Métodos Numéricos Aplicados a Finanças — Turma 2025

Aula 02

Prof. Frega
PPGOLD/Universidade Federal do Paraná

18/03/2025

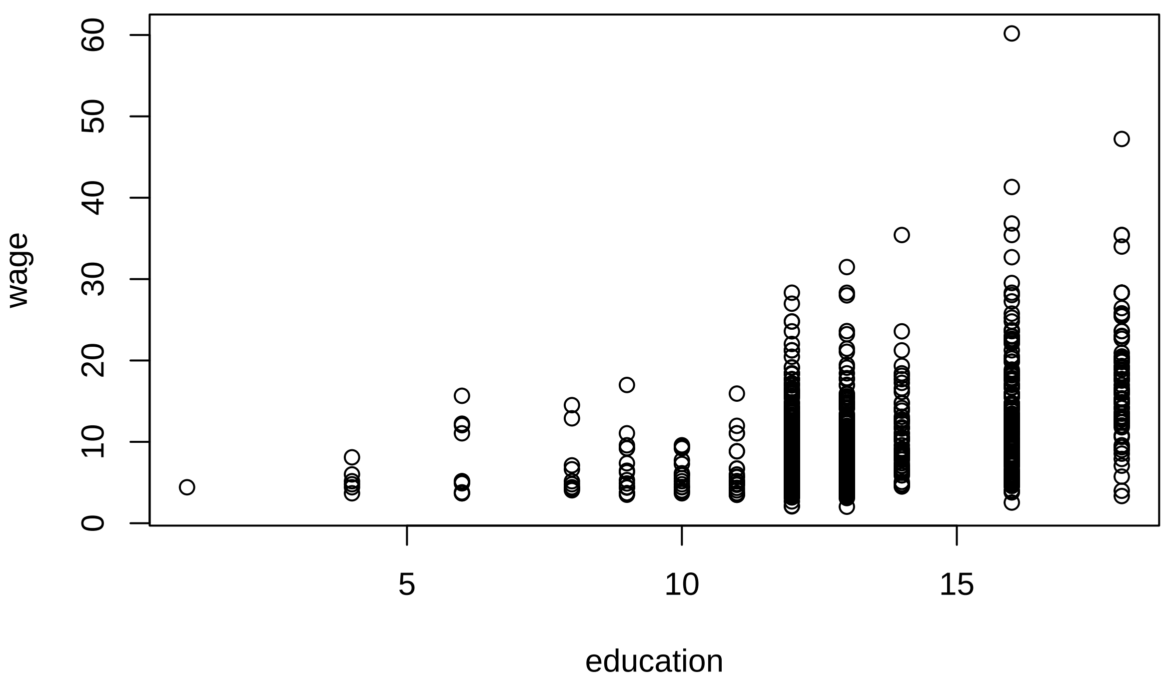
Sumário

Capítulo 1

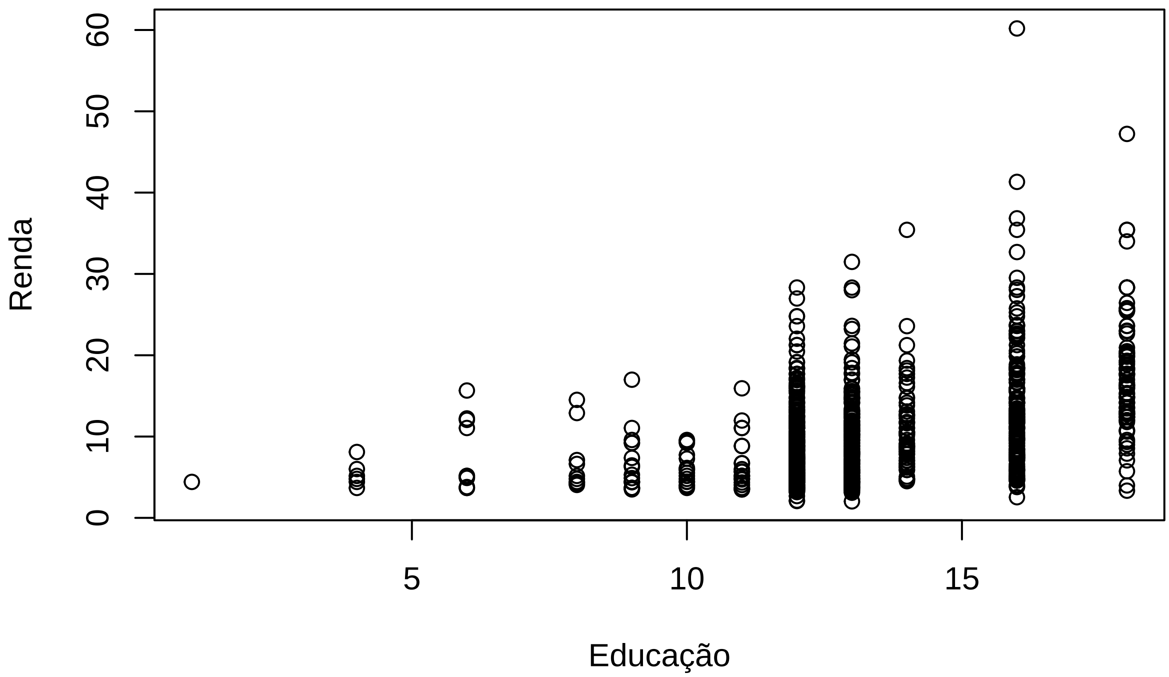
Regressão linear simples

1.1

```r
library(PoEdata)
data("cps_small")
plot(cps_small$educ, cps_small$wage, xlab = "education", ylab = "wage")
```



