# Métodos Numéricos Aplicados a Finanças — Turma 2025

Todas as aulas

Prof. Frega

11/03/2025

# Sumário

# AULA 1

# Capítulo 1

# Introdução

Aqui começamos a escrever a introdução do nosso material.

Aqui continuamos

A seguir vamos colocando outros ítens tipográficos

## 1.1 Objetivos

### 1.1.1 Objetivo Geral

### 1.1.2 Objetivo Específico

#### 1.1.2.1 Subsubseção

##### 1.1.2.1.1 Parágrafo

###### 1.1.2.1.1.1 **Subparágrafo** Subsubparágrafo

Um subsubparágrafo é aceito pelo *markdown* mas não é definido tipograficamente. Normalmente usamos só até o nível 6, que é o subparágrafo.

# Capítulo 2

# Referencial teórico-empírico

# Capítulo 3

# Metodologia

Escrevendo uma equação

```
$$
y = ax^2+bx+c
$$
```

$$y = ax^2 + bx + c$$

# Capítulo 4

# Análise de dados e discussão dos resultados

```
2 + 2
```

```
## [1] 4
```

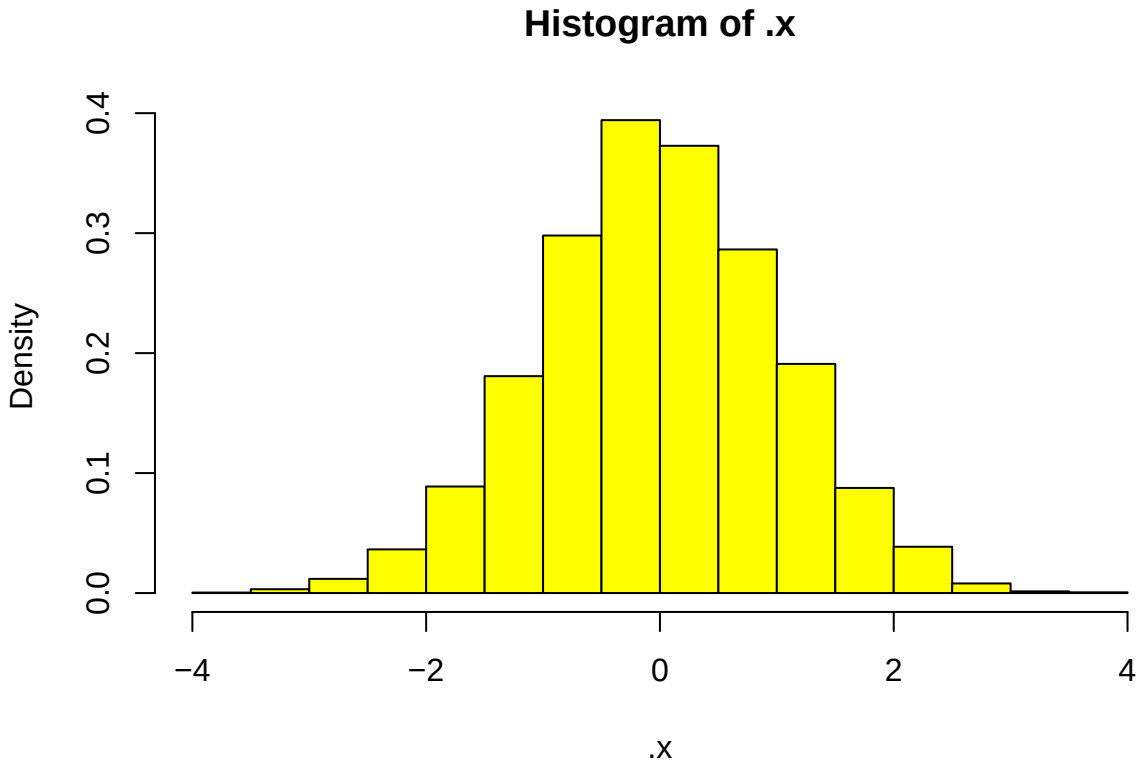
```
set.seed(1)
.x <- rnorm(10000)
head(.x, 10)
```

```
##  [1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.3295078 -0.8204684
##  [7]  0.4874291  0.7383247  0.5757814 -0.3053884
```

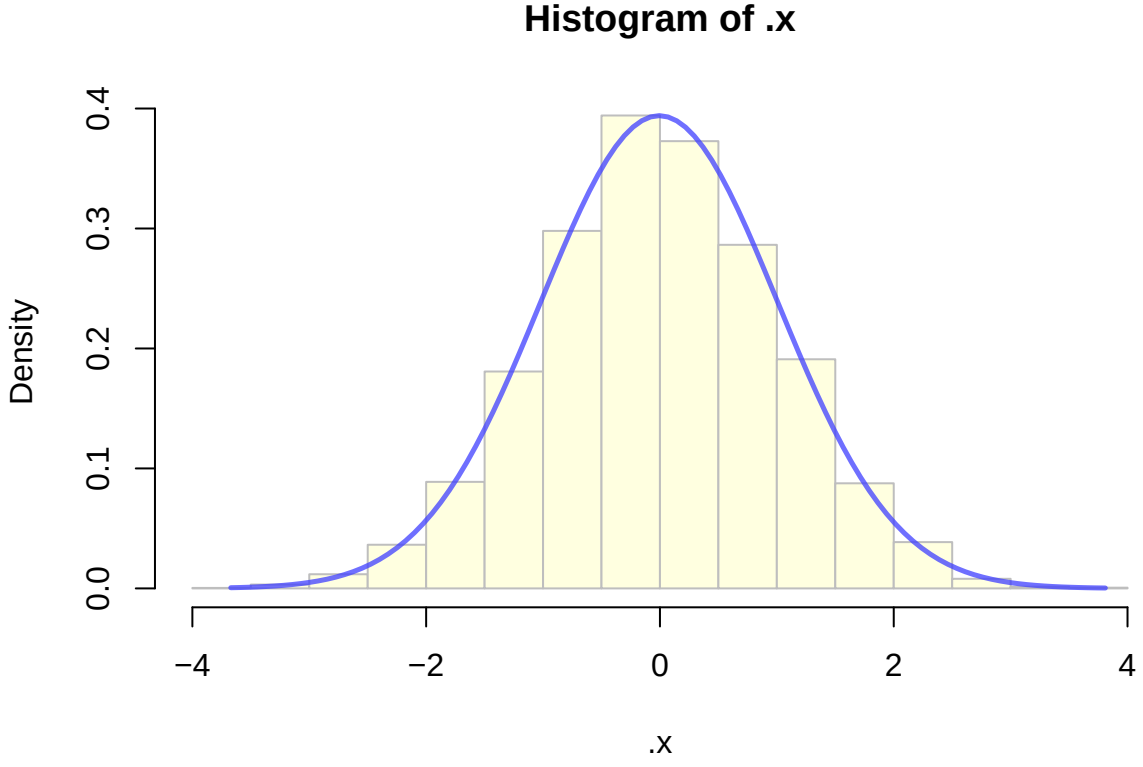
```
tail(.x, 10)
```

```
##  [1]  1.04776175 -0.02428861 -0.47787499 -0.02971747  0.20966546  0.95950757
##  [7]  0.43660362  0.49936656  0.89397983  0.25738706
```

```
hist(.x, freq = FALSE, col = "yellow", border = "black")
```




```
hist(.x, freq = FALSE, col = "lightyellow", border = "gray")
curve(dnorm(x, mean(.x), sd(.x)), xlim = c(min(.x), max(.x)),
    add = TRUE, col = "#4040FFC0", lwd = 2.5)
```



```r
1/2
```

```
## [1] 0.5
```

