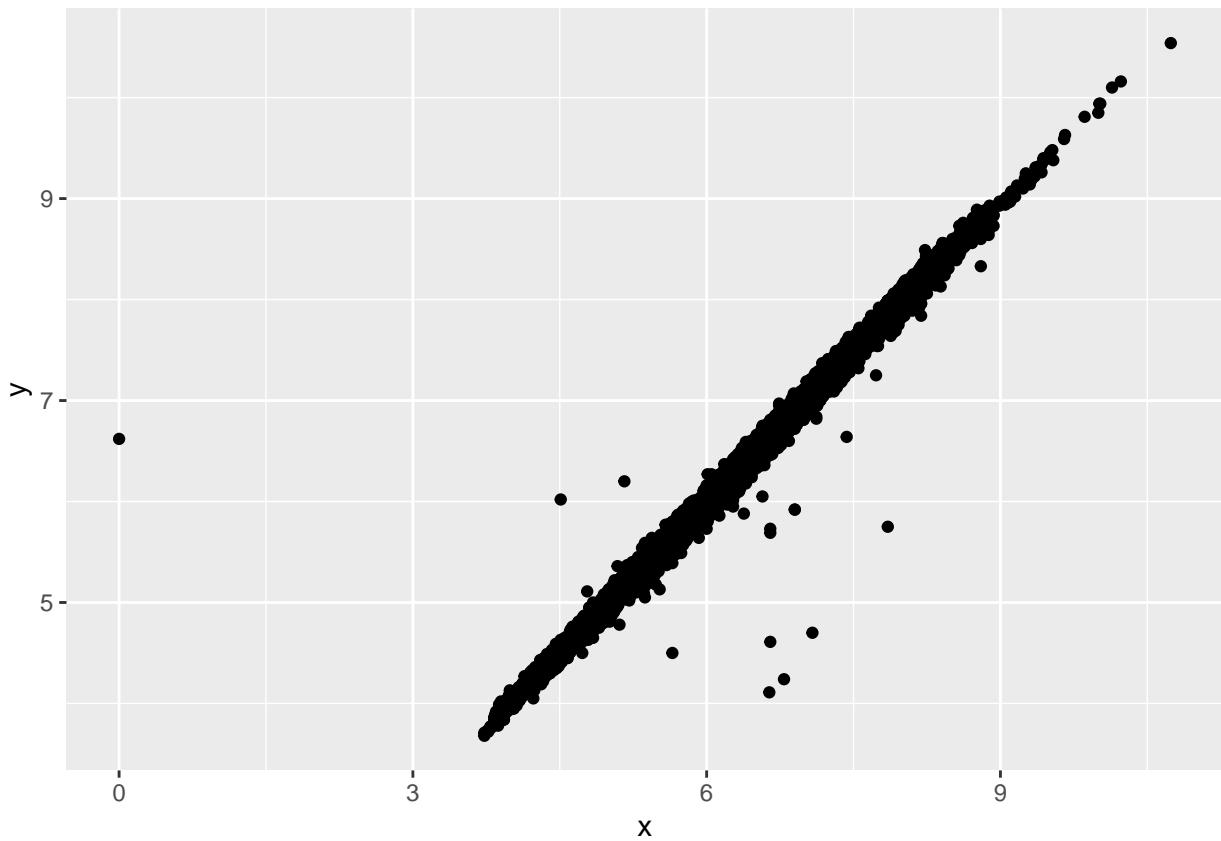
# Utilizar outro método para excluir valores extremos; substituí-los
# por "NA" através da função mutate e definir os valores a serem
# substituídos a partir da função "ifelse":
diamonds2 <- diamonds %>%
  mutate(y = ifelse(y < 3 | y > 20, NA, y))

# Gerar scatterplot da base "diamonds2" com "ggplot", que retornará o
# diagrama de dispersão e um aviso dos valores omitidos:
ggplot(data = diamonds2, mapping = aes(x = x, y = y)) +
  geom_point()
```

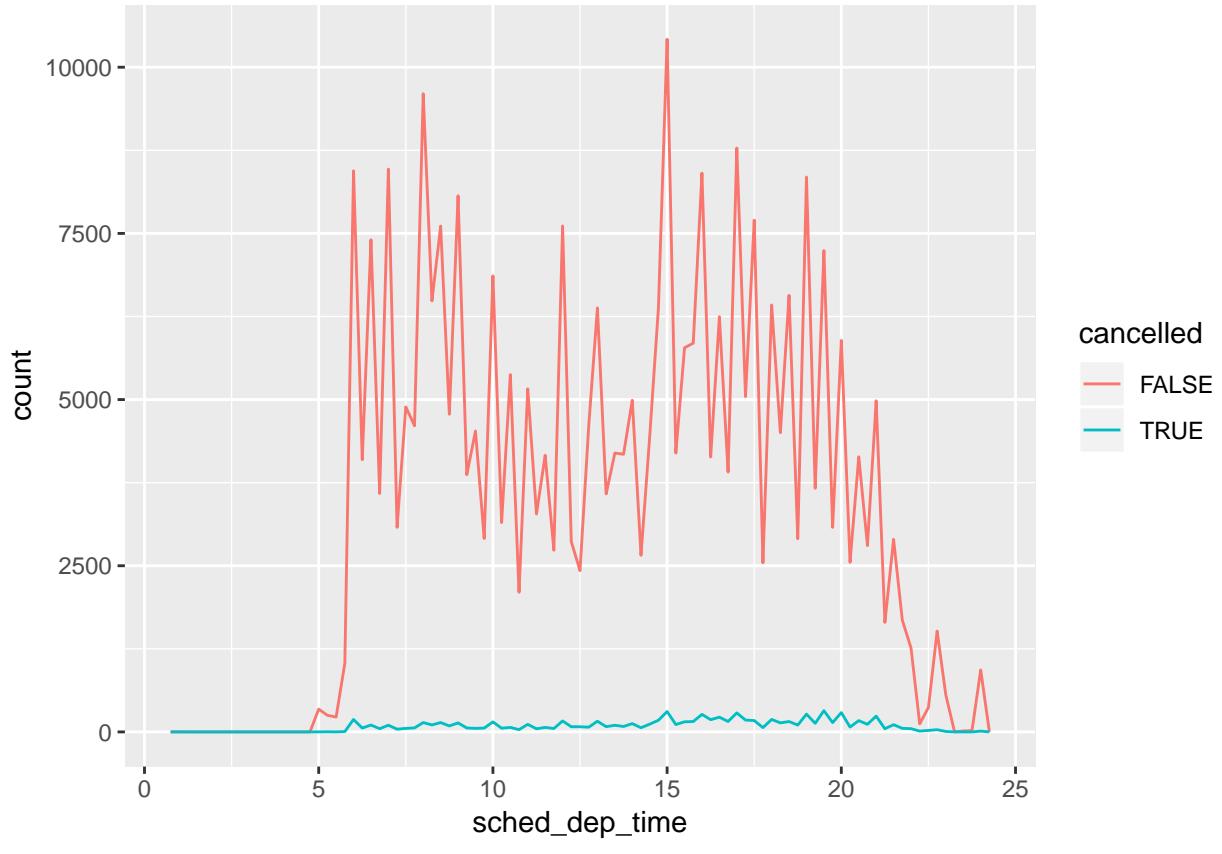
Warning: Removed 9 rows containing missing values (geom_point).



```
# Gerar scatterplot com "ggplot" e suprimir o aviso dos valores omitidos:
ggplot(data = diamonds2, mapping = aes(x = x, y = y)) +
  geom_point(na.rm = TRUE)
```

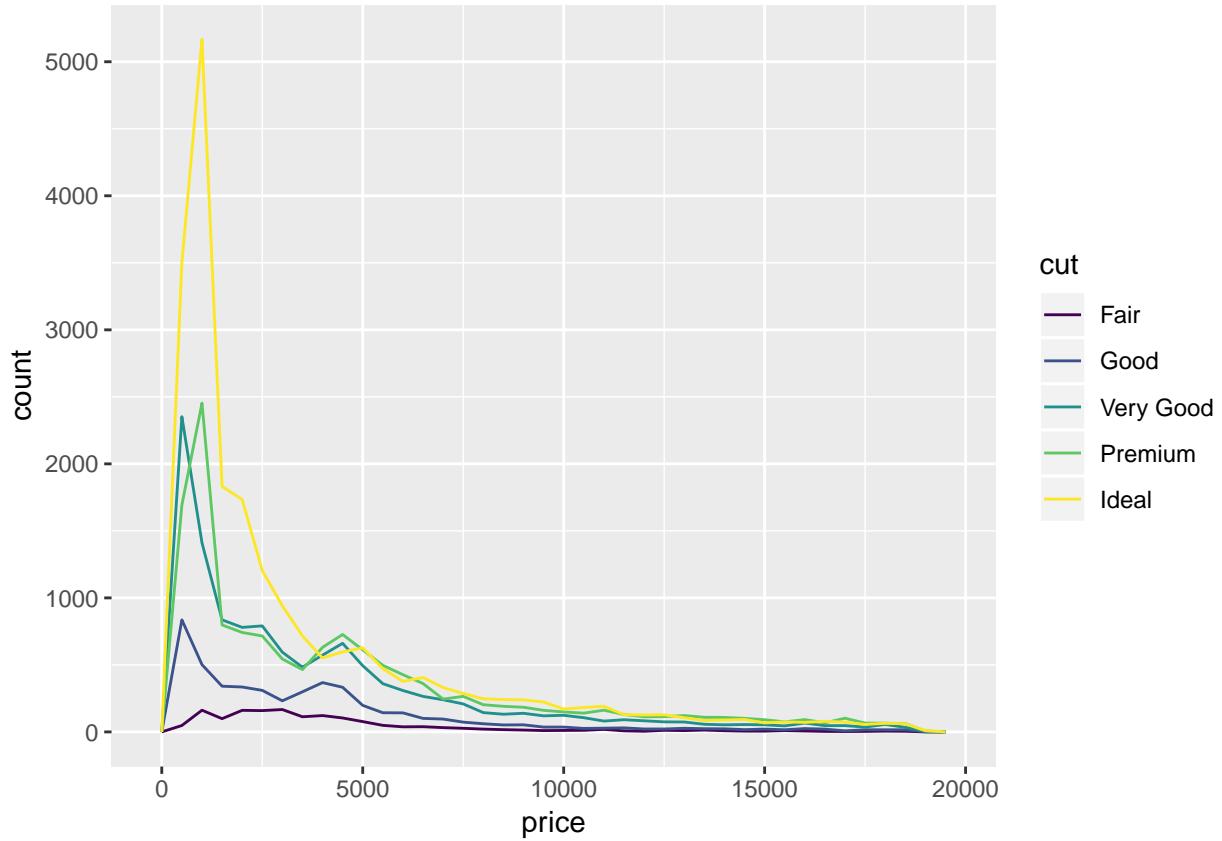