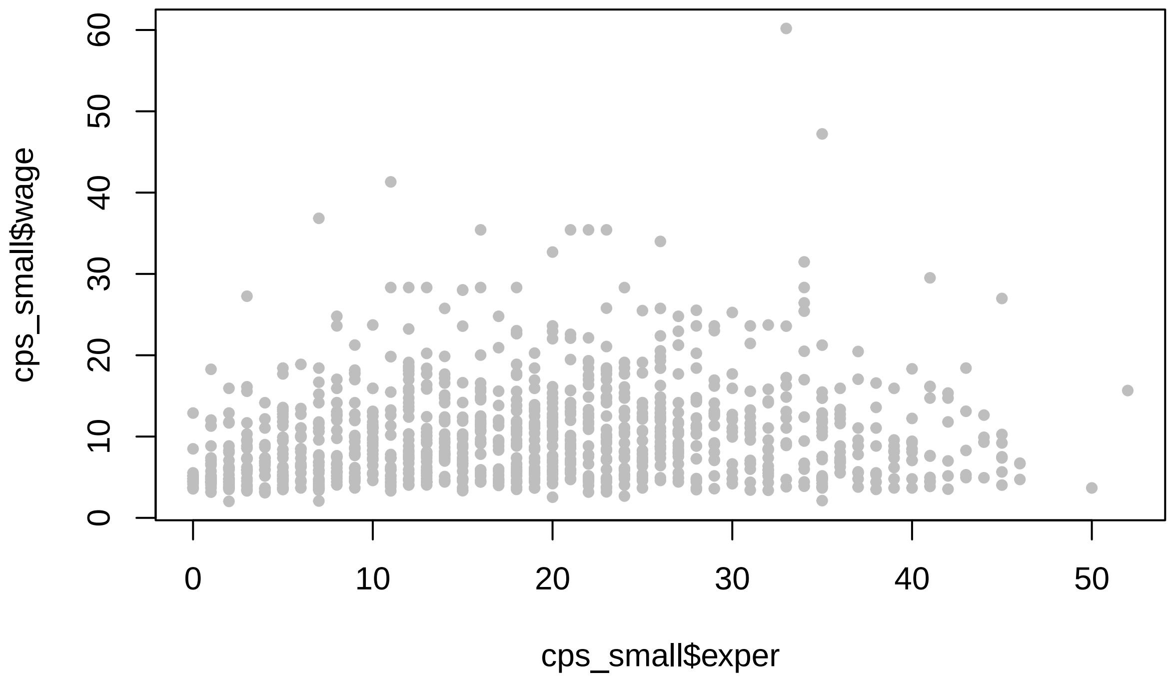

```r
plot(cps_small$educ, cps_small$wage, xlab = "Educação", ylab = "Renda")
```




```r
head(cps_small, 15)
```

```
##    wage educ exper female black white midwest south west
## 1  2.03   13     2      1     0     1       0     1    0
## 2  2.07   12     7      0     0     1       1     0    0
## 3  2.12   12    35      0     0     1       0     1    0
## 4  2.54   16    20      1     0     1       0     1    0
## 5  2.68   12    24      1     0     1       0     1    0
## 6  3.09   13     4      0     0     1       0     1    0
## 7  3.16   13     1      0     0     1       0     0    1
## 8  3.17   12    22      1     0     1       0     1    0
## 9  3.20   12    23      0     0     1       0     1    0
## 10 3.27   12     4      1     0     1       0     0    1
## 11 3.32   12    11      1     0     1       0     0    1
## 12 3.32   13     3      1     0     1       1     0    0
## 13 3.34   18    15      0     0     1       1     0    0
## 14 3.39   13     7      1     0     1       0     0    0
## 15 3.39   12    15      1     0     1       0     0    1
```

```r
plot(cps_small$exper, cps_small$wage, col = "gray", pch = 20)
```

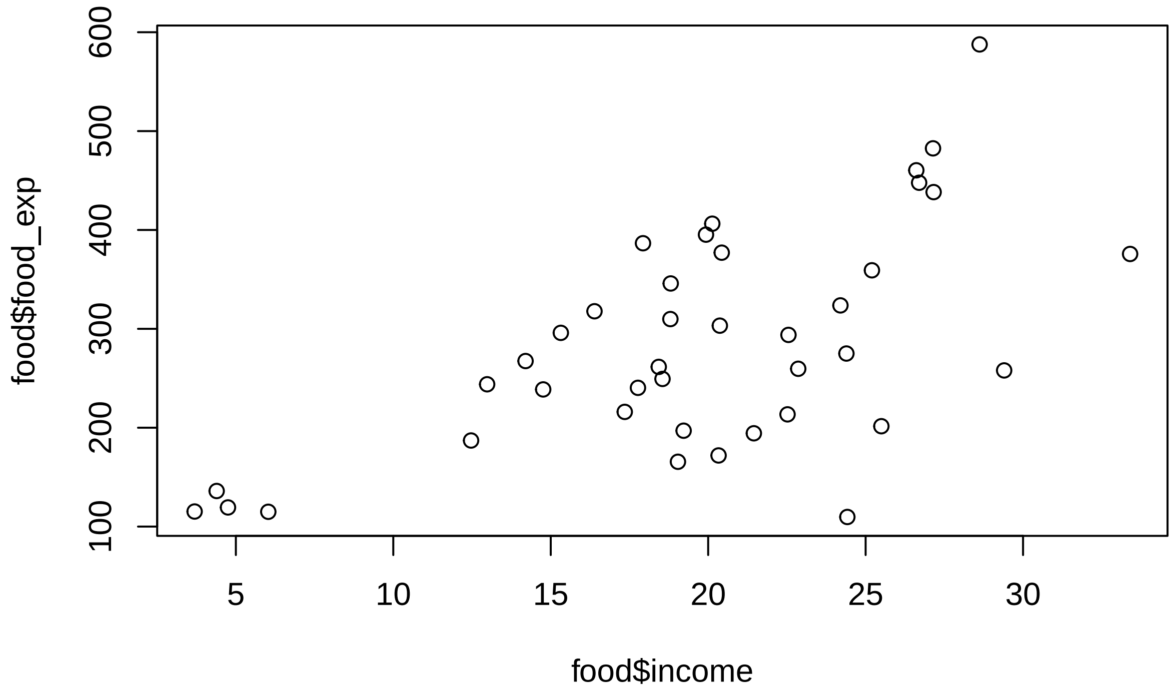



```r
library(PoEdata)
data(food)
head(food)
```