# Capítulo 5

# Dados do Principles of Econometrics

## 5.1 Pacote `PoEdata_0.1.0.tar.gz`

Uma vez instalado o pacote PoEdata_0.1.0.tar.gz

```
library(PoEdata)
```

```
library(printr)
```

```
## Registered S3 method overwritten by 'printr':
##   method                from
##   knit_print.data.frame rmarkdown
```

```
data(mroz)
head(mroz[, 1:5])
```

| taxableinc | federaltax | hsiblings | hfathereduc | hmothereduc |
|---|---|---|---|---|
| 12200 | 1494 | 1 | 14 | 16 |
| 18000 | 2615 | 8 | 7 | 3 |
| 24000 | 3957 | 4 | 7 | 10 |
| 16400 | 2279 | 6 | 7 | 12 |
| 10000 | 1063 | 3 | 7 | 7 |
| 6295 | 370 | 8 | 7 | 7 |

```
tail(mroz[, 1:5])
```

| | taxableinc | federaltax | hsiblings | hfathereduc | hmothereduc |
|---|---|---|---|---|---|
| 748 | 16100 | 1825 | 0 | 7 | 12 |
| 749 | 32000 | 4701 | 8 | 12 | 7 |
| 750 | 18500 | 2720 | 4 | 12 | 12 |
| 751 | 13000 | 1642 | 6 | 7 | 12 |
| 752 | 17200 | 2447 | 2 | 7 | 10 |
| 753 | 18700 | 2327 | 4 | 10 | 7 |

## 5.2 Pequeno exemplo de programação em R

```
library(DescTools)
plotSquare = function(deltay = 1.5, deltax = abs(deltay/Asp()),
    xbase = 12, ybase = 3, col = "#FF808040") {
    polygon(c(0, deltax, deltax, 0, 0) + xbase, c(0, 0, deltay,
        deltay, 0) + ybase, col = col)
}
plotRegressao = function(modelo1 = modelo1, horas = horas, nota = nota,
    sequencia = 5, sub = NULL) {
    # desenho os pontos observados
    plot(horas, nota, pch = 20, col = "darkgray", xlim = c(0,
        14), ylim = c(0, 100), axes = FALSE, sub = "Regressão linear simples",
        xlab = "Horas de estudo", ylab = "Nota na avaliação",
        main = sub)
    # desenho os eixos no (0, 0)
    axis(1, pos = 0)
    axis(2, pos = 0)
    # traça a linha do modelo1 em azul
    if (sequencia > 1)
        abline(modelo1, col = "blue")
    # calcula os pontos sobre a reta
    estimados <- predict(modelo1, horas = horas)
    # desenha os pontos sobre a reta
    if (sequencia > 2)
        points(horas, estimados, pch = 20, col = "blue")
    # desenha as barras de erro (y - ychapéu) e dá nome aos
    # pontos
    delta = 0.3
    if (sequencia > 3) {
        for (i in 1:length(horas)) {
            # desenha as barras de erro verticais
            lines(c(horas[i], horas[i]), c(estimados[i], nota[i]),
                col = "red")
            # desenha as linhas horizontais
            lines(c(horas[i] - delta, horas[i] + delta), c(estimados[i],
                estimados[i]), col = "red")
            lines(c(horas[i] - delta, horas[i] + delta), c(nota[i],
                nota[i]), col = "red")
            # coloca o nome do ponto acima ou abaixo dele
            # conforme a estética
            text(horas[i], nota[i], bquote(y[.(i)]), pos = ifelse(estimados[i] >
                nota[i], 1, 3))
        }
    }
    if (sequencia > 4) {
        for (i in 1:length(horas)) {
            deltay = nota[i] - estimados[i]
            deltax = abs(deltay/Asp())
            if (deltay > 0)
                deltax = -deltax
            plotSquare(xbase = horas[i], ybase = estimados[i],
                deltay = deltay, deltax = deltax)
        }
```

```
    }
    text(8, 10, expression(min(Sigma(y[i] - bar(y[i]))^2)), cex = 1.2)
    text(8, 15, "Mínimos quadrados ordinários minimiza o somatório dos quadrados dos erros",
        cex = 0.7)
}
```

```
# conjuntos de dados
nota <- c(40, 30, 60, 65, 70, 90)
horas <- c(2, 4, 6, 8, 10, 12)
# função lm (linear model) -> guarda em modelo1
lm(nota ~ horas) -> modelo1
plotRegressao(modelo1, horas, nota, 1, expression("Nuvem de pontos"))
```