```
# Criar novas variáveis "cancelled" e "sched_dep_time" na base de dados
# "nycflights13" (nova variável "cancelled" criada a partir dos valores "NA"
# encontrados na variável existente "dep_time"; e nova variável
# "sched_dep_time" criada a partir das variáveis existentes "sched_hour"
# e "sched_min"). Gerar gráfico com função "ggplot" para plotar frequência de
# vôos cancelados versus não cancelados:
nycflights13::flights %>%
  mutate(
    cancelled = is.na(dep_time),
    sched_hour = sched_dep_time %/%
      100,
    sched_min = sched_dep_time %% 100,
    sched_dep_time = sched_hour + sched_min / 60
  ) %>%
  ggplot(mapping = aes(sched_dep_time)) +
  geom_freqpoly(mapping = aes(colour = cancelled), binwidth = 1/4)
```

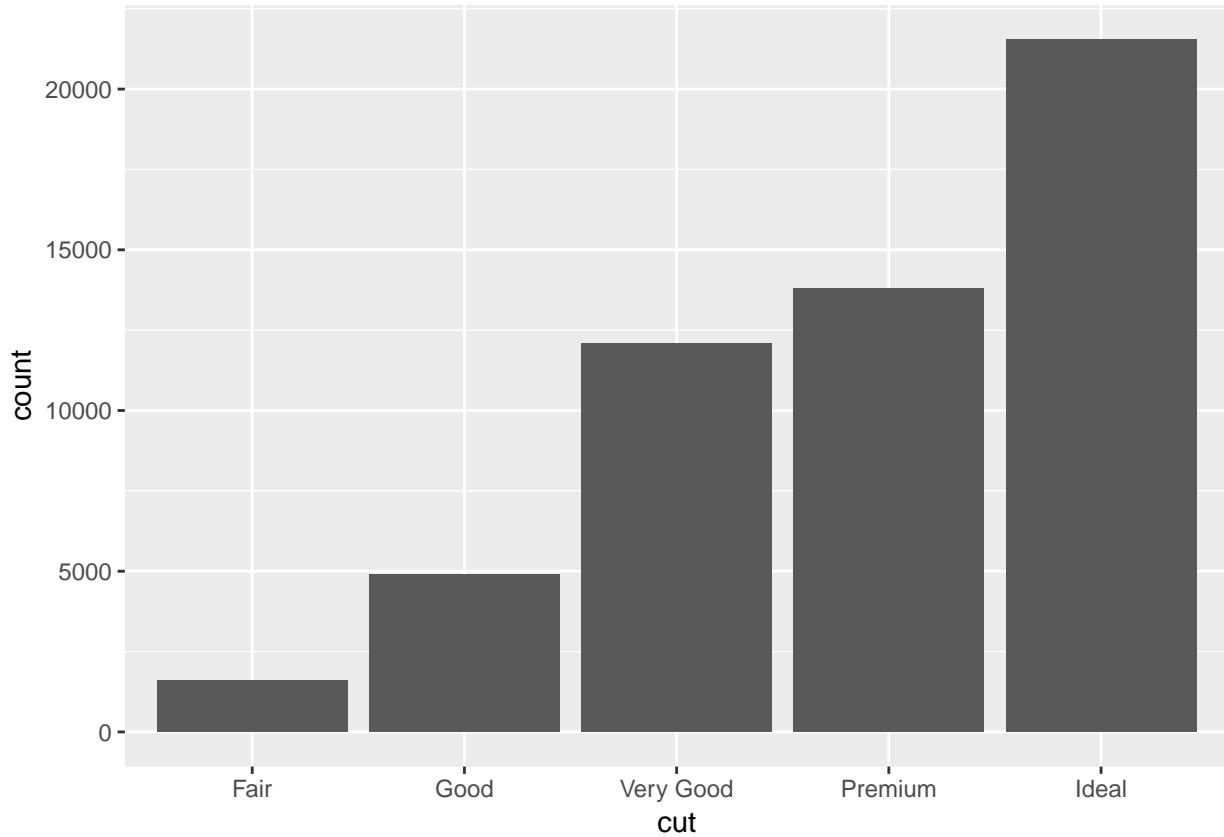


Exemplo 7.5.1:

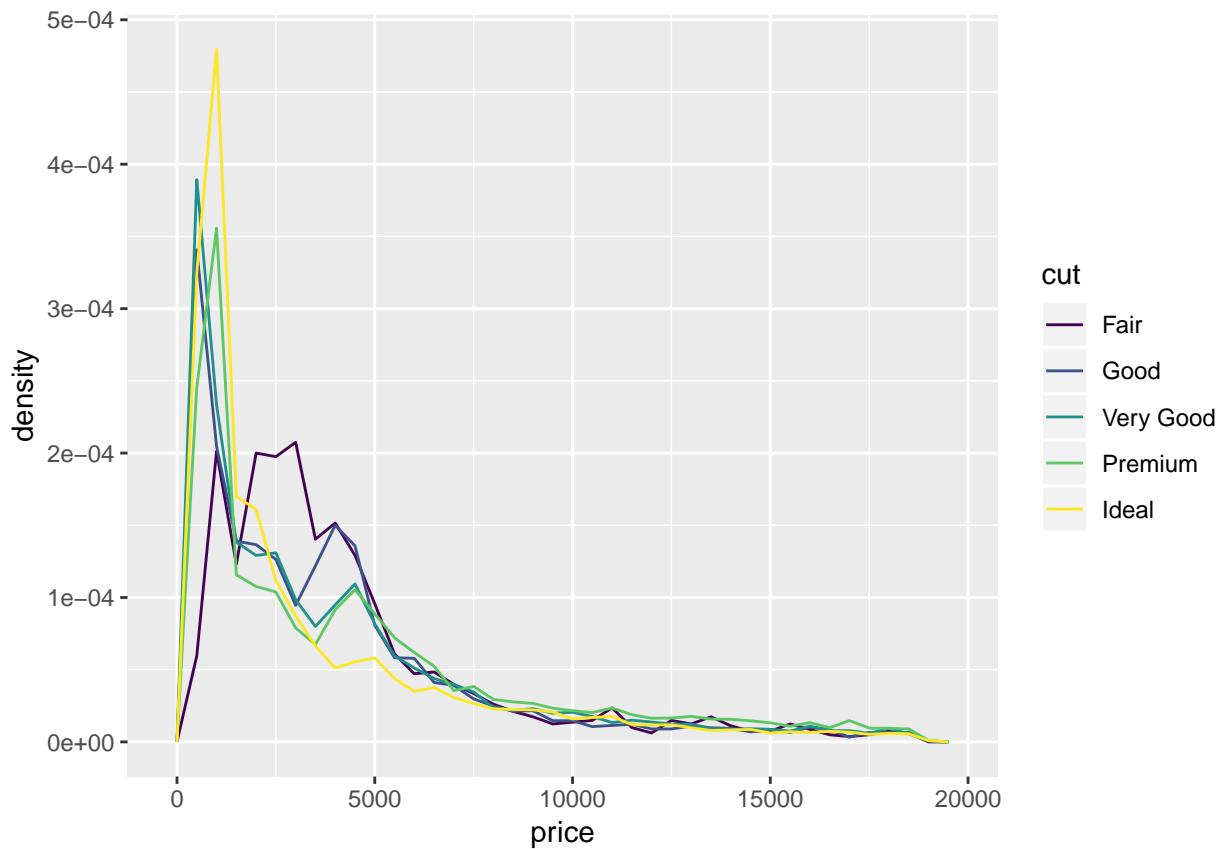
```
# Gerar gráfico de linhas para plotar a frequência de preços e a qualidade
# dos diamantes (base "diamonds") com a função "ggplot":
ggplot(data = diamonds, mapping = aes(x = price)) +
  geom_freqpoly(mapping = aes(colour = cut), binwidth = 500)
```



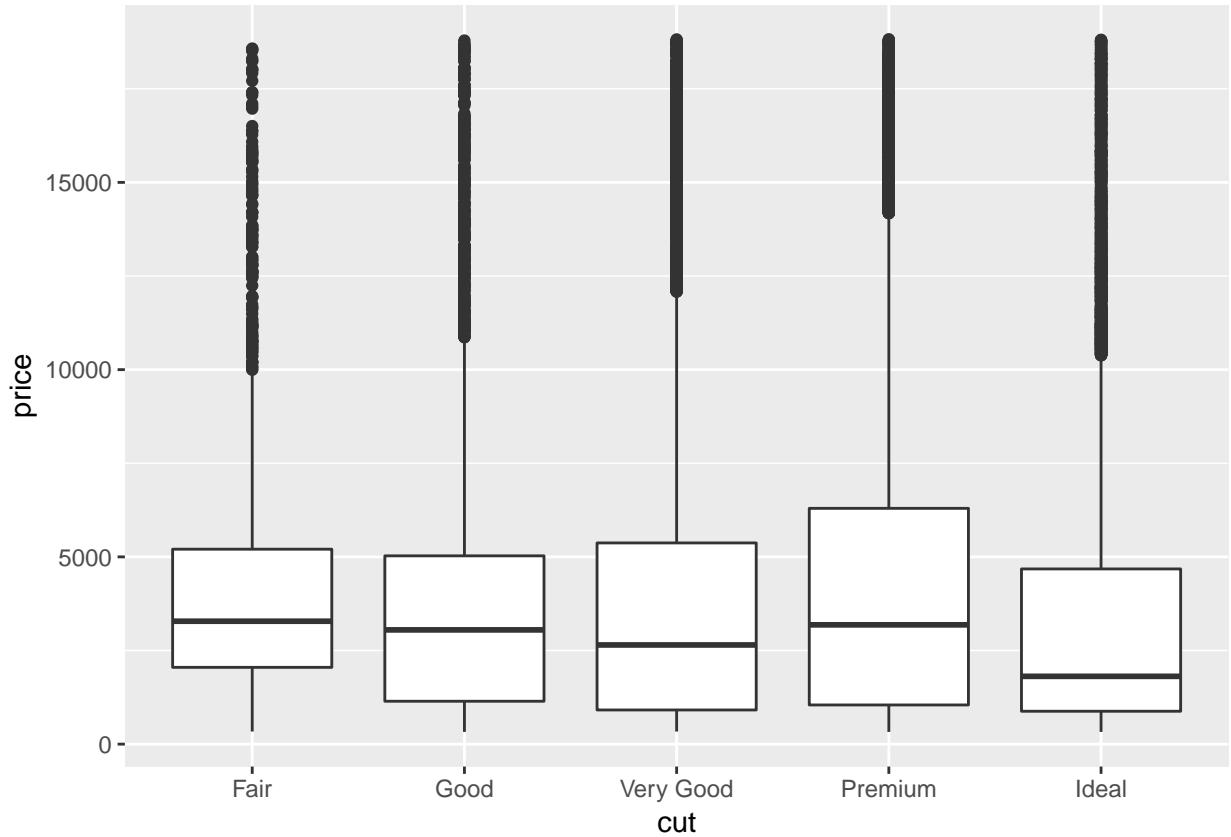
```
# Gerar gráfico de barras com "ggplot" para checar frequência da variável  
# "cut" da base "diamonds":  
ggplot(diamonds) +  
  geom_bar(mapping = aes(x = cut))
```



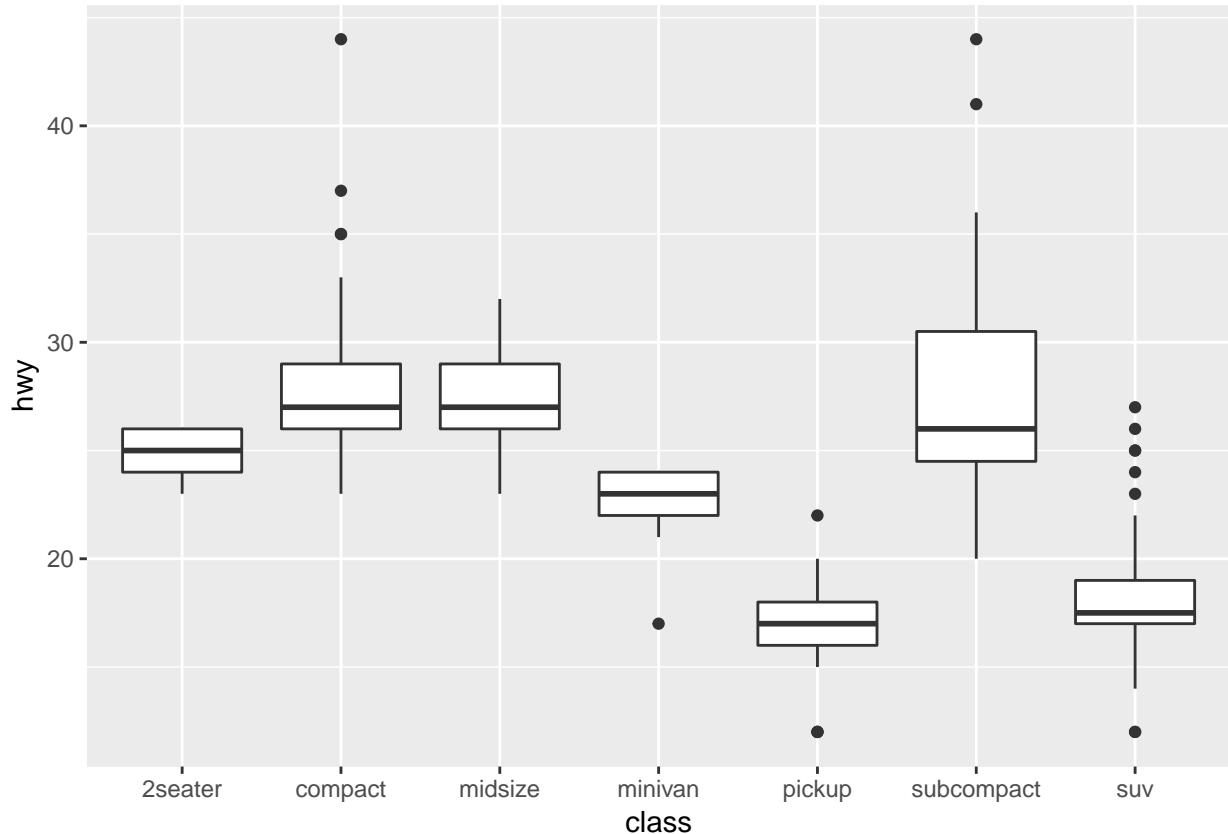
```
# Gerar gráfico de linhas para plotar a densidade de preços e a qualidade  
# dos diamantes (base "diamonds") com a função "ggplot",  
ggplot(data = diamonds, mapping = aes(x = price, y = ..density..)) +  
  geom_freqpoly(mapping = aes(colour = cut), binwidth = 500)
```



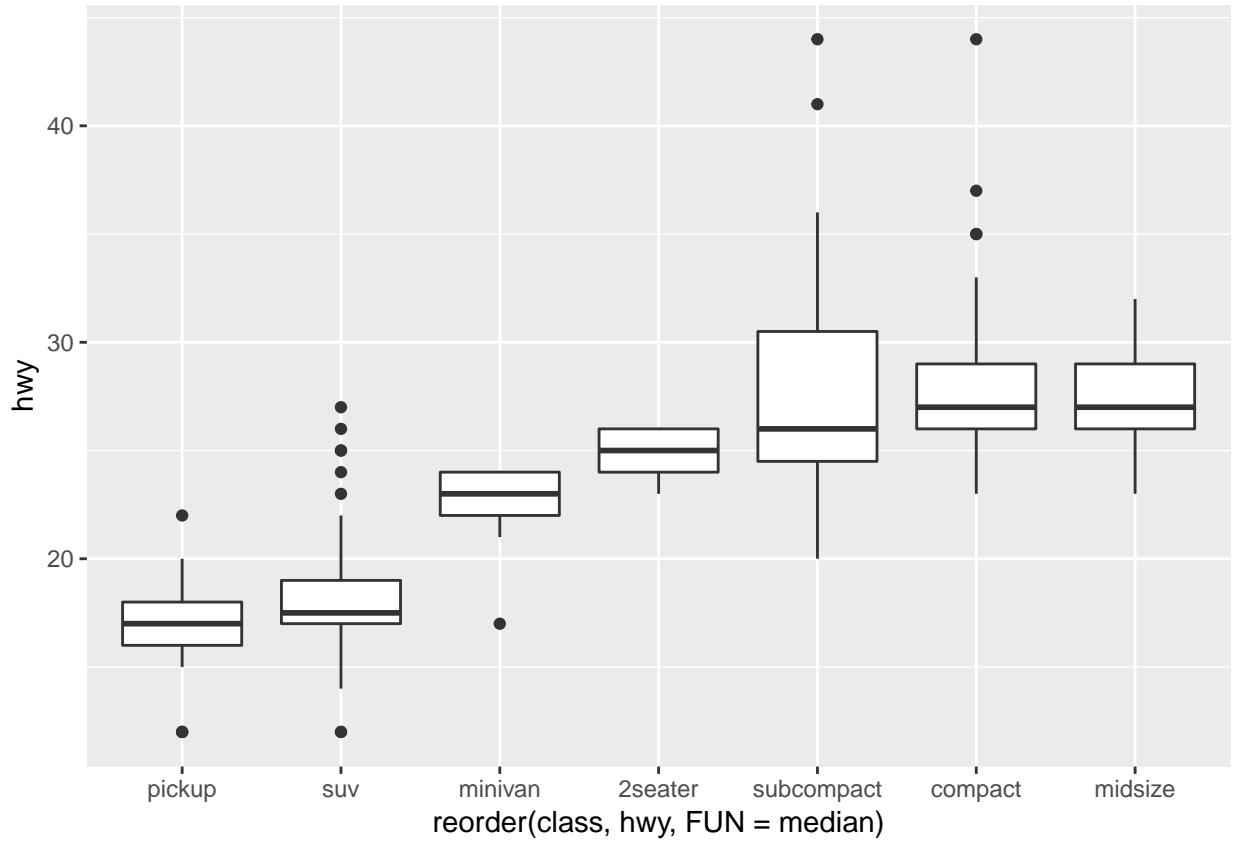
```
# Gerar boxplot das variáveis "cut" (eixo x) e "price" (eixo y) da base
# "diamonds" com "ggplot":
ggplot(data = diamonds, mapping = aes(x = cut, y = price)) +
  geom_boxplot()
```



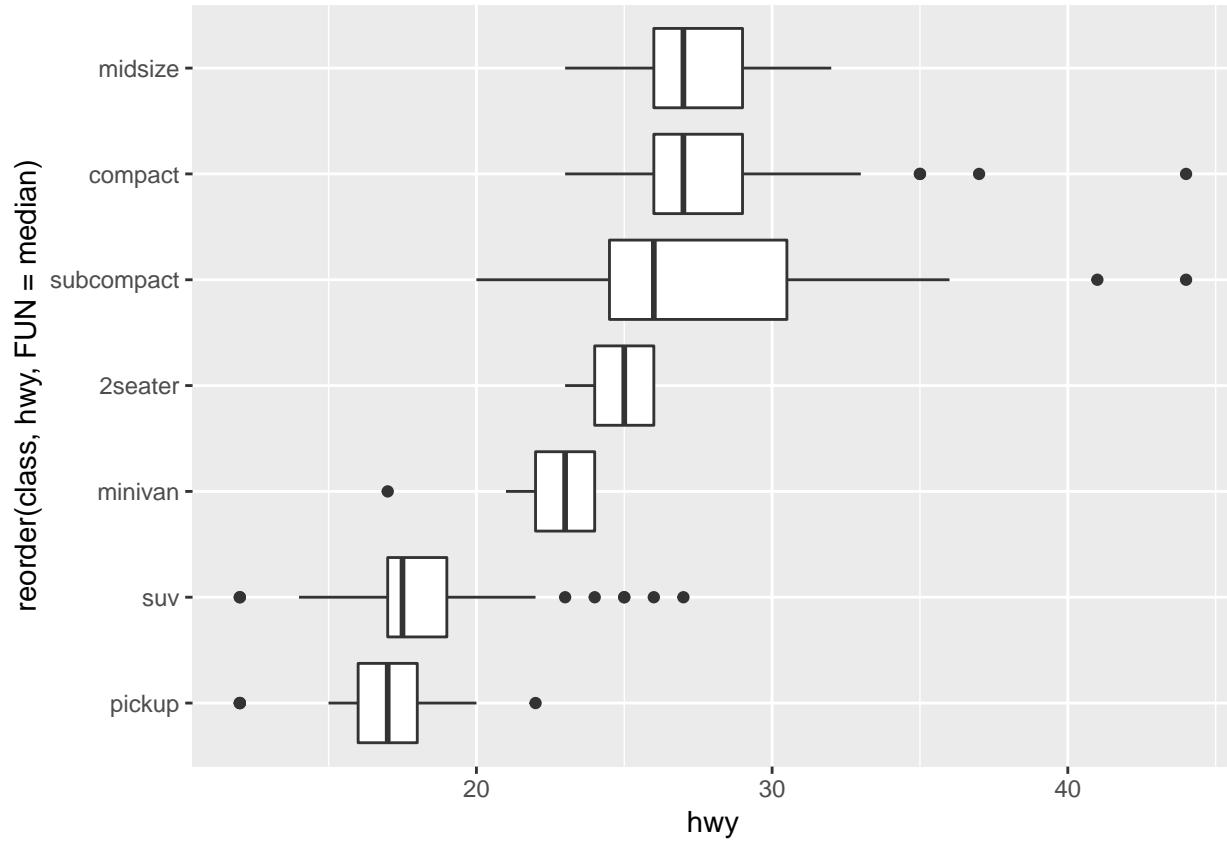
```
# Gerar boxplot das variáveis "class" (eixo x) e "hwy" (eixo y) da base
# "mpg" com "ggplot":
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot()
```