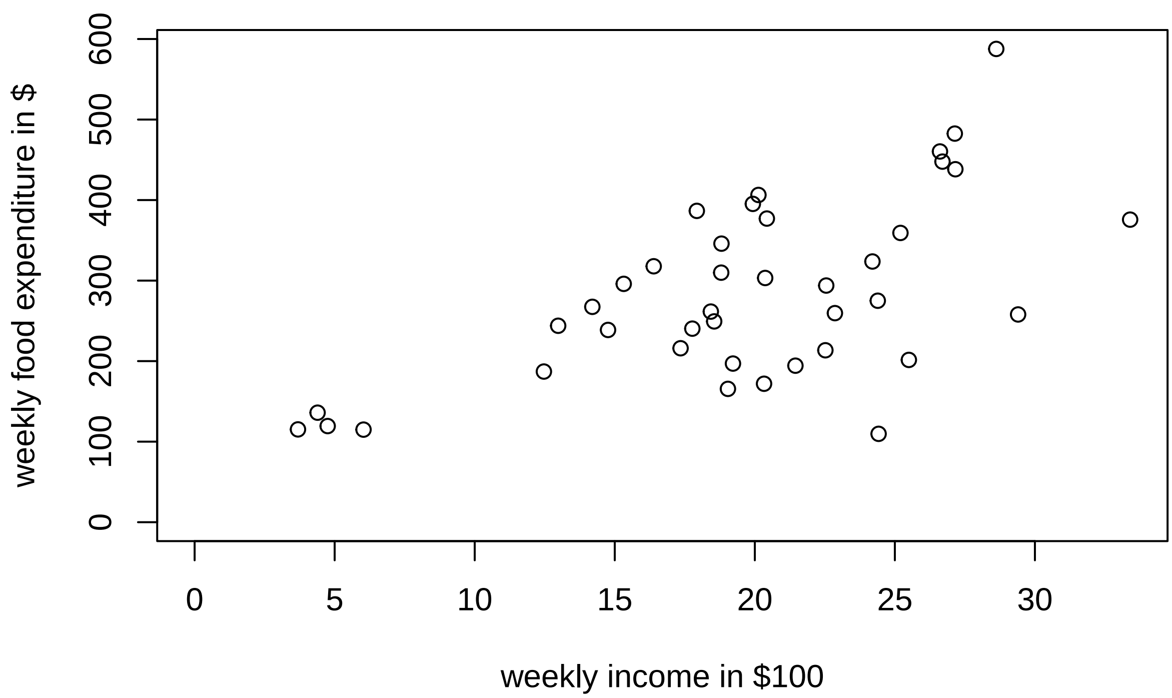
```
##   food_exp income
## 1   115.22   3.69
## 2   135.98   4.39
## 3   119.34   4.75
## 4   114.96   6.03
## 5   187.05  12.47
## 6   243.92  12.98
```

```r
`?`(food)
```

```r
data("food", package = "PoEdata")
plot(food$income, food$food_exp)
```




```r
# Gráfico de dispersão ou scatter plot
plot(food$income, food$food_exp, ylim = c(0, max(food$food_exp)),
    xlim = c(0, max(food$income)), xlab = "weekly income in $100",
    ylab = "weekly food expenditure in $", type = "p")
```



1.3 Estimating a Linear Regression

$$food\_exp = \beta_0 + \beta_1 income + e\widehat{food\_exp} = \beta_0 + \beta_1 income$$

```r
library(PoEdata)
# roda a regressão
mod1 <- lm(formula = food_exp ~ income, data = food)
# olha os coeficientes
mod1$coefficients
```

```
## (Intercept)      income
##    83.41600    10.20964
```

```r
# ou
coef(mod1)
```

```
## (Intercept)      income
##    83.41600    10.20964
```

```r
# um por um
mod1$coefficients[1]
```

```
## (Intercept)
##      83.416
```

```r
mod1$coefficients[2]
```

```
##   income
## 10.20964
```

```r
# ou
(b1 <- coef(mod1)[[1]])
```

```
## [1] 83.416
```

```r
(b2 <- coef(mod1)[[2]])
```