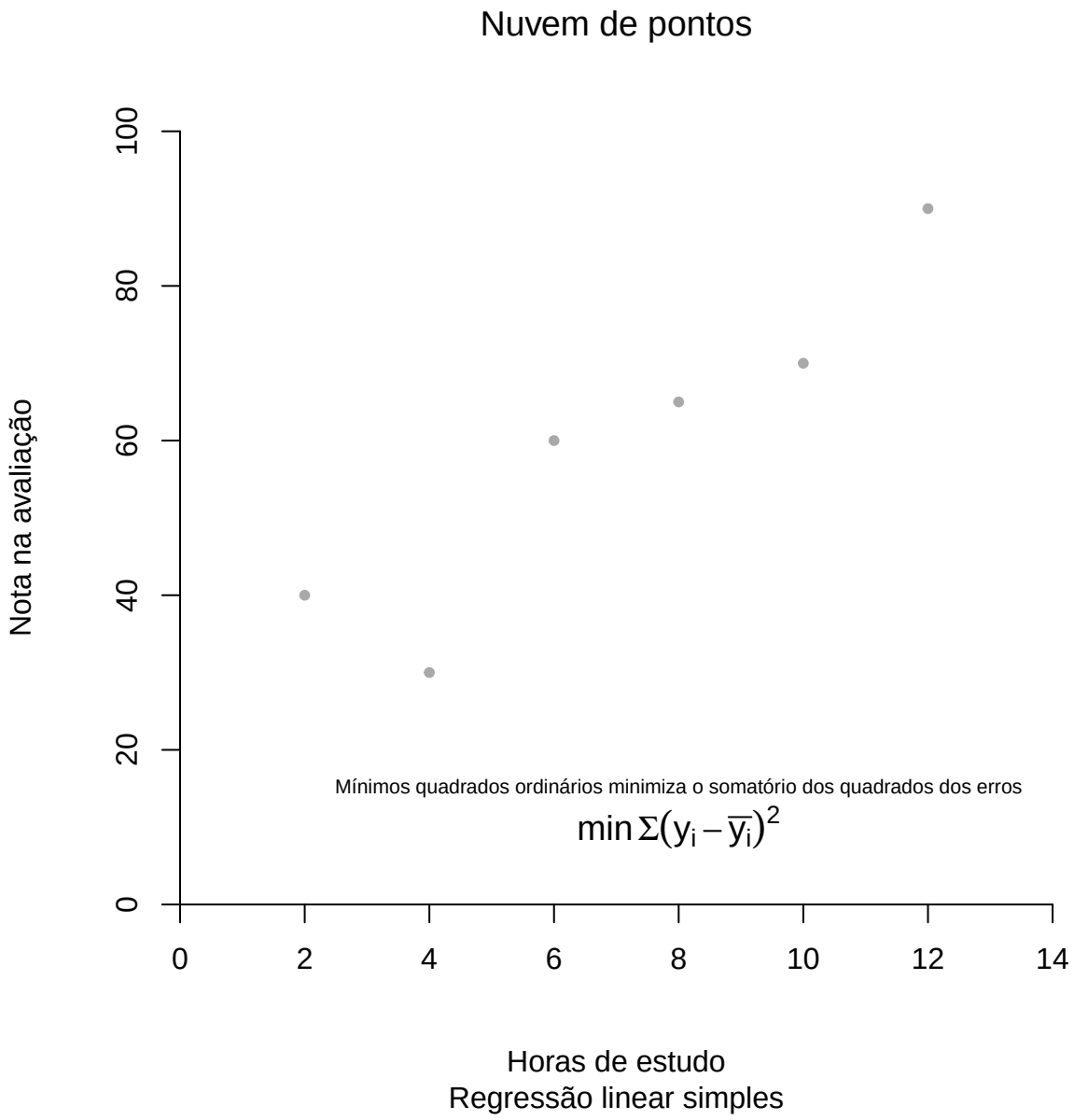



Figura 5.1: Nota na avaliação *versus* horas de estudo

```
plotRegressao(modelo1, horas, nota, 2, expression(paste("Reta de regressão: ",
    nota == b[0] + b[1] * horas)))
```

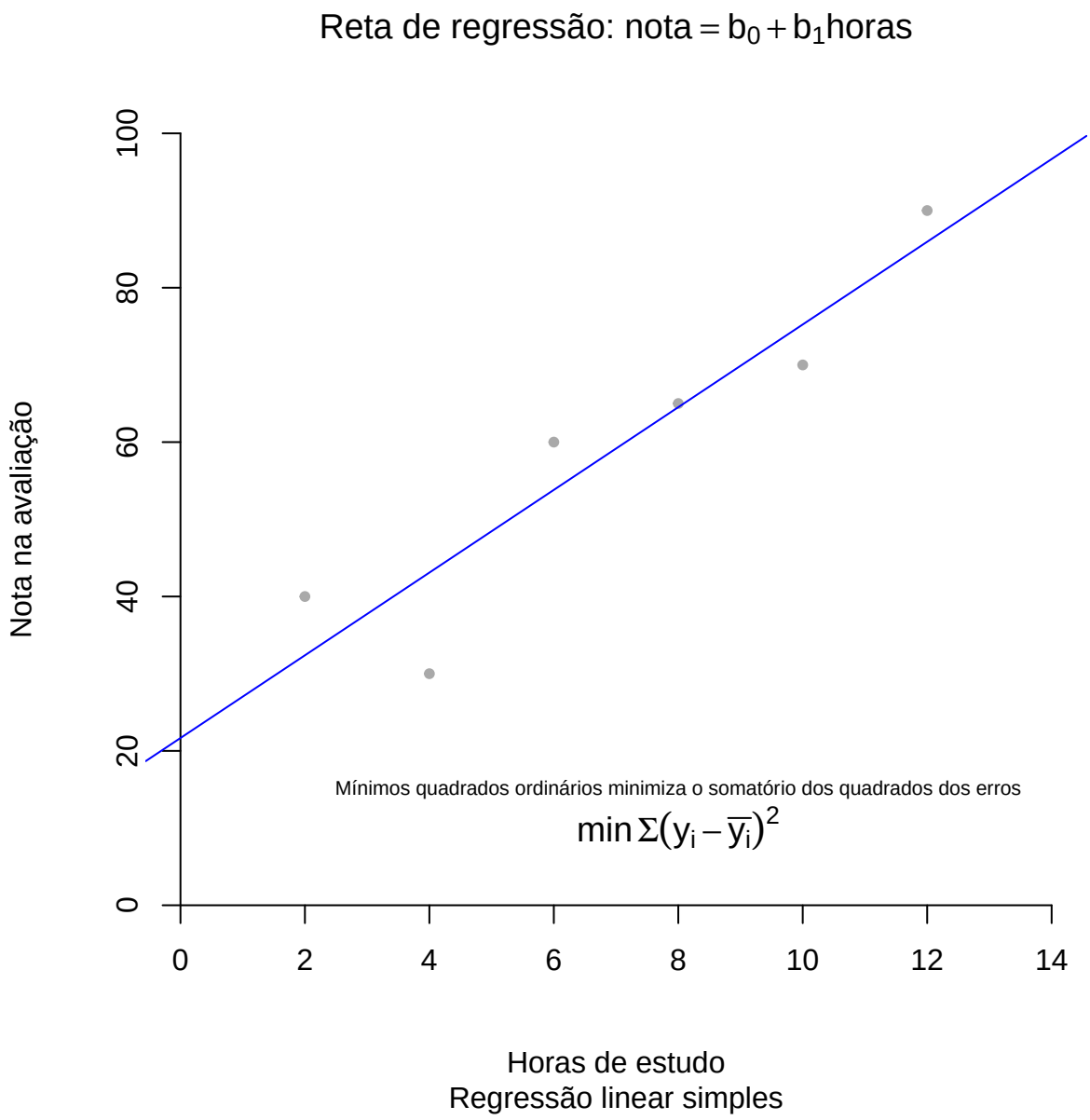



Figura 5.2: Nota na avaliação *versus* horas de estudo

```
plotRegressao(modelo1, horas, nota, 3, expression(paste("Reta de regressão com as estimativas ",
    hat(y))))
```

```
plotRegressao(modelo1, horas, nota, 4, expression(paste("Erros observados: ",
    y - hat(y))))
```

```
plotRegressao(modelo1, horas, nota, 5, expression(paste("Quadrados dos erros: ",
    (y - hat(y))^2)))
```

```
modelo1
```

```
##
```