```
# Ordenar os níveis da variável "class" (eixo x) no boxplot de acordo com  
# a mediana da variável "hwy" (eixo y):  
ggplot(data = mpg) +  
  geom_boxplot(mapping = aes(x = reorder(class, hwy, FUN = median), y = hwy))
```

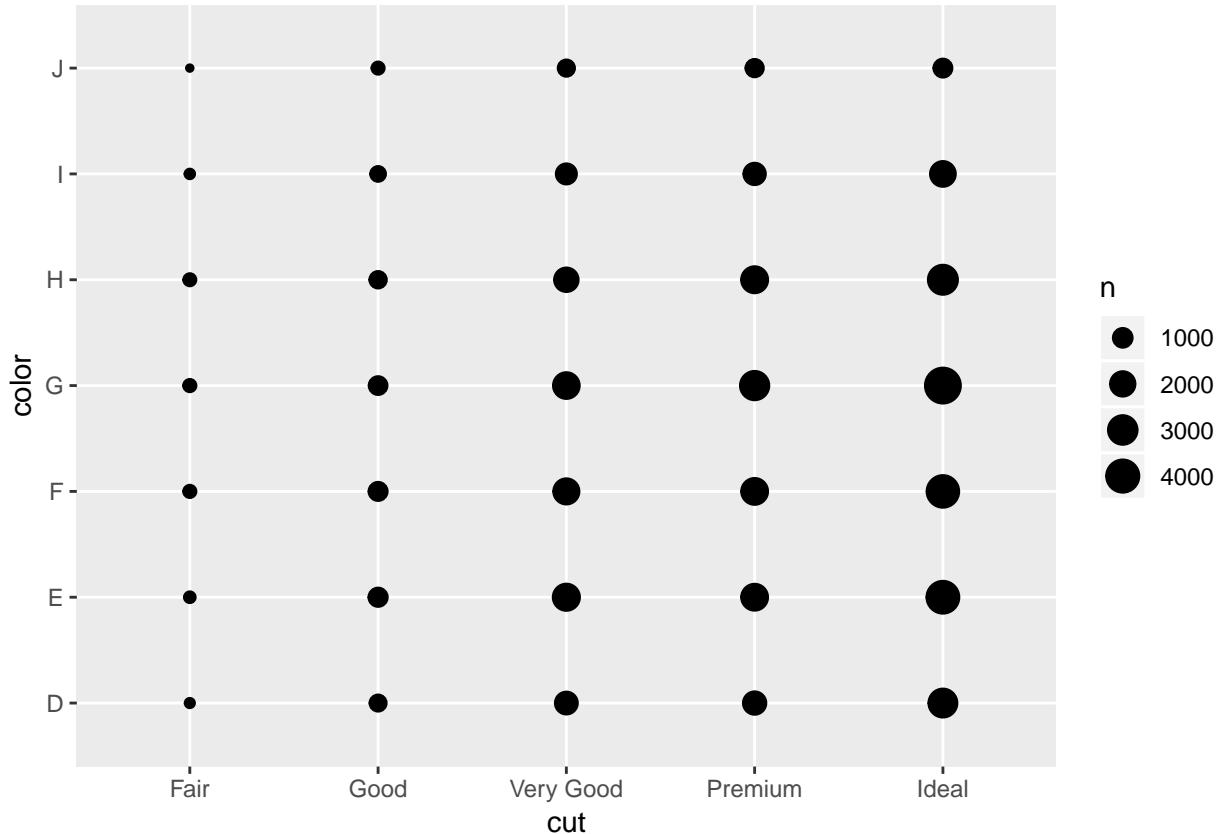