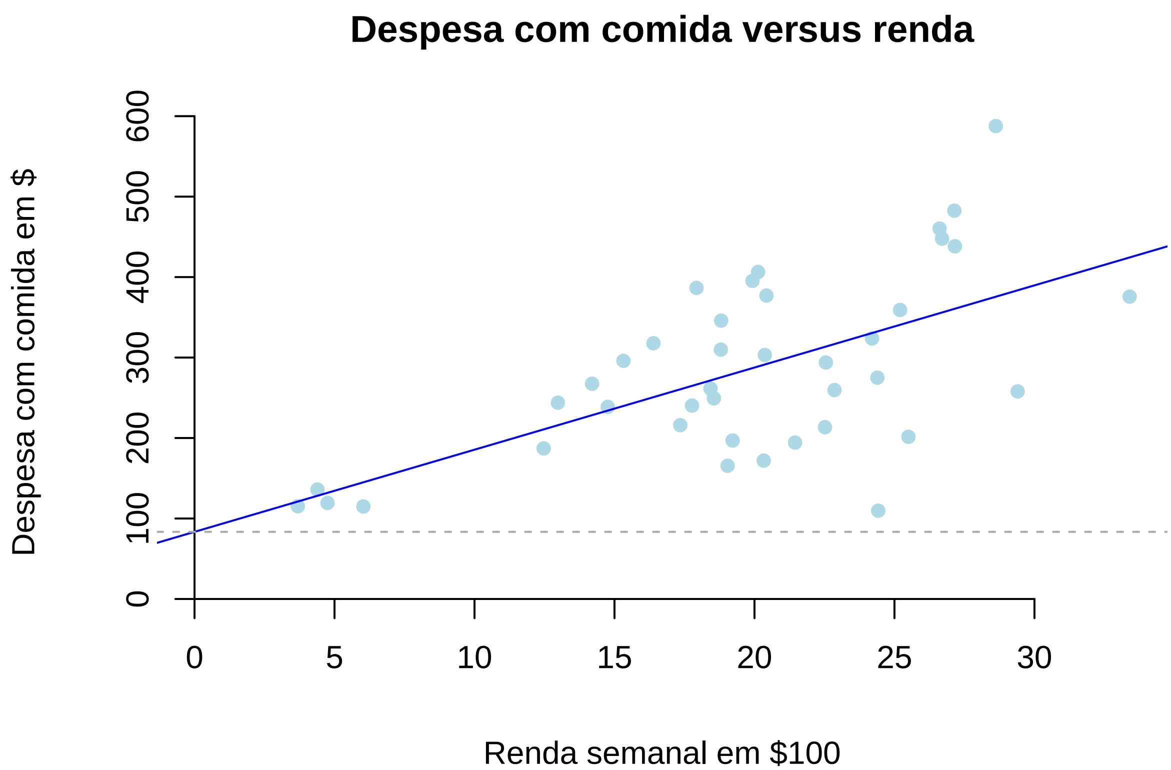
```
## [1] 10.20964
```

```r
# mostra o resultado da regressão
smod1 <- summary(mod1)
smod1
```

```
##
## Call:
## lm(formula = food_exp ~ income, data = food)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -223.025  -50.816   -6.324   67.879  212.044
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   83.416     43.410   1.922   0.0622 .
## income        10.210      2.093   4.877 1.95e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 89.52 on 38 degrees of freedom
## Multiple R-squared:  0.385,	Adjusted R-squared:  0.3688
## F-statistic: 23.79 on 1 and 38 DF,  p-value: 1.946e-05
```

```r
plot(food$income, food$food_exp, xlab = "Renda semanal em $100",
    ylab = "Despesa com comida em $", ylim = c(0, max(food$food_exp)),
```

```
    xlim = c(0, max(food$income)), type = "p", col = "lightblue",
    pch = 16, frame.plot = FALSE, axes = FALSE, main = "Despesa com comida versus renda")
axis(1, pos = 0)
axis(2, pos = 0)
# abline(b1,b2)
abline(mod1, col = "blue")
abline(h = b1, col = "darkgray", lty = 2)
```



1.4 Prediction with the Linear Regression Model

Qual a despesa com comida de um indivíduo que ganha $2000 por semana?

$$\widehat{food\_exp} = 83.416 + 10.210 \cdot income \widehat{food\_exp} = 83.416 + 10.210 \cdot 20$$

```
coef(mod1)[1] + coef(mod1)[2] * 20
```

```
## (Intercept)
##    287.6089
```

```
predict(mod1, data.frame(income = 20))
```

```
##        1
## 287.6089
```

1.5 Repeated Samples to Assess Regression Coefficients

Tecnicamente isso se chama *bootstrap* e trataremos depois.

1.6 Estimated Variances and Covariance of Regression Coefficients

Será útil mais tarde.