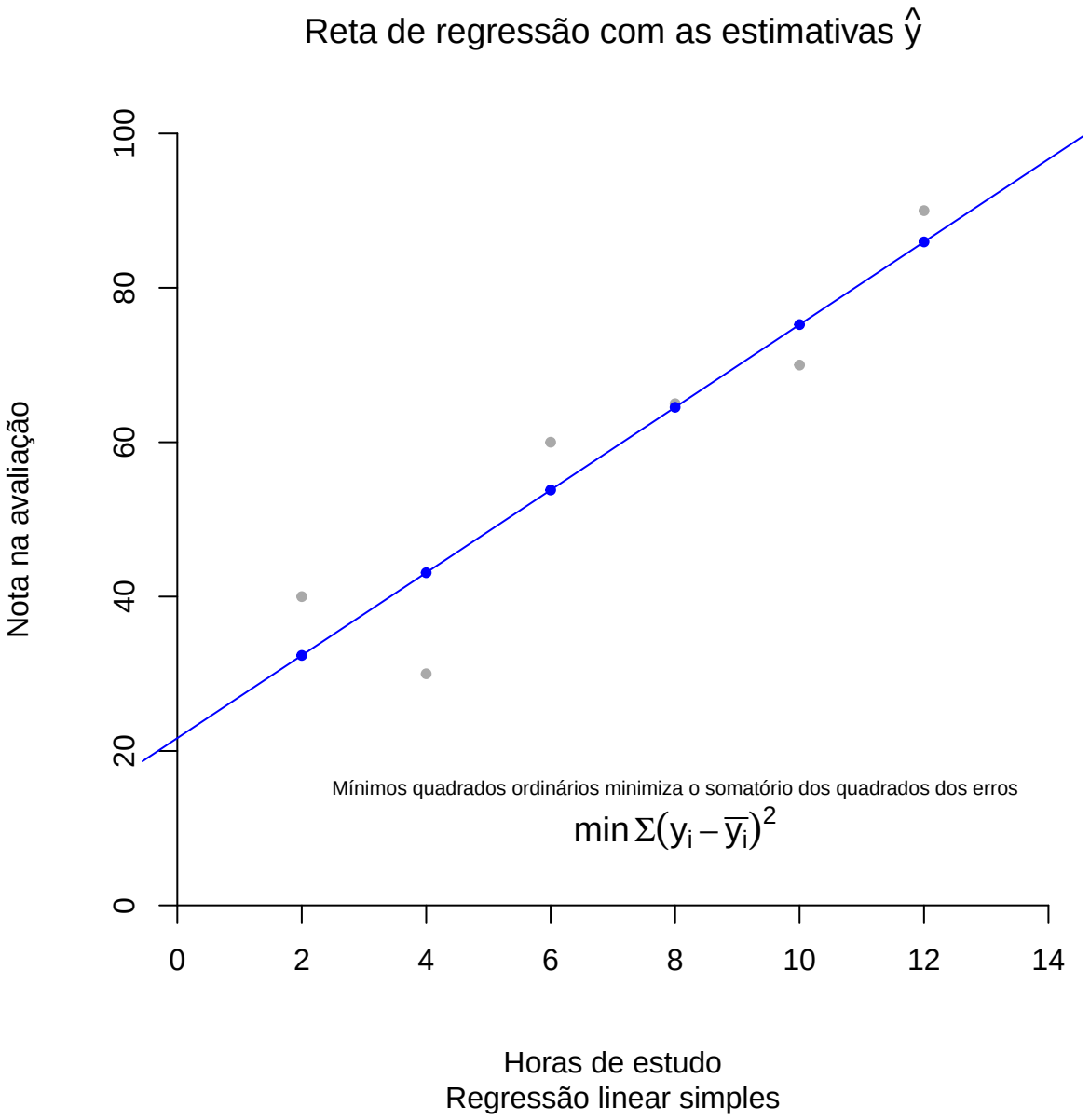



Figura 5.3: Nota na avaliação *versus* horas de estudo

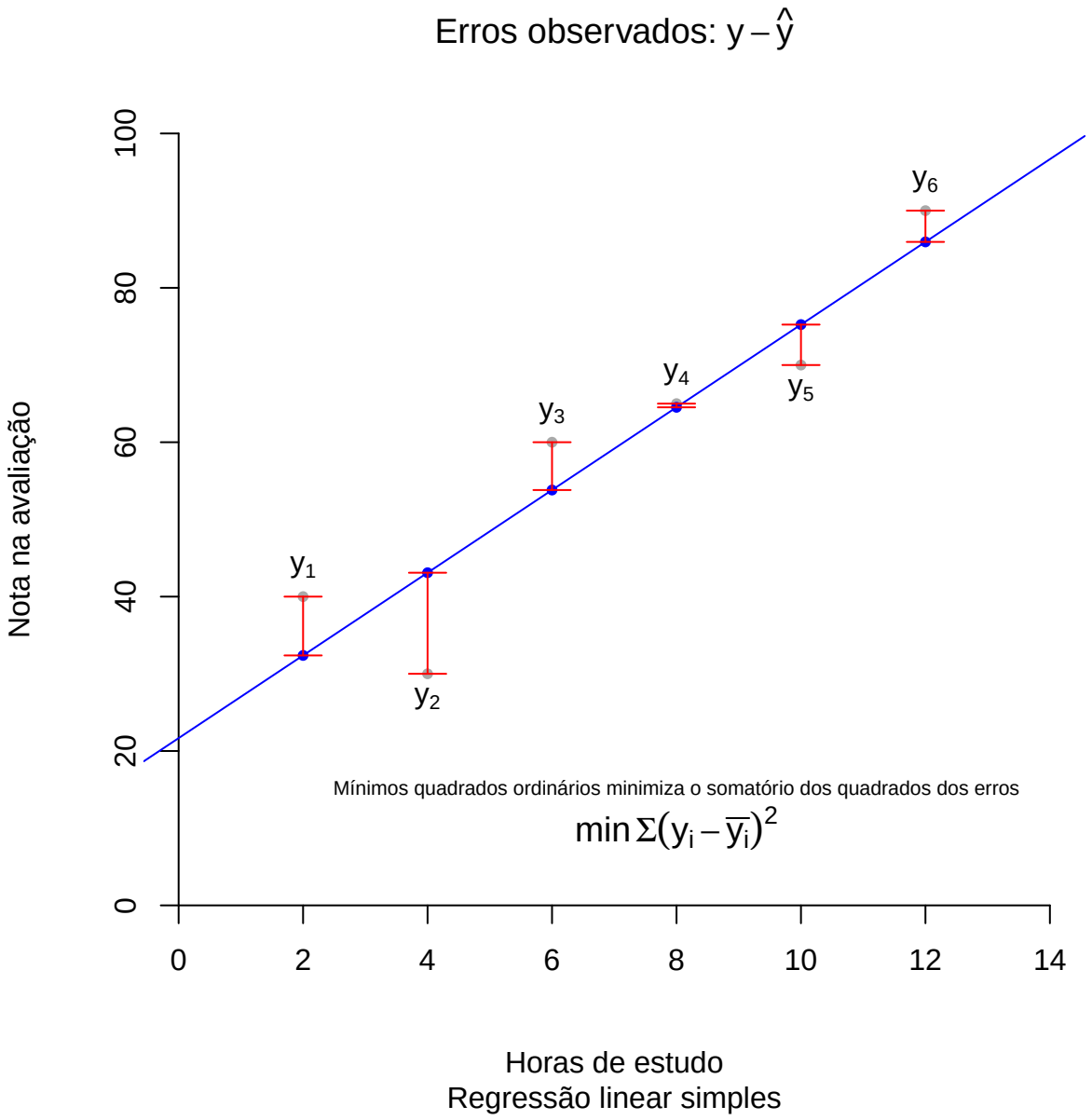



Figura 5.4: Nota na avaliação *versus* horas de estudo

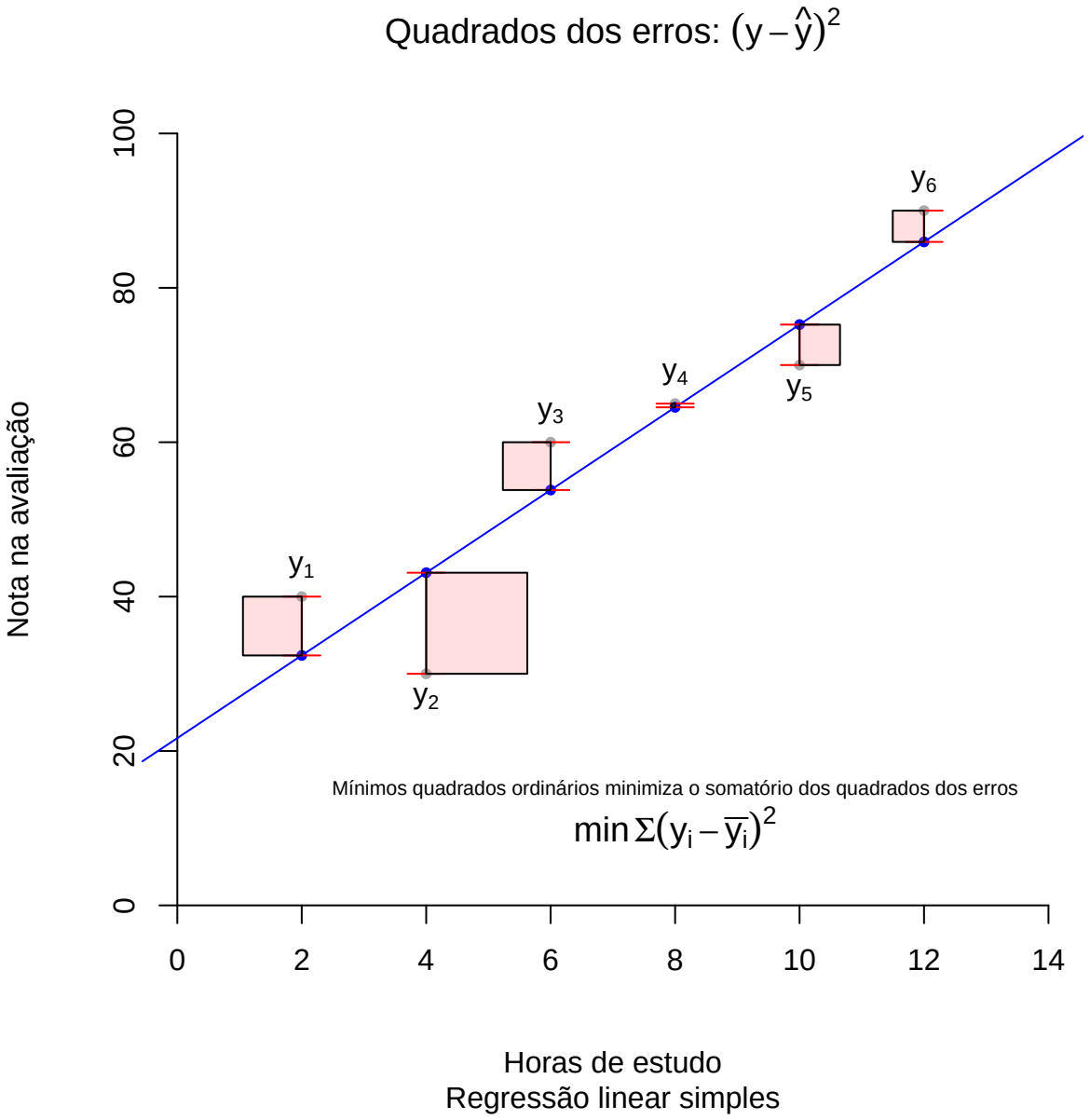



Figura 5.5: Nota na avaliação *versus* horas de estudo

```
## Call:
## lm(formula = nota ~ horas)
##
## Coefficients:
## (Intercept)        horas
##      21.667        5.357
```

```
modelo1$coefficients[1]
```

```
## (Intercept)
##    21.66667
```

```
modelo1$coefficients[2]
```

```
##    horas
## 5.357143
```

```
$$
\widehat{\text{nota}} = b_0+b_1\cdot \text{horas} = 21.6666667+5.3571429\cdot \text{horas}
$$
```

$$\widehat{\text{nota}} = b_0 + b_1 \cdot \text{horas} = 21.6666667 + 5.3571429 \cdot \text{horas}$$

Diagnósticos do modelo

Estatística = testes de hipóteses

Uma hipótese pode ser rejeitada ou não

Existe um valor calculado para cada teste que se chama p.value (p-valor) para o qual existe um valor crítico, normalmente tomado como 0,05 (ou 5%) que chamamos de significância do teste.

Todo teste tem uma hipótese nula ($H_0$), se o p-valor for menor que o limite, rejeita-se $H_0$, se for igual ou maior, aceita-se $H_0$.