```
# Gerar boxplot horizontal:  
ggplot(data = mpg) +  
  geom_boxplot(mapping = aes(x = reorder(class, hwy, FUN = median), y = hwy)) +  
  coord_flip()
```



Exemplo 7.5.2:

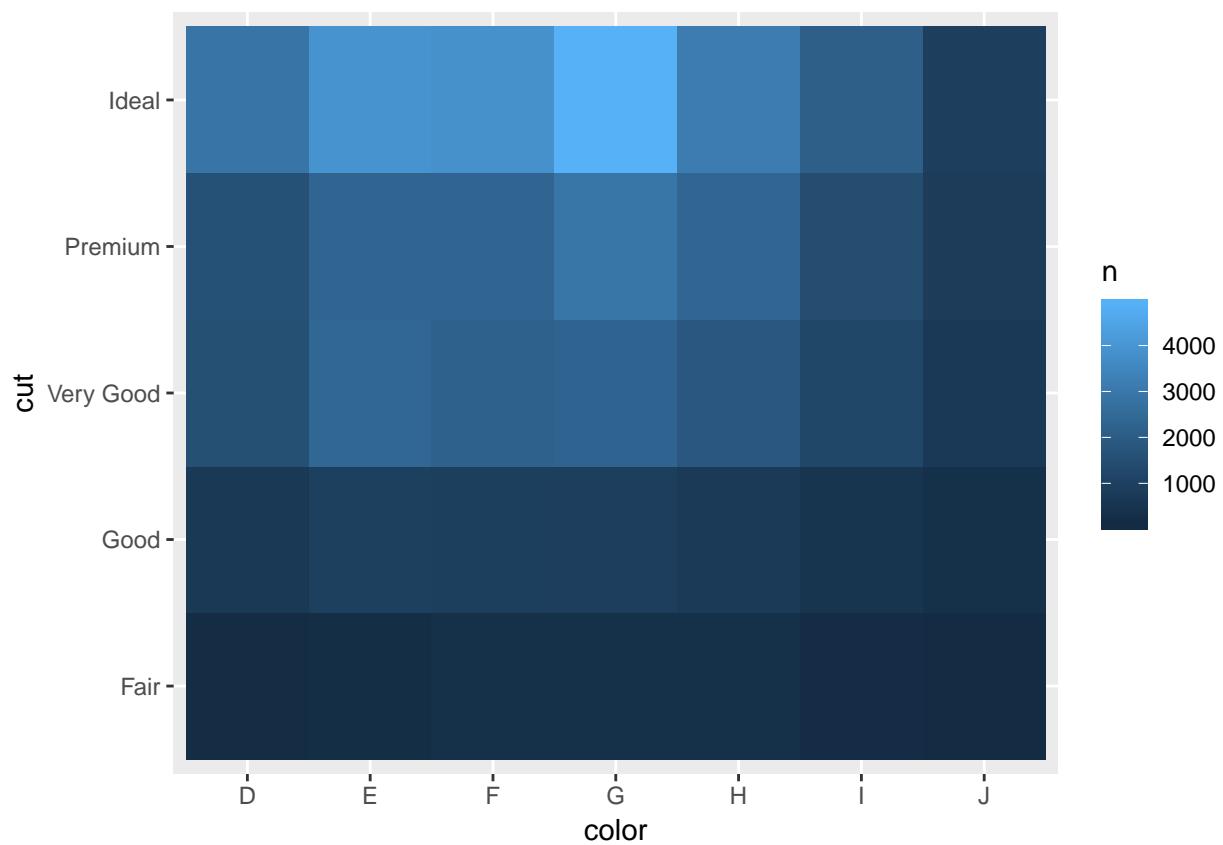
```
# Plotar frequência das combinações entre as variáveis categóricas "cut"
# e "color" de "diamonds":
ggplot(data = diamonds) +
  geom_count(mapping = aes(x = cut, y = color))
```



```
# Gerar tabela para analisar a frequência das combinações entre "color"
# e "cut":
diamonds %>%
  count(color, cut)
```

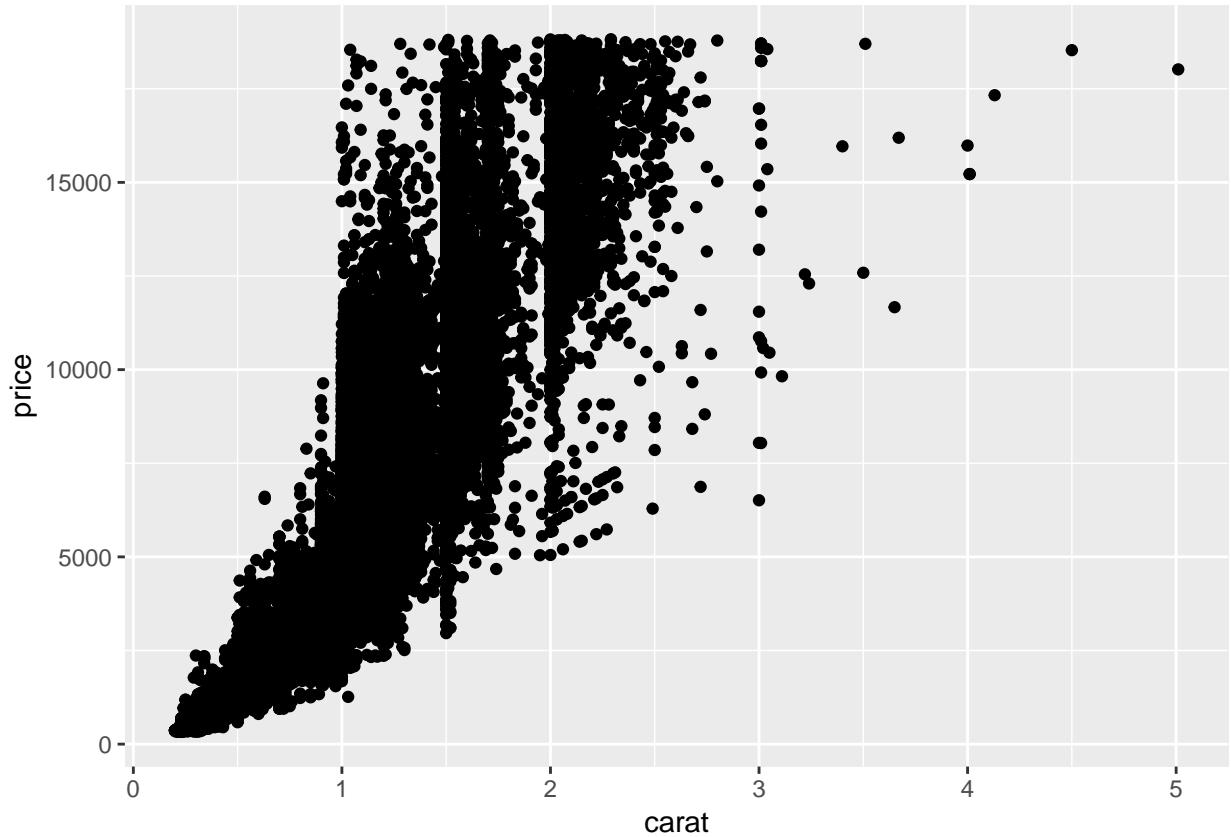
```
## # A tibble: 35 x 3
##   color   cut     n
##   <ord> <ord> <int>
## 1 D      Fair    163
## 2 D      Good    662
## 3 D      Very Good 1513
## 4 D      Premium 1603
## 5 D      Ideal   2834
## 6 E      Fair    224
## 7 E      Good    933
## 8 E      Very Good 2400
## 9 E      Premium 2337
## 10 E     Ideal   3903
## # ... with 25 more rows
```