1.7 Non-Linear Relationships

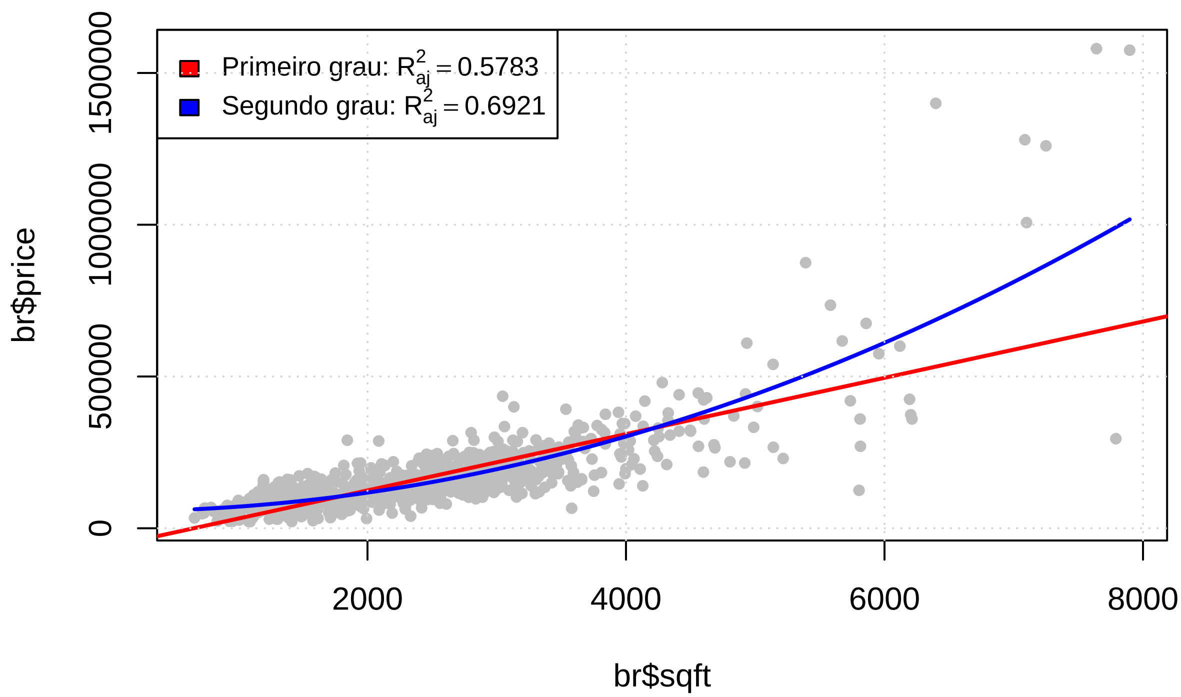
```r
library(PoEdata)
data(br)
# testando uma relação quadrática
mod3 <- lm(formula = price ~ I(sqft^2), data = br)
# versus uma do primeiro grau
mod3.a <- lm(formula = price ~ sqft, data = br)
(summary(mod3) -> s3)
```

```
##
## Call:
## lm(formula = price ~ I(sqft^2), data = br)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -696604  -23366     779   21869  713159
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.578e+04  2.890e+03   19.30   <2e-16 ***
## I(sqft^2)   1.542e-02  3.131e-04   49.25   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 68210 on 1078 degrees of freedom
## Multiple R-squared:  0.6923, Adjusted R-squared:  0.6921
## F-statistic:  2426 on 1 and 1078 DF,  p-value: < 2.2e-16
```

```r
(summary(mod3.a) -> s3a)
```

```
##
## Call:
## lm(formula = price ~ sqft, data = br)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -366641  -31399   -1535   25601  932272
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -60861.462   6110.187  -9.961   <2e-16 ***
## sqft            92.747      2.411  38.476   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 79820 on 1078 degrees of freedom
## Multiple R-squared:  0.5786, Adjusted R-squared:  0.5783
## F-statistic:  1480 on 1 and 1078 DF,  p-value: < 2.2e-16
```

```
# desenhando os dados com as curvas de regressão
plot(br$sqft, br$price, pch = 20, col = "gray")
x = seq(min(br$sqft), max(br$sqft), length.out = 100)
y = predict(mod3, data.frame(sqft = x))
abline(mod3.a, col = "red", lwd = 2)
lines(x, y, col = "blue", lwd = 2)
legend("topleft", legend = c(bquote(paste("Primeiro grau: ",
    R[aj]^2 == .(s3a$adj.r.squared %>%
        round(4)))), bquote(paste("Segundo grau: ", R[aj]^2 ==
    .(s3$adj.r.squared %>%
        round(4))))), cex = 0.8, fill = c("red", "blue"))
grid()
```




```
b1 <- coef(mod3)[[1]]
b2 <- coef(mod3)[[2]]
sqftx = c(2000, 4000, 6000)  #given values for sqft
pricex = b1 + b2 * sqftx^2  #prices corresponding to given sqft
DpriceDsqft <- 2 * b2 * sqftx  # marginal effect of sqft on price
elasticity = DpriceDsqft * sqftx/pricex
b1
```

```
## [1] 55776.57
```

```
b2
```

```
## [1] 0.0154213
```

```
DpriceDsqft
```

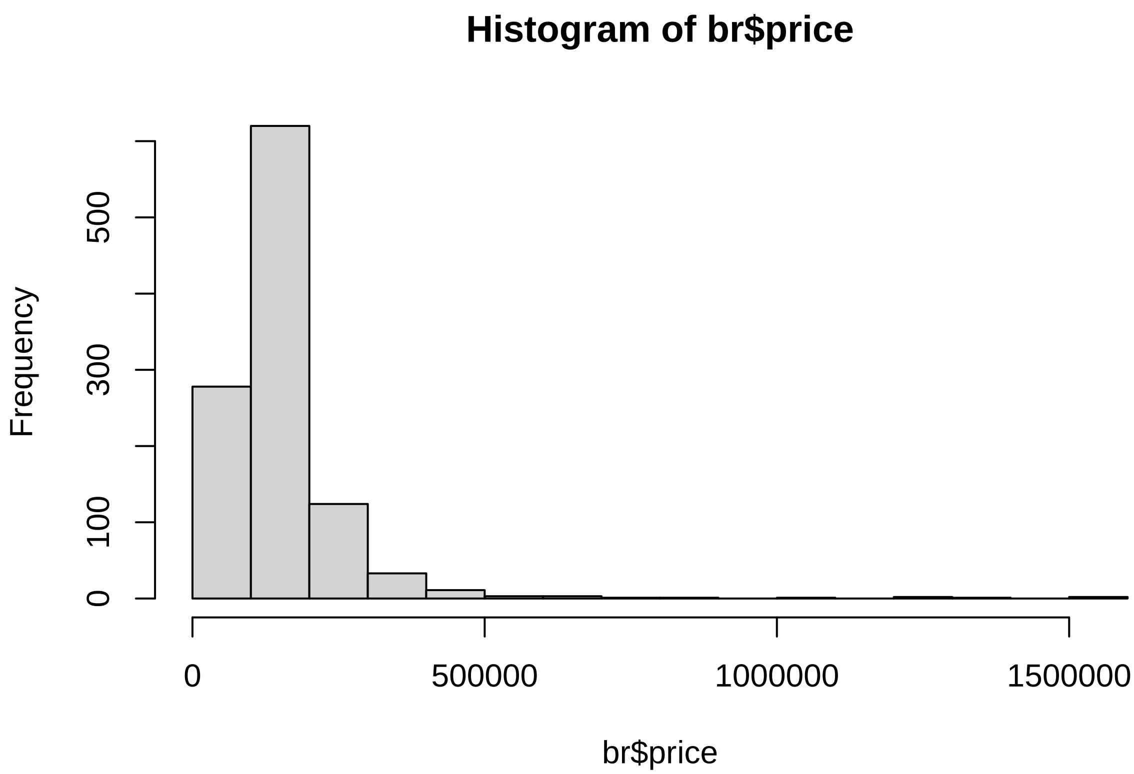
```
## [1]  61.68521 123.37041 185.05562
```

```
elasticity  #prints results
```

```
## [1] 1.050303 1.631251 1.817408
```