$H_0$ do teste F: não há relação entre as variáveis.

$H_0$ do teste t: o coeficiente associado é igual a zero.

```
summary(nota)
```

| Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. |
|---|---|---|---|---|---|
| 30 | 45 | 62.5 | 59.16667 | 68.75 | 90 |

```
summary(horas)
```

| Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. |
|---|---|---|---|---|---|
| 2 | 4.5 | 7 | 7 | 9.5 | 12 |

```
summary(.x)
```

| Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. |
|---|---|---|---|---|---|
| -3.6713 | -0.6733944 | -0.0159288 | -0.006537 | 0.6776605 | 3.810277 |

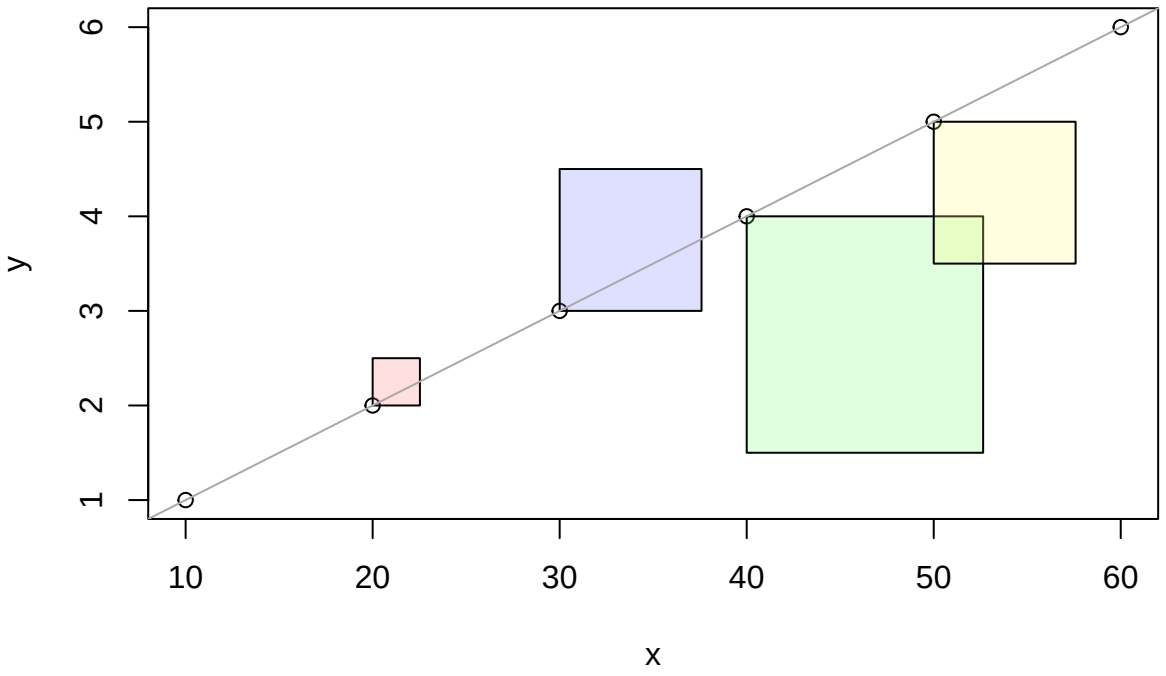
```r
summary(modelo1)
```

```
##
## Call:
## lm(formula = nota ~ horas)
##
## Residuals:
##        1        2        3        4        5        6
##   7.6190 -13.0952   6.1905   0.4762  -5.2381   4.0476
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   21.667      8.221   2.636   0.0578 .
## horas          5.357      1.055   5.076   0.0071 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.83 on 4 degrees of freedom
## Multiple R-squared:  0.8656, Adjusted R-squared:  0.832
## F-statistic: 25.76 on 1 and 4 DF,  p-value: 0.007102
```