```
# Utilizar função "geom_tile" do "ggplot" para visualizar combinações entre
# "color" e "cut":
diamonds %>%
  count(color, cut) %>%
  ggplot(mapping = aes(x = color, y = cut)) +
  geom_tile(mapping = aes(fill = n))
```

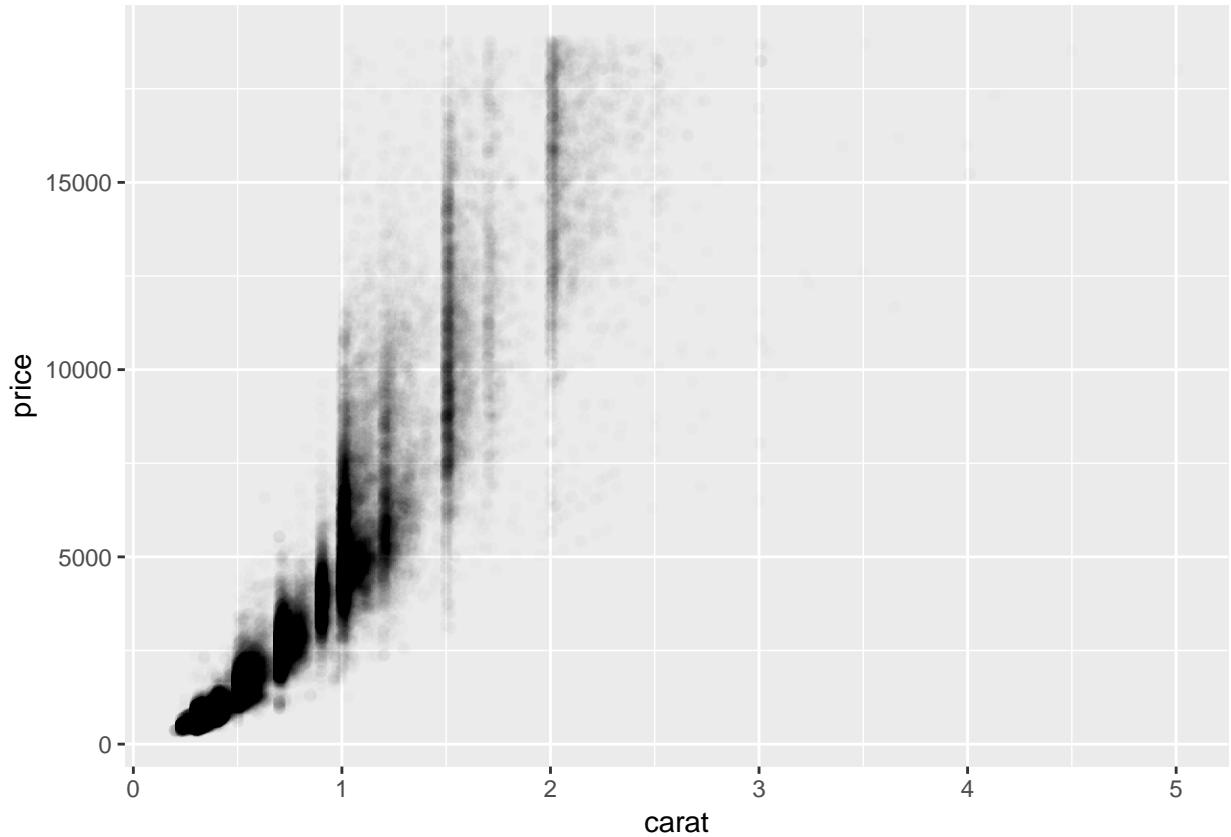


Exemplo 7.5.3:

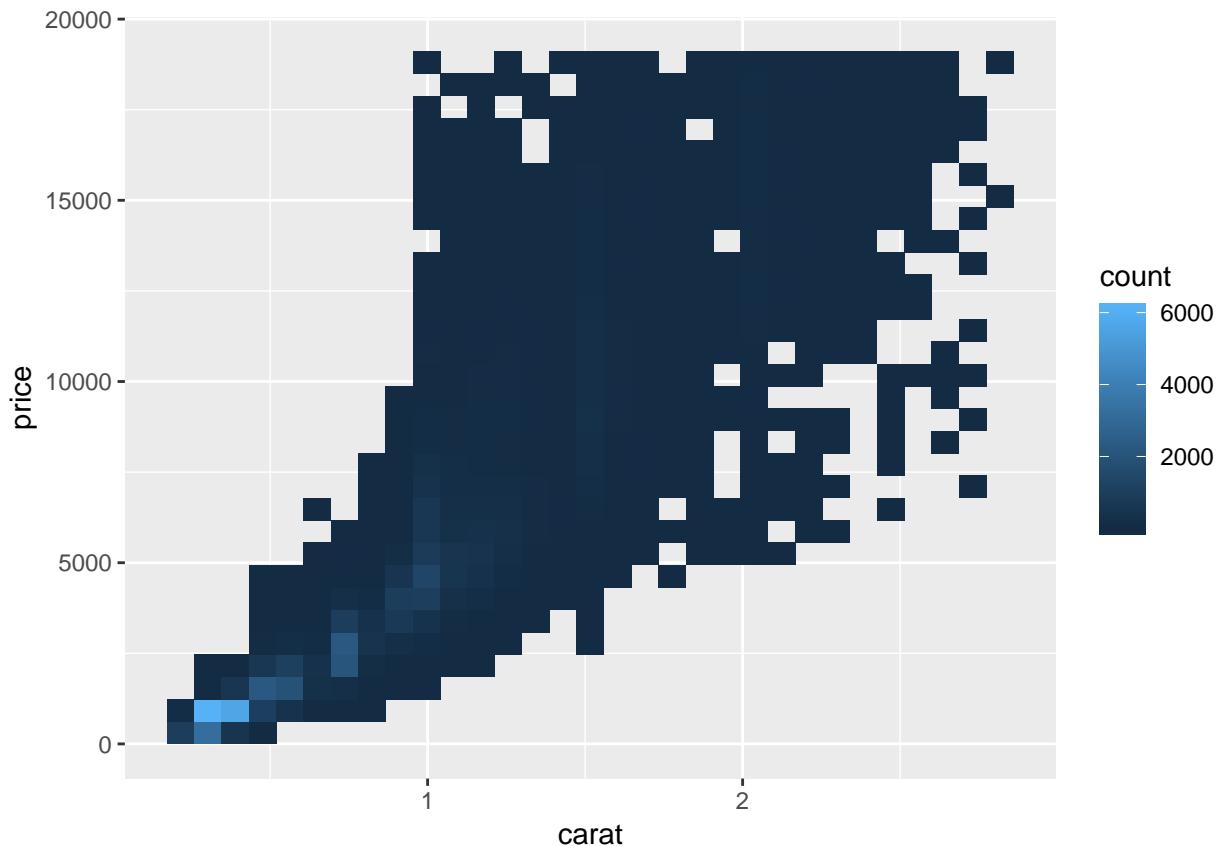
```
# Gerar scatterplot das variáveis contínuas "carat" e "price" da
# base "diamonds":
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price))
```



```
# Adicionar transparência ao scatterplot:  
ggplot(data = diamonds) +  
  geom_point(mapping = aes(x = carat, y = price), alpha = 1 / 100)
```

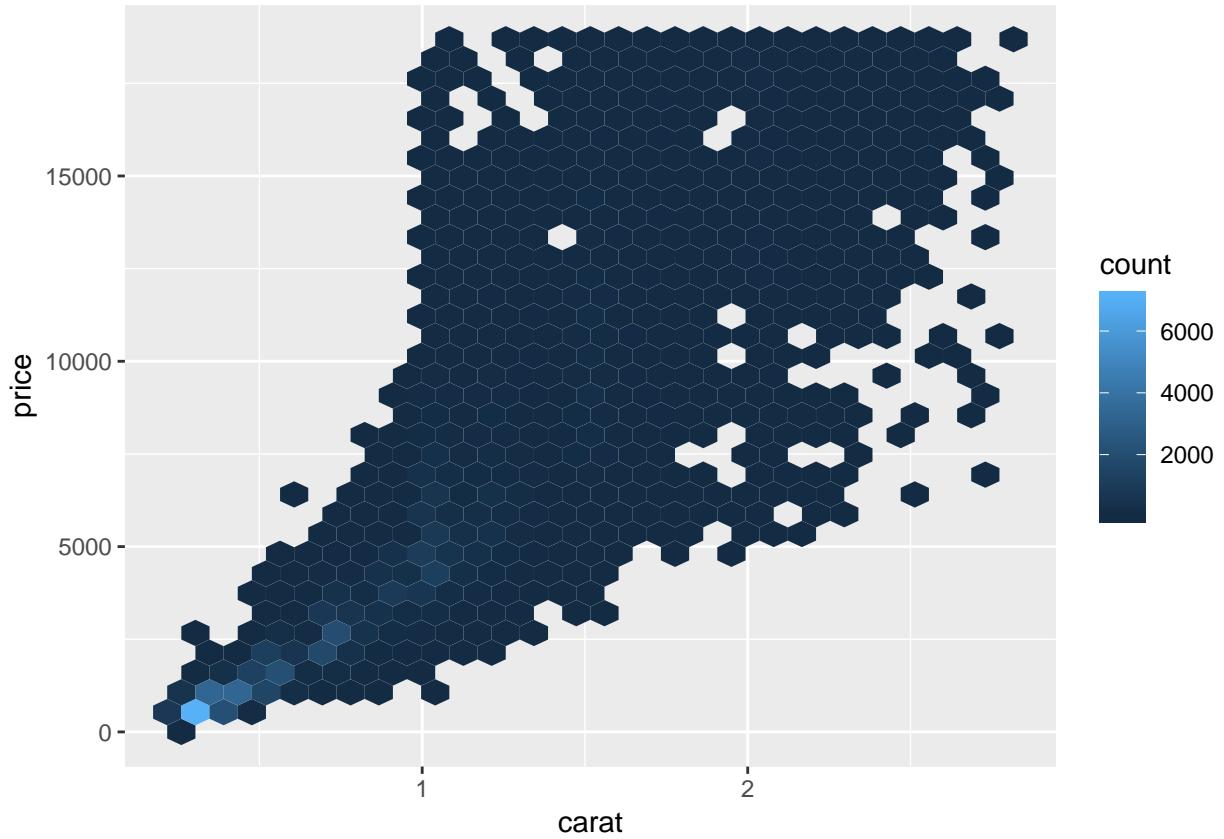


```
# Gerar scatterplot em duas dimensões:  
ggplot(data = smaller) +  
  geom_bin2d(mapping = aes(x = carat, y = price))
```

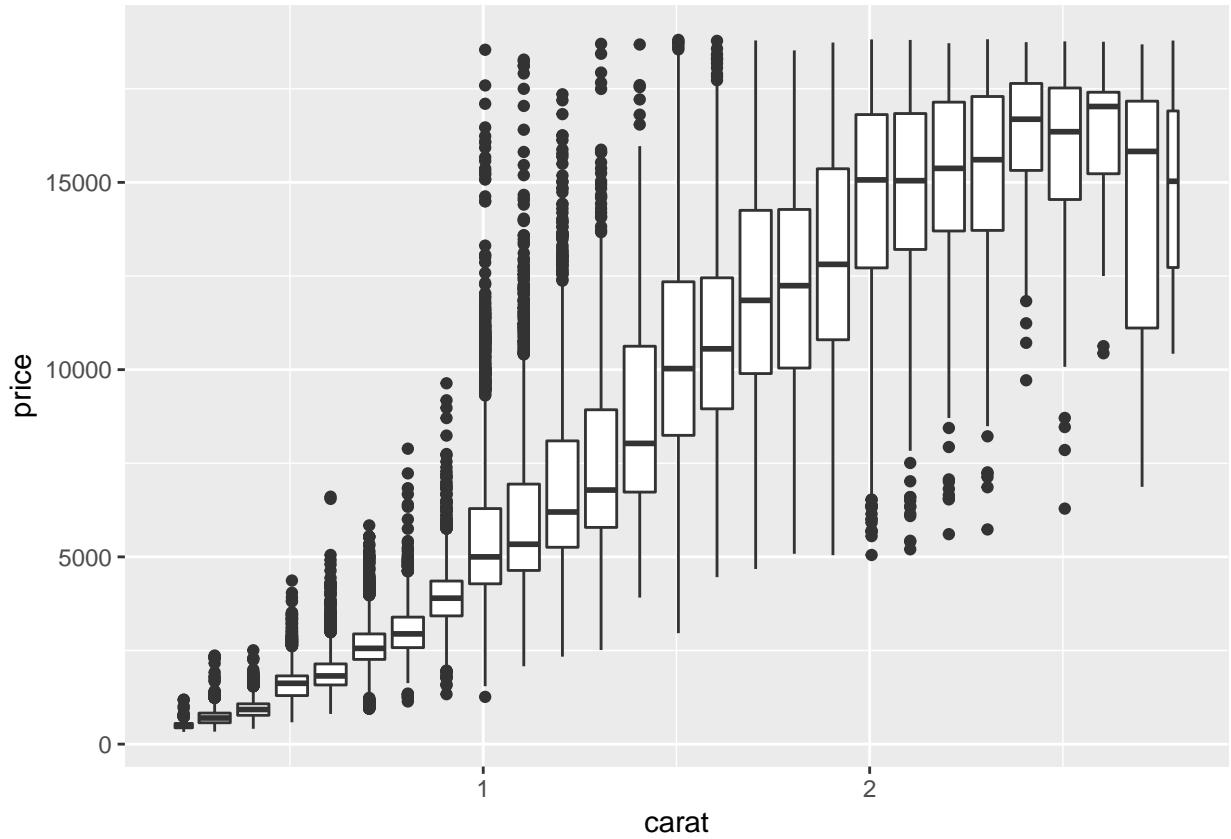