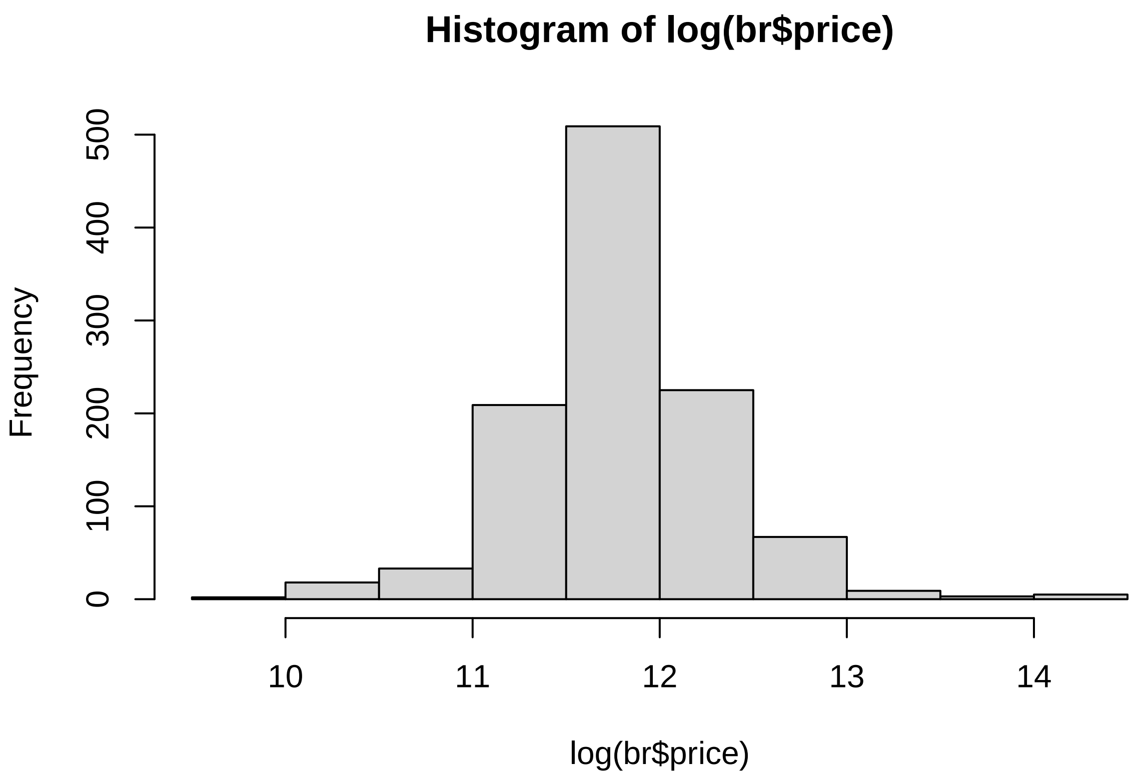
```r
hist(br$price)
```


1.7.1 Verificando a variável dependente



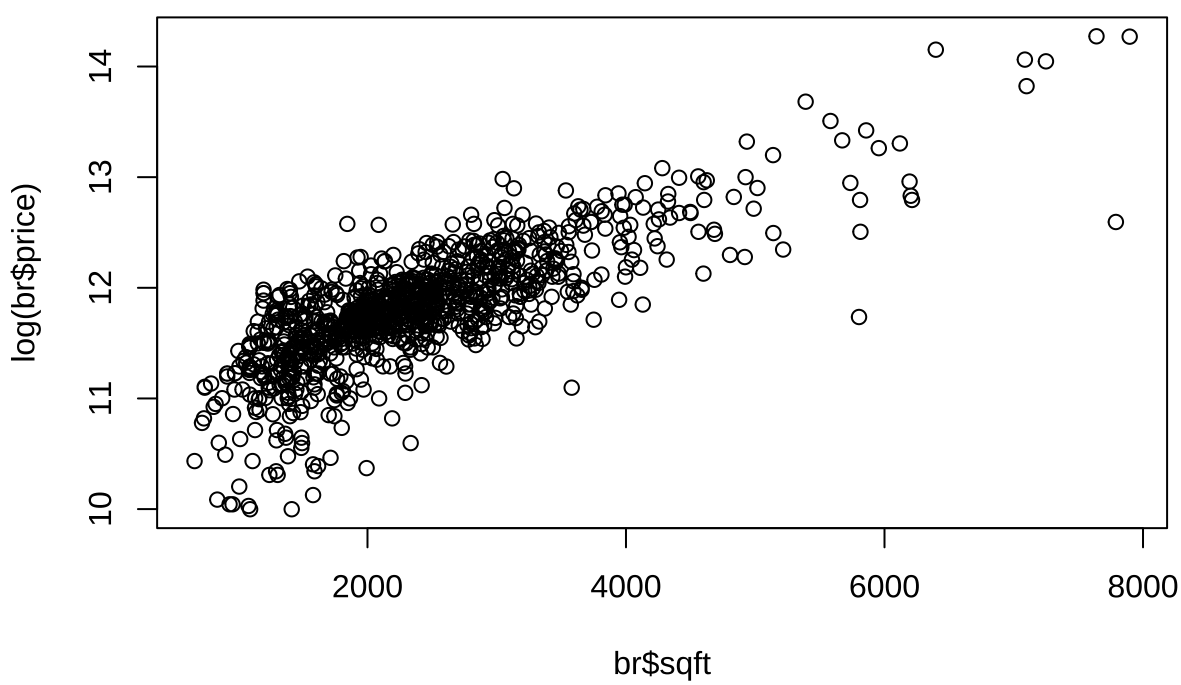
Transformação de variáveis

```r
hist(log(br$price))
```