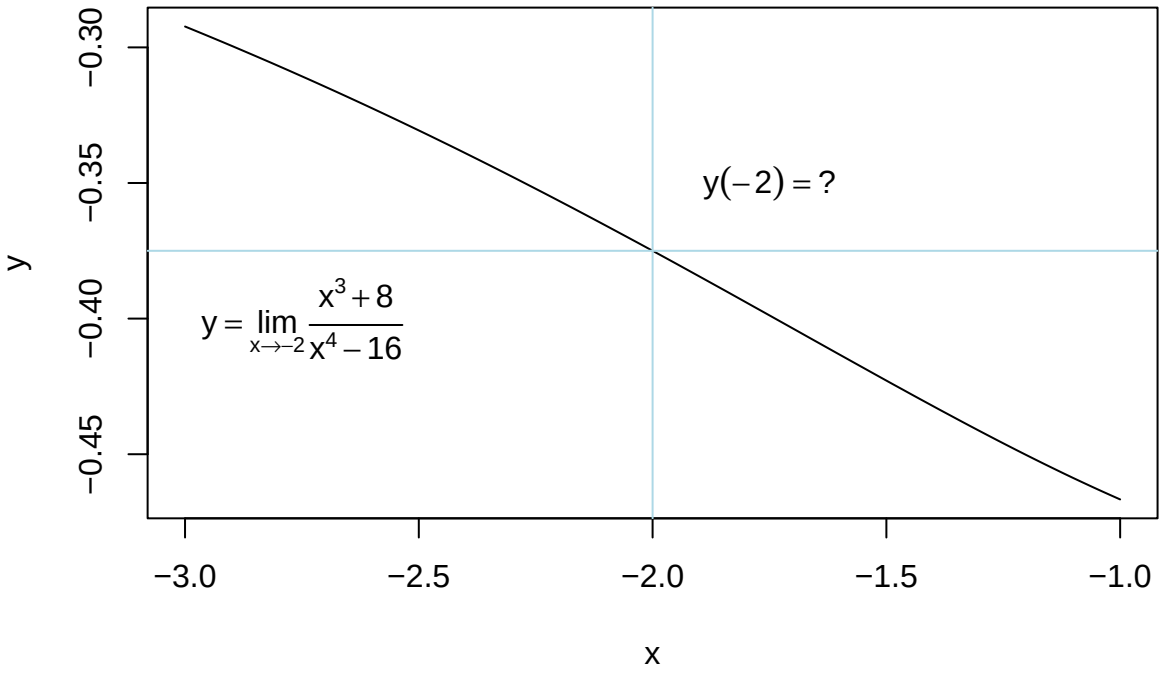
$R^2$ é a porção de variação da nota que é explicada pelas horas.

Ou seja, aproximadamente 83% da variação da nota é explicada pelas horas de estudo.

```r
x = 1:6 * 10
y = 1:6
plot(x, y)
plotSquare(xbase = 20, ybase = 2, deltay = 0.5)
plotSquare(xbase = 30, ybase = 3, deltay = 1.5, col = "#8080FF40")
plotSquare(xbase = 40, ybase = 4, deltay = -2.5, col = "#80FF8040")
plotSquare(xbase = 50, ybase = 5, deltay = -1.5, col = "#FFFF8040")
abline(c(0, 0.1), col = "darkgray")
```



```
f = function(x) (x^3 + 8)/(x^4 - 16)
x = seq(-3, -1.0000001, length.out = 100)
y = f(x)
plot(x, y, type = "l")
abline(v = -2, h = -3/8, col = "lightblue")
text(-2.75, -0.4, expression(y == lim(frac(x^3 + 8, x^4 - 16),
    x %->% -2)))
text(-1.75, -0.35, expression(y(-2) == "?"))
```




```
f(-2.000000001)
```

```
## [1] -0.375
```

```
-3/8
```

```
## [1] -0.375
```

```
plot.new()
plot.window(c(0, 4), c(15, 1))
text(1, 1, "universal", adj = 0)
text(2.5, 1, "\\042")
text(3, 1, expression(symbol("\"")))
text(1, 2, "existential", adj = 0)
text(2.5, 2, "\\044")
text(3, 2, expression(symbol("$")))
text(1, 3, "suchthat", adj = 0)
text(2.5, 3, "\\047")
text(3, 3, expression(symbol("'")))
text(1, 4, "therefore", adj = 0)
text(2.5, 4, "\\134")
text(3, 4, expression(symbol("\\")))
text(1, 5, "perpendicular", adj = 0)
text(2.5, 5, "\\136")
text(3, 5, expression(symbol("^")))
```

```
text(1, 6, "circlemultiply", adj = 0)
text(2.5, 6, "\\304")
text(3, 6, expression(symbol("\xc4")))
text(1, 7, "circleplus", adj = 0)
text(2.5, 7, "\\305")
text(3, 7, expression(symbol("\xc5")))
text(1, 8, "emptyset", adj = 0)
text(2.5, 8, "\\306")
text(3, 8, expression(symbol("\xc6")))
text(1, 9, "angle", adj = 0)
text(2.5, 9, "\\320")
text(3, 9, expression(symbol("\xd0")))
text(1, 10, "leftangle", adj = 0)
text(2.5, 10, "\\341")
text(3, 10, expression(symbol("\xe1")))
text(1, 11, "rightangle", adj = 0)
text(2.5, 11, "\\361")
text(3, 11, expression(symbol("\xf1")))
```

| | | |
|---|---|---|
| universal | \042 | ∀ |
| existential | \044 | ∃ |
| suchthat | \047 | ∍ |
| therefore | \134 | ∴ |
| perpendicular | \136 | ⊥ |
| circlemultiply | \304 | ⊗ |
| circleplus | \305 | ⊕ |
| emptyset | \306 | ∅ |
| angle | \320 | ∠ |
| leftangle | \341 | ⟨ |
| rightangle | \361 | ⟩ |