```
# Instalar pacote "hexbin" e gerar scatterplot com "pontos" na forma
# de hexágonos:
#install.packages("hexbin")
ggplot(data = smaller) +
  geom_hex(mapping = aes(x = carat, y = price))
```

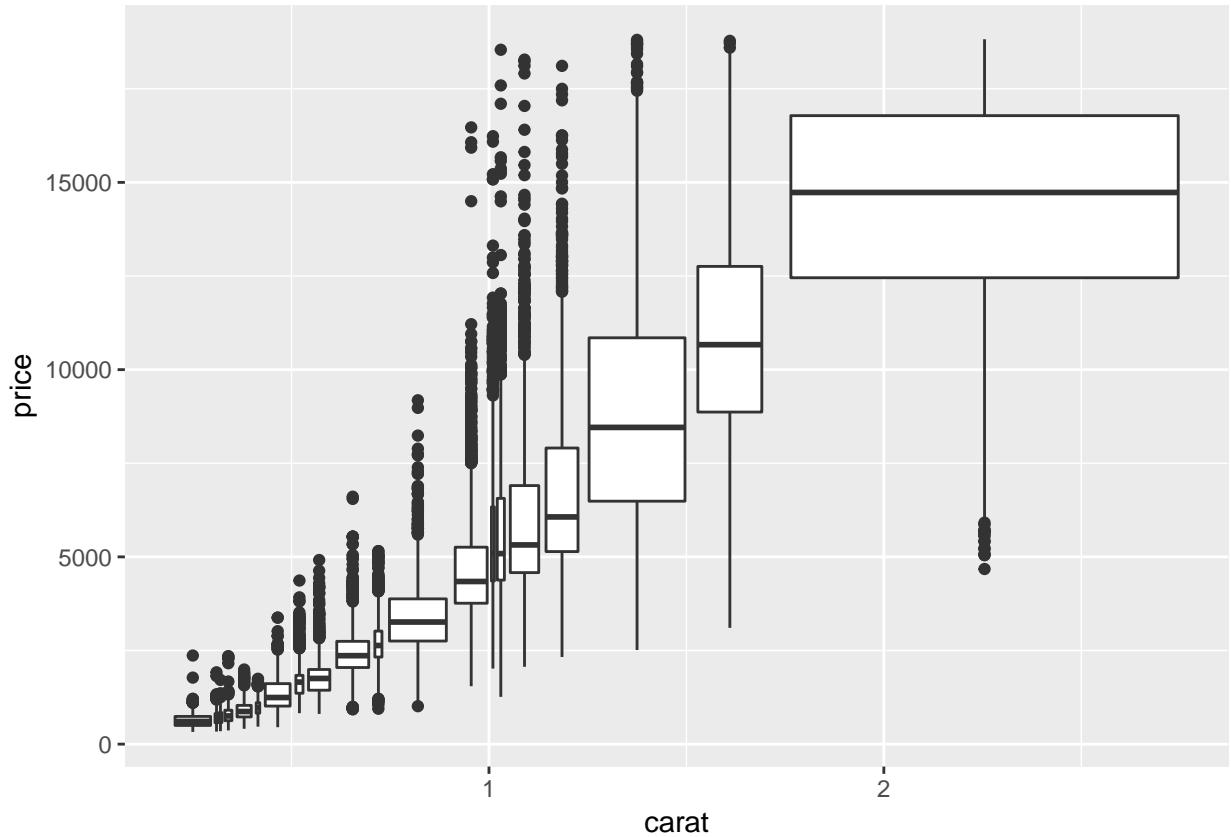
```
## Warning: package 'hexbin' was built under R version 3.5.3
```



```
# Gerar boxplot das variáveis "carat" (criar diferentes grupos, ou caixas,  
# para cada aumento de 0.1 no quilate) e "price" da base "diamonds":  
ggplot(data = smaller, mapping = aes(x = carat, y = price)) +  
  geom_boxplot(mapping = aes(group = cut_width(carat, 0.1)))
```

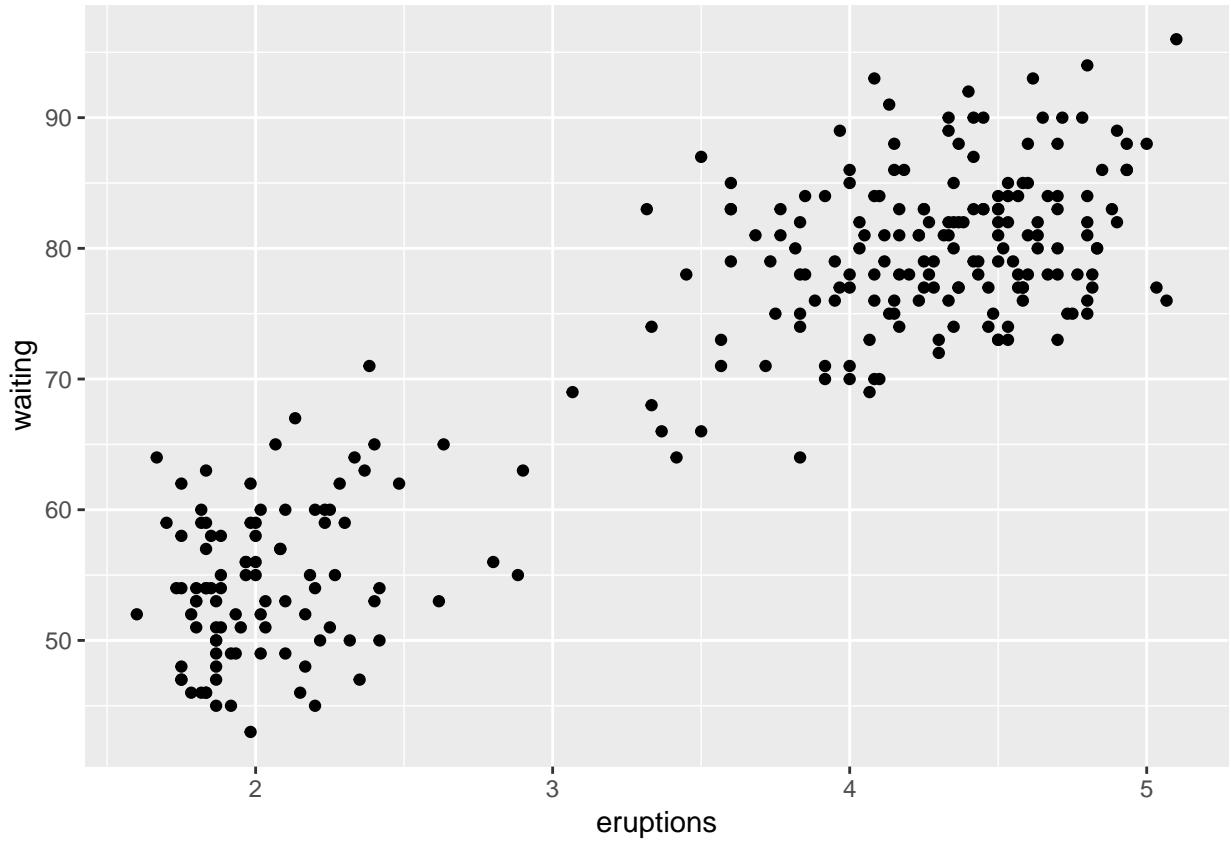


```
# Gerar boxplot das variáveis "carat" (criar diferentes grupos, ou caixas,
# para cada 20 observações) e "price" da base "diamonds":
ggplot(data = smaller, mapping = aes(x = carat, y = price)) +
  geom_boxplot(mapping = aes(group = cut_number(carat, 20)))
```



Exemplo 7.6:

```
# Gerar scatterplot com "ggplot" para a base "faithful", eixo X com a  
# variável "eruptions" e y "waiting":  
ggplot(data = faithful) +  
  geom_point(mapping = aes(x = eruptions, y = waiting))
```



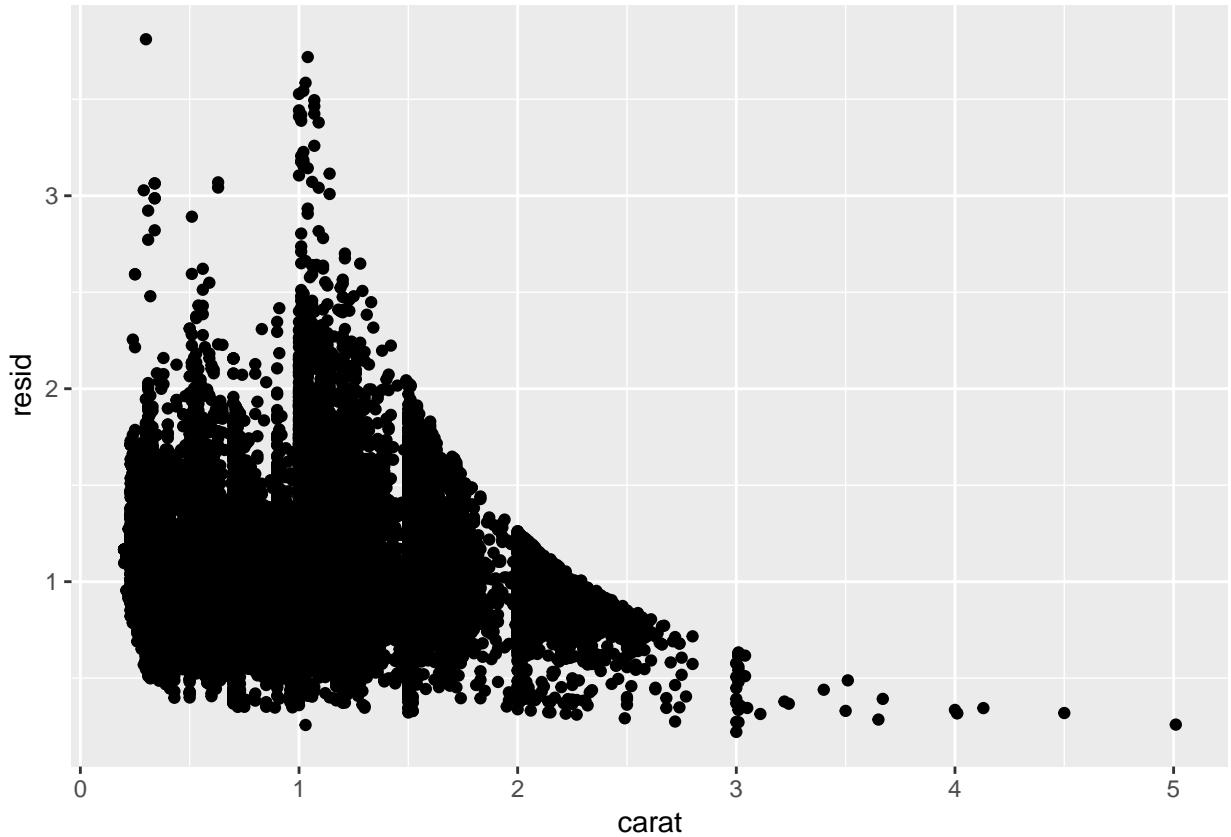
```
# Gerar modelo que prevê preço do quilate e calcular os resíduos com o
# pacote "modelr", adicioná-los à base "diamonds2" plotar resultado
# com "ggplot":
library(modelr)
```