1.7.3 Transformação logarítmica

```
plot(br$sqft, log(br$price))
```



1.8 Using Indicator Variables in a Regression

Variável indicadora = dummy

$$dummy \in \{0, 1\}$$

utown = university town

```
data(utown)
head(utown)
```

```
##     price  sqft age utown pool fplace
## 1 205.452 23.46   6     0    0      1
## 2 185.328 20.03   5     0    0      1
## 3 248.422 27.77   6     0    0      0
## 4 154.690 20.17   1     0    0      0
## 5 221.801 26.45   0     0    0      1
## 6 199.119 21.56   6     0    0      1
```

```
mod5 = lm(price ~ utown, data = utown)
summary(mod5)
```

```
##
## Call:
## lm(formula = price ~ utown, data = utown)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -85.672 -20.359  -0.462  20.646  67.955
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  215.732      1.318  163.67   <2e-16 ***
## utown         61.509      1.830   33.62   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.91 on 998 degrees of freedom
## Multiple R-squared:  0.5311, Adjusted R-squared:  0.5306
## F-statistic:  1130 on 1 and 998 DF,  p-value: < 2.2e-16
```

Fora de utown preço = 215.732 (215.7324948)

Dentro de utown preço = 215.7324948 + 61.5091064 = 277.2416012

```
mean(utown[utown$utown == 1, "price"])
```

```
## [1] 277.2416
```

```
mean(utown[utown$utown == 0, "price"])
```

```
## [1] 215.7325
```

```
library(magrittr)
utown[utown$utown == 1, "price"] %>%
    mean
```

```
## [1] 277.2416
```

```
utown[utown$utown == 0, "price"] %>%
    mean
```

```
## [1] 215.7325
```

1.9 Monte Carlo

Vamos ver depois

Capítulo 2

Chapter 3 Interval Estimation and Hypothesis Testing

2.1 Example: Confidence Intervals in the food Model

```
library(PoEdata)
data("food")
alpha <- 0.05  # chosen significance level
mod1 <- lm(food_exp ~ income, data = food)
b2 <- coef(mod1)[[2]]
df <- df.residual(mod1)  # degrees of freedom
smod1 <- summary(mod1)
seb2 <- coef(smod1)[2, 2]  # se(b2)
tc <- qt(1 - alpha/2, df)
lowb <- b2 - tc * seb2  # lower bound
upb <- b2 + tc * seb2  # upper bound
c(lowb, b2, upb)
```

```
## [1]  5.972052 10.209643 14.447233
```

Tenho 1-significância = 1 - 0.05 = 0.95 = 95% de CONFIANÇA que o valor do coeficiente angular está situado entre 5.9720525 e 14.4472334.

```
confint(mod1, level = 0.95)
```

```
##                 2.5 %    97.5 %
## (Intercept) -4.463279 171.29528
## income       5.972052  14.44723
```

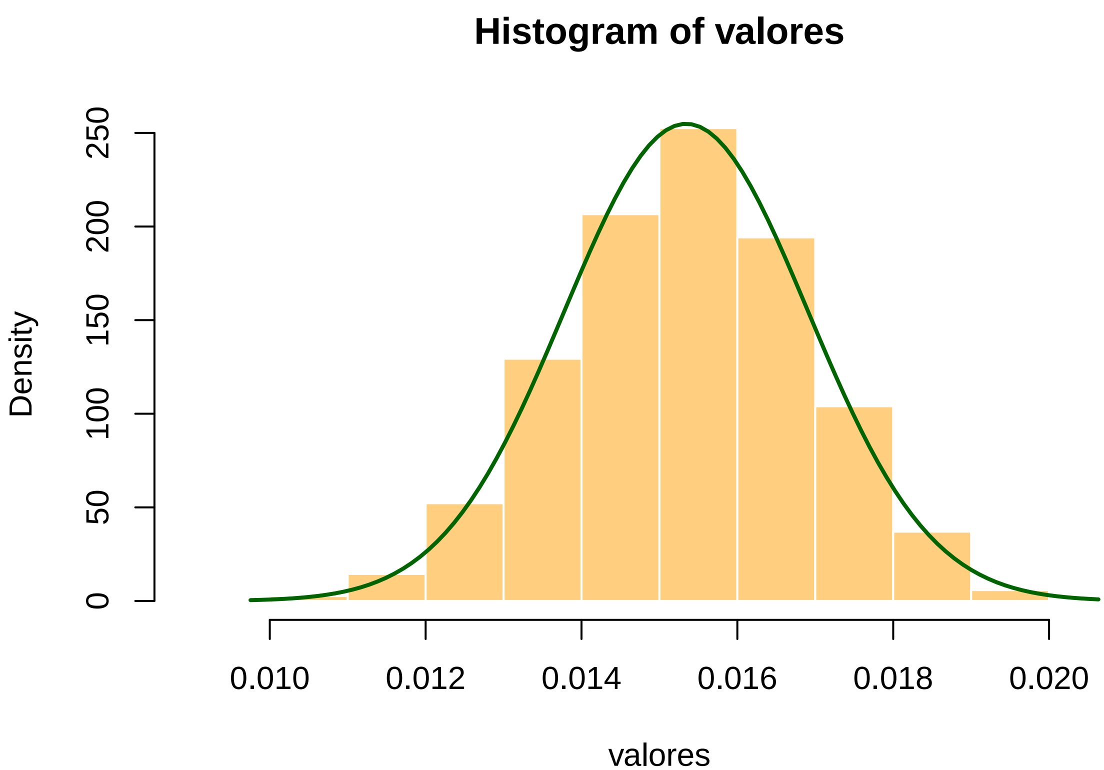
2.2 Bootstrap

Reamostragem

```
# Travo o gerador de números pseudo-aleatórios
set.seed(1)
# Número de simulações
N = 5000
# Número de elementos na amostra
nrow(br) -> n
```

```r
# Inicializo vetor de valores
valores = NULL
# loop de reamostragem
for (i in 1:N) {
    # crio amostra de tamanho n com repetição
    sample(1:n, n, replace = TRUE) -> idx
    # faço a regressão
    lm(price ~ I(sqft^2), data = br[idx, ]) -> modb
    # guardo o valor do coeficiente angular
    valores = c(valores, modb$coefficients[2])
}
```

```r
# desenho um histograma com uma curva normal superimposta
hist(valores, freq = FALSE, col = "#FFA00080", border = "white")
curve(dnorm(x, mean(valores), sd(valores)), xlim = c(min(valores),
    max(valores)), add = TRUE, col = "darkgreen", lwd = 2)
```




```r
c(mod3$coefficients[2], mean(valores))
```

```
##  I(sqft^2)
## 0.01542130 0.01534299
```