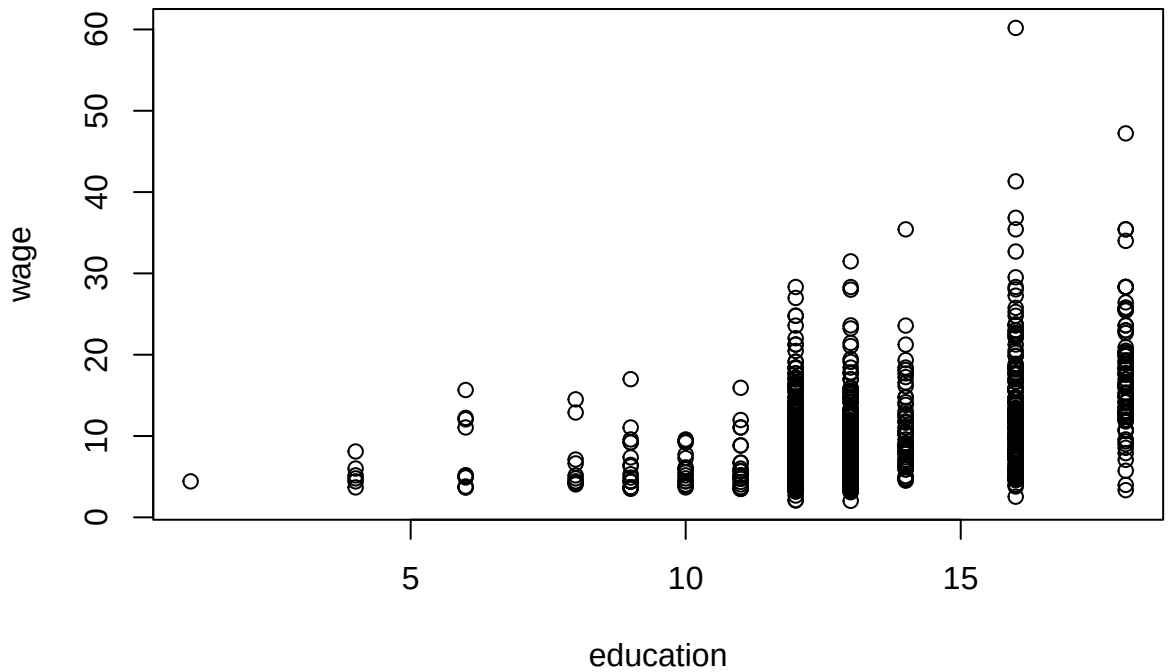
# AULA 2

```
library(magrittr)
```

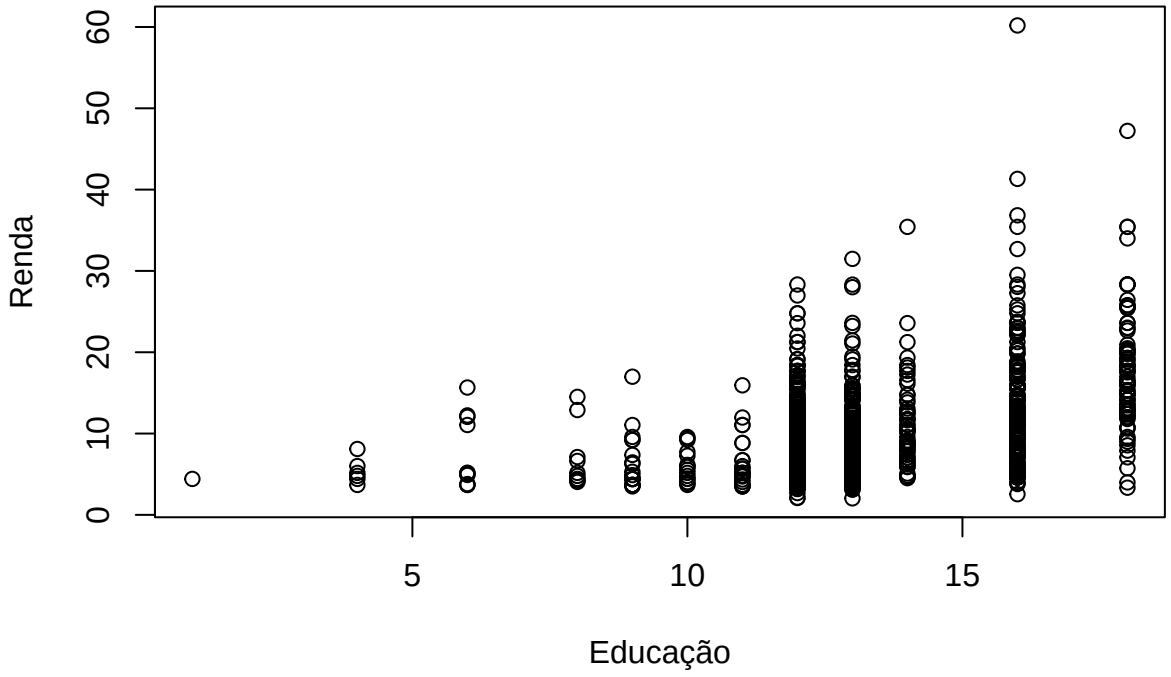
# Capítulo 6

# Regressão linear simples

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



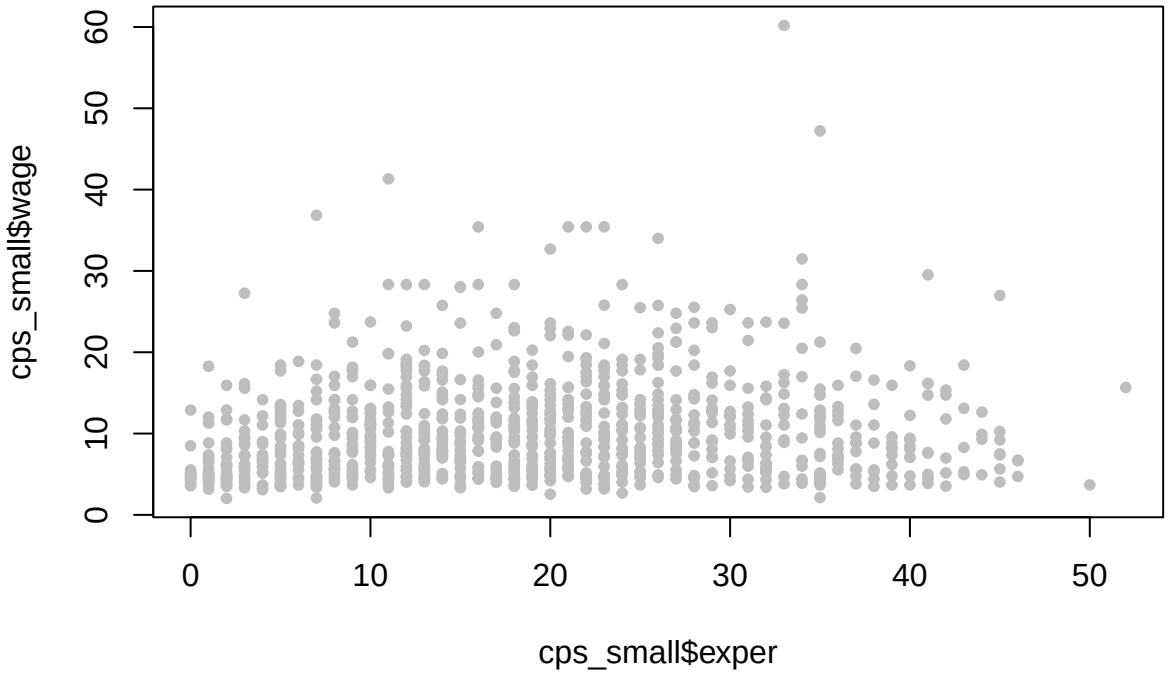

```
plot(cps_small$educ, cps_small$wage, xlab = "Educação", ylab = "Renda")
```