```
## Warning: package 'modelr' was built under R version 3.5.3
```

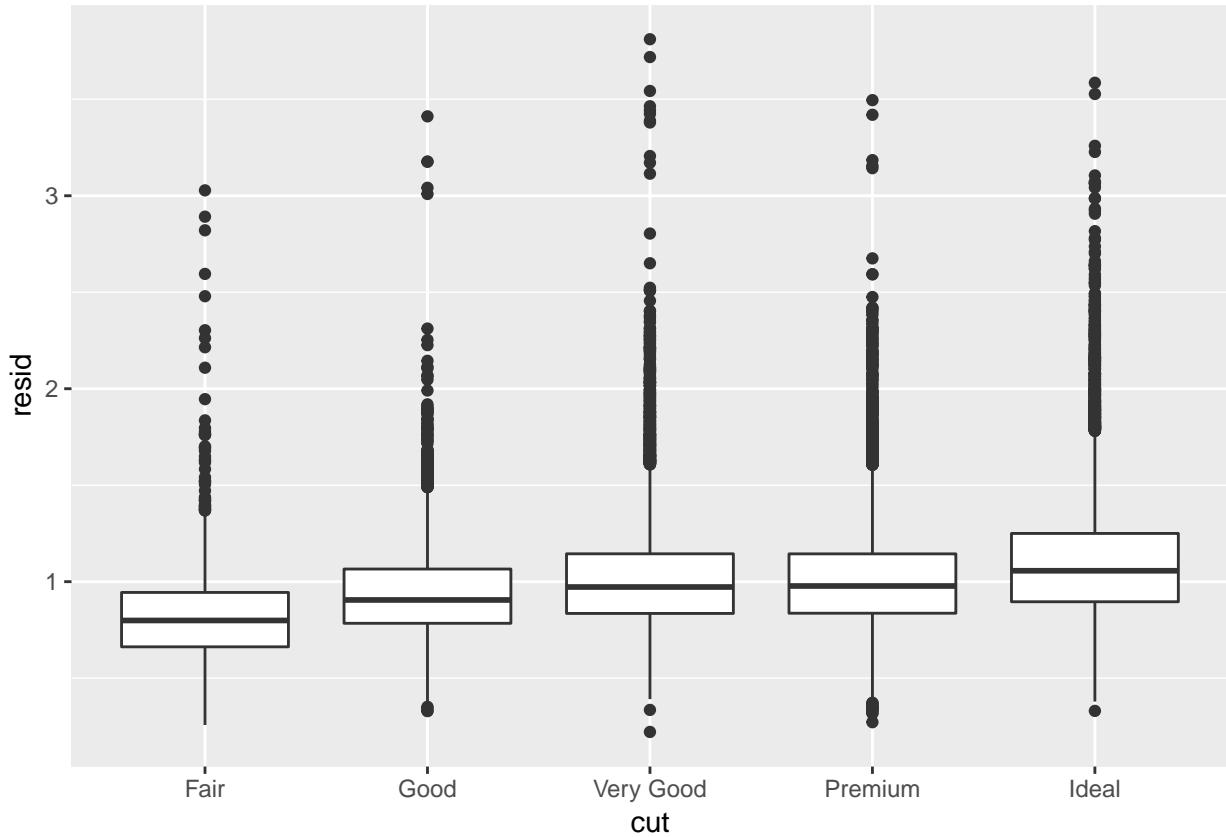
```
mod <- lm(log(price) ~ log(carat), data = diamonds)

diamonds2 <- diamonds %>%
  add_residuals(mod) %>%
  mutate(resid = exp(resid))

ggplot(data = diamonds2) +
  geom_point(mapping = aes(x = carat, y = resid))
```

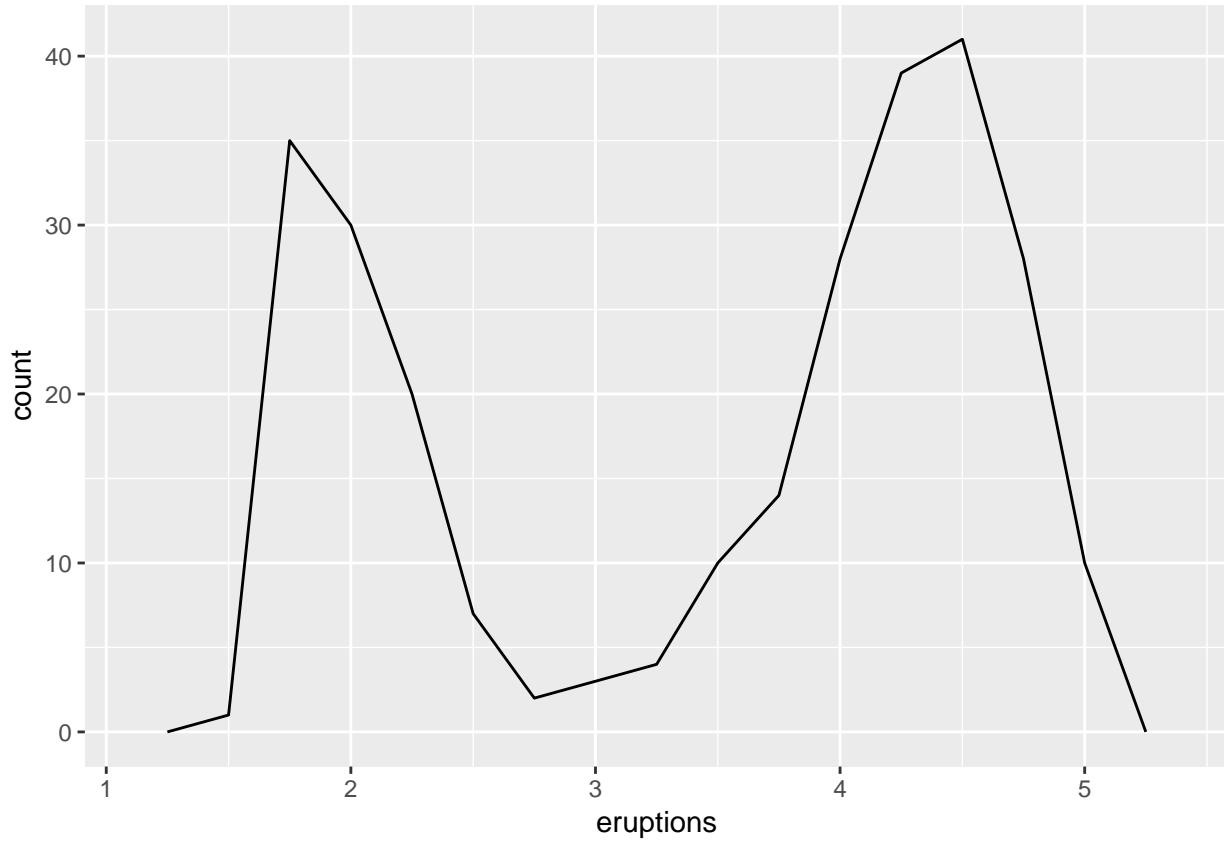