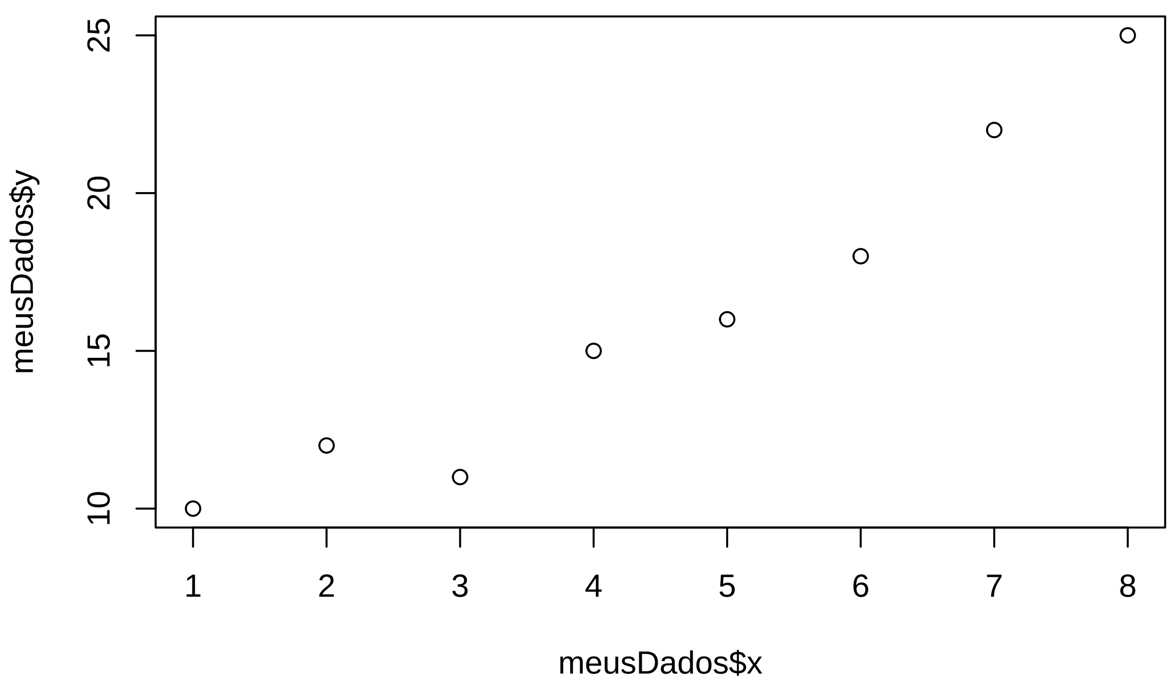
```r
# Teste de normalidade (veremos em um futuro próximo)
shapiro.test(sample(valores, 400))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  sample(valores, 400)
## W = 0.99552, p-value = 0.3093
```

Capítulo 3

Adendo - ler dados do EXCEL

```r
# file.choose()
library(openxlsx)
read.xlsx("/Users/jfrega/Downloads/DadosTeste.xlsx", sheet = "Planilha1",
    startRow = 1) -> meusDados
plot(meusDados$x, meusDados$y)
```




```r
lm(y ~ x, meusDados) %>%
    summary
```

```
##
## Call:
## lm(formula = y ~ x, data = meusDados)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9643 -1.2054  0.2679  1.1696  1.5000
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.6429     1.1198   5.932  0.00102 **
## x             2.1071     0.2218   9.502 7.75e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.437 on 6 degrees of freedom
## Multiple R-squared:  0.9377, Adjusted R-squared:  0.9273
## F-statistic: 90.29 on 1 and 6 DF,  p-value: 7.745e-05
```

Capítulo 4

Adendo — Regressão OLS em Python

```python
#
# ATENÇÃO: para rodar este trecho do código é necessário ter o Python instalado e configurado
#
import statsmodels.formula.api as smf
# vou acessar os dados que foram lidos no meu código em R
r.meusDados
```

```
##      x     y
## 0  1.0  10.0
## 1  2.0  12.0
## 2  3.0  11.0
## 3  4.0  15.0
## 4  5.0  16.0
## 5  6.0  18.0
## 6  7.0  22.0
## 7  8.0  25.0
```

```python
model = smf.ols(formula="y~x", data=r.meusDados)
model.fit().summary()
```

```
## <class 'statsmodels.iolib.summary.Summary'>
## """
##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                      y   R-squared:                       0.938
## Model:                            OLS   Adj. R-squared:                  0.927
## Method:                 Least Squares   F-statistic:                     90.29
## Date:                Wed, 19 Mar 2025   Prob (F-statistic):           7.75e-05
## Time:                        14:38:36   Log-Likelihood:                -13.102
## No. Observations:                   8   AIC:                             30.20
## Df Residuals:                       6   BIC:                             30.36
## Df Model:                           1
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## Intercept      6.6429      1.120      5.932      0.001       3.903       9.383
## x              2.1071      0.222      9.502      0.000       1.565       2.650
## ==============================================================================
```

```
## Omnibus:                       2.183   Durbin-Watson:                   1.522
## Prob(Omnibus):                 0.336   Jarque-Bera (JB):                0.848
## Skew:                         -0.279   Prob(JB):                        0.654
## Kurtosis:                      1.505   Cond. No.                         11.5
## ==============================================================================
##
## Notes:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
## """
##
## /Users/jfrega/Library/r-miniconda-arm64/lib/python3.10/site-packages/scipy/stats/_axis_nan_policy.py:418
##   return hypotest_fun_in(*args, **kwds)
```