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

| wage | educ | exper | female | black | white | midwest | south | west |
|---|---|---|---|---|---|---|---|---|
| 2.03 | 13 | 2 | 1 | 0 | 1 | 0 | 1 | 0 |
| 2.07 | 12 | 7 | 0 | 0 | 1 | 1 | 0 | 0 |
| 2.12 | 12 | 35 | 0 | 0 | 1 | 0 | 1 | 0 |
| 2.54 | 16 | 20 | 1 | 0 | 1 | 0 | 1 | 0 |
| 2.68 | 12 | 24 | 1 | 0 | 1 | 0 | 1 | 0 |
| 3.09 | 13 | 4 | 0 | 0 | 1 | 0 | 1 | 0 |
| 3.16 | 13 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 3.17 | 12 | 22 | 1 | 0 | 1 | 0 | 1 | 0 |
| 3.20 | 12 | 23 | 0 | 0 | 1 | 0 | 1 | 0 |
| 3.27 | 12 | 4 | 1 | 0 | 1 | 0 | 0 | 1 |
| 3.32 | 12 | 11 | 1 | 0 | 1 | 0 | 0 | 1 |
| 3.32 | 13 | 3 | 1 | 0 | 1 | 1 | 0 | 0 |
| 3.34 | 18 | 15 | 0 | 0 | 1 | 1 | 0 | 0 |
| 3.39 | 13 | 7 | 1 | 0 | 1 | 0 | 0 | 0 |
| 3.39 | 12 | 15 | 1 | 0 | 1 | 0 | 0 | 1 |

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



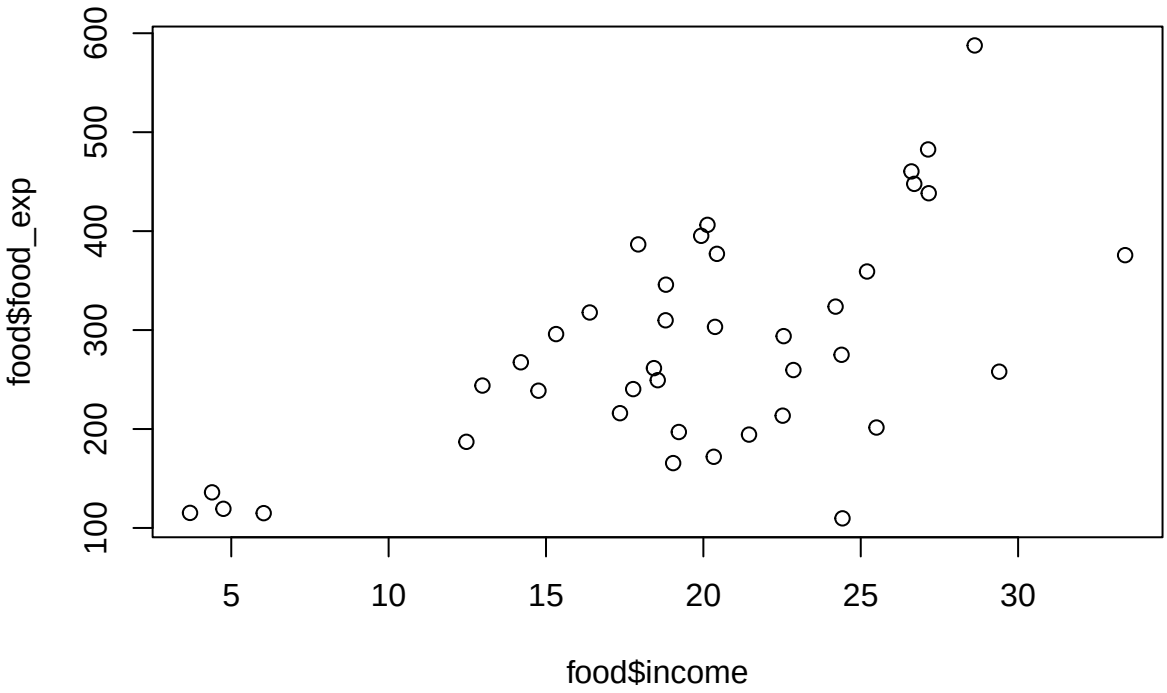

## 6.2 Example: Food Expenditure versus Income

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

| food_exp | income |
|---|---|
| 115.22 | 3.69 |
| 135.98 | 4.39 |
| 119.34 | 4.75 |
| 114.96 | 6.03 |
| 187.05 | 12.47 |
| 243.92 | 12.98 |

```
# help(food)
```

```
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")
```


## 6.3 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]])
```

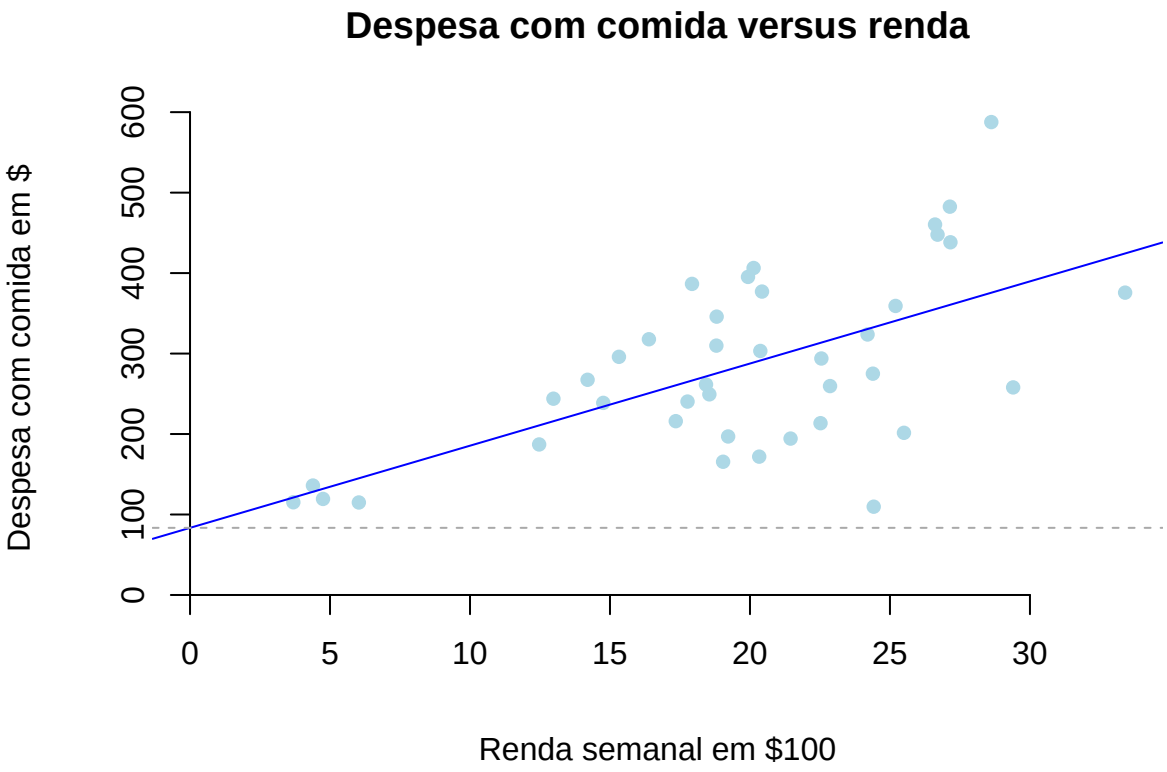
```
## [1] 10.20964
```

```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)
```




## 6.4 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
```

## 6.5 Repeated Samples to Assess Regression Coefficients

Tecnicamente isso se chama *bootstrap* e trataremos depois.

## 6.6 Estimated Variances and Covariance of Regression Coefficients

Será útil mais tarde.

## 6.7 Non-Linear Relationships