```
# Gerar boxplot com "ggplot", com a nova coluna "resid" no eixo y
# e "cut" no x:
ggplot(data = diamonds2) +
  geom_boxplot(mapping = aes(x = cut, y = resid))
```

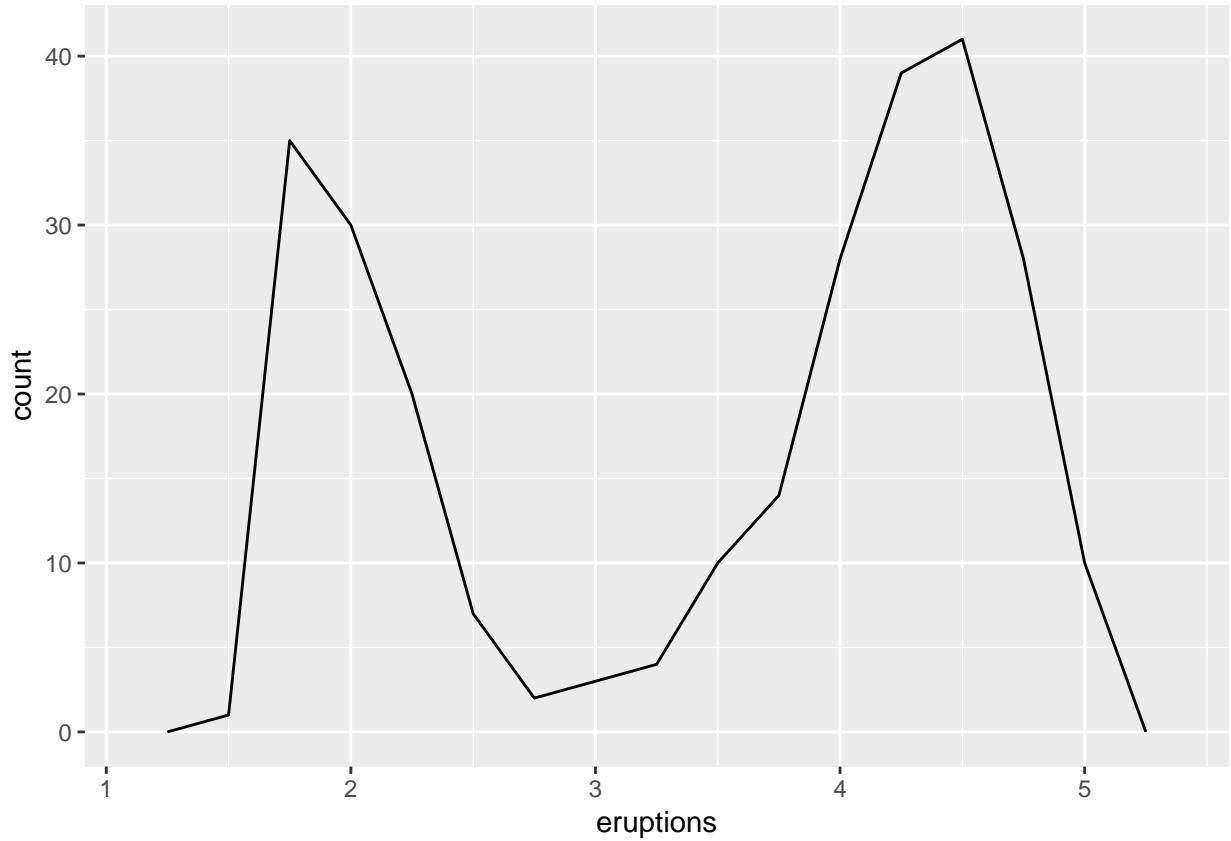


Exemplo 7.7:

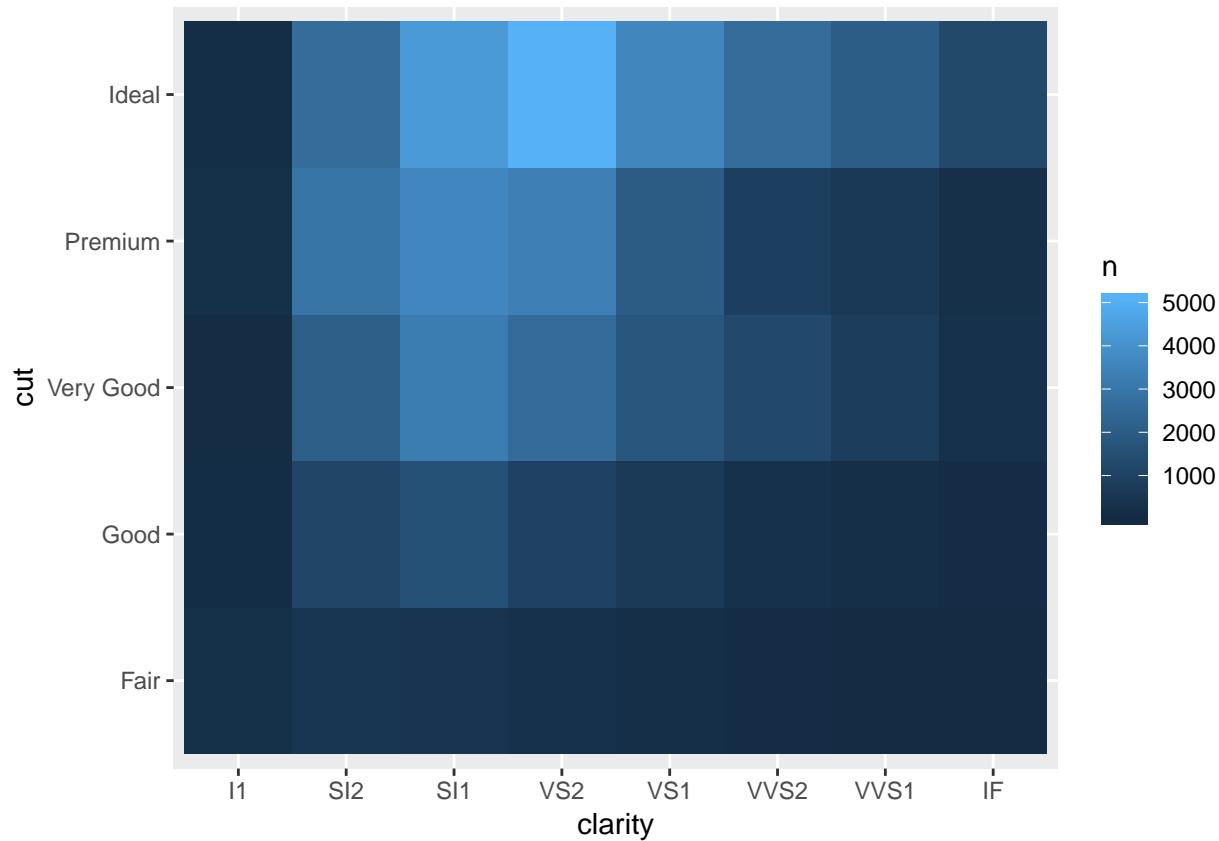
```
# Gerar gráfico de linhas com a frequência da variável "eruptions" da base
# "faithful" com "ggplot":
ggplot(data = faithful, mapping = aes(x = eruptions)) +
  geom_freqpoly(binwidth = 0.25)
```



```
# Repetir ação anterior, de forma mais concisa:  
ggplot(faithful, aes(eruptions)) +  
  geom_freqpoly(binwidth = 0.25)
```



```
# Utilizar função "geom_tile" do "ggplot" para visualizar combinações entre
# "clarity" e "cut" da base "diamonds":
diamonds %>%
  count(cut, clarity) %>%
  ggplot(aes(clarity, cut, fill = n)) +
  geom_tile()
```



Questão 10:

```
# Carregar base de dados:  
setwd("C:/Users/Duda/Desktop/PPGCP/Análise de Dados/lista_06")  
load("vote_growth_usa.RData")  
  
# Análise de regressão dos votos (VD) e crescimento econômico (VI) nos EUA:  
reg <- lm(Vote ~ Growth, data = bd)  
summary(reg)
```

```

## 
## Call:
## lm(formula = Vote ~ Growth, data = bd)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8.1968 -3.7667 -0.7972  3.1294 10.0107
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```

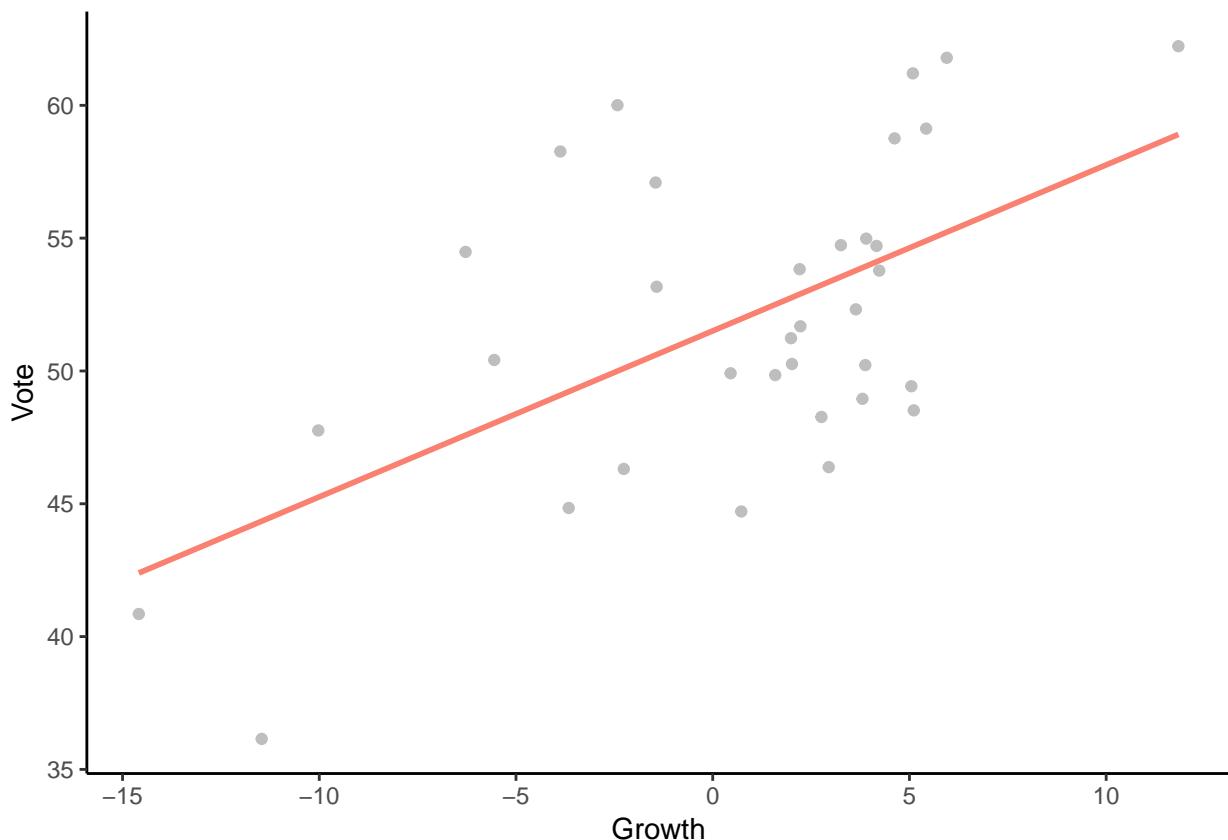
## (Intercept) 51.5082      0.8569   60.110 < 2e-16 ***
## Growth       0.6249      0.1577    3.962  0.00039 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 4.955 on 32 degrees of freedom
## Multiple R-squared:  0.3291, Adjusted R-squared:  0.3081
## F-statistic: 15.7 on 1 and 32 DF,  p-value: 0.0003898

```

```

# Plotar dados e reta de regressão sem IC:
ggplot(data = bd, aes(y = Vote, x = Growth)) +
  geom_point(color = "grey") +
  theme_classic() +
  geom_smooth(method="lm", color = "salmon", se = FALSE)

```



Questão 11:

```

# Filtrar base para o período de 1876 a 1932 (as primeiras 15 observações):
bd_selec <- bd[1:15,]
reg_selec <- reg<-lm(Vote ~ Growth, data = bd_selec)
summary(reg_selec)

```

```

## 
## Call:
## lm(formula = Vote ~ Growth, data = bd_selec)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -9.7209 -3.5931  0.5013  3.1253  9.3127 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 51.9850    1.5371 33.821 4.66e-14 ***
## Growth      0.5336    0.2366  2.255   0.042 *  
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 5.638 on 13 degrees of freedom
## Multiple R-squared:  0.2811, Adjusted R-squared:  0.2258 
## F-statistic: 5.083 on 1 and 13 DF,  p-value: 0.04205

```

letra a)

Modelo completo: No modelo completo, como existem mais observações (34), os resultados apresentaram menor p-valor do que o modelo com as 15 observações. Tanto o resultado do intercepto (^a) quanto o do coeficiente de variação (^b), apresentaram p-valor < 0.001.

Modelo reduzido: O resultado do intercepto (^a) apresentou p-valor < 0.001, enquanto o resultado do coeficiente de variação (^b) apresentou p-valor < 0.05 (0.04205).

Conclusão: Como esperado, devido ao maior número de observações, o modelo completo apresentou resultados mais significantes que o modelo reduzido.

letra b)

Modelo completo: Neste modelo o valor de $\hat{b} = 0.6249$ e o erro = 0.1577, portanto, o IC de 95% para \hat{b} é igual a:

$$0.6249 + 2 \times (0.1577)$$

```
## [1] 0.9403
```

$$0.6249 - 2 \times (0.1577)$$

```
## [1] 0.3095
```

Modelo reduzido: Neste modelo o valor de $\hat{b} = 0.5336$. e o erro = 0.2366, portanto, o IC de 95% para \hat{b} é igual a:

```
0.5336 + 2*(0.2366)
```

```
## [1] 1.0068
```

```
0.5336 - 2*(0.2366)
```

```
## [1] 0.0604
```

Conclusão: Os intervalos de confiança de \hat{b} para o modelo completo são mais estreitos, visto que o desvio padrão apresentado é menor do que aquele observado no modelo reduzido.

letra c)

Modelo completo: $R^2 = 0.3291$, RMSE: 4.955

Modelo reduzido: $R^2 = 0.2811$, RMSE: 5.638

Conclusão: No modelo completo, a taxa de crescimento da economia (VI) explica aproximadamente 0.33 da variação dos votos no partido incumbente (VD); enquanto para o modelo reduzido, 0.28 dessa variação na VD é explicada pela VI. O modelo completo também apresentou menor erro médio quadrático da diferença entre o valor apresentado e o valor estimado (4.9 versus 5.6), ou seja, a diferença entre Y_i e \hat{Y}_i (que é igual a \hat{u}) é menor no modelo completo do que no reduzido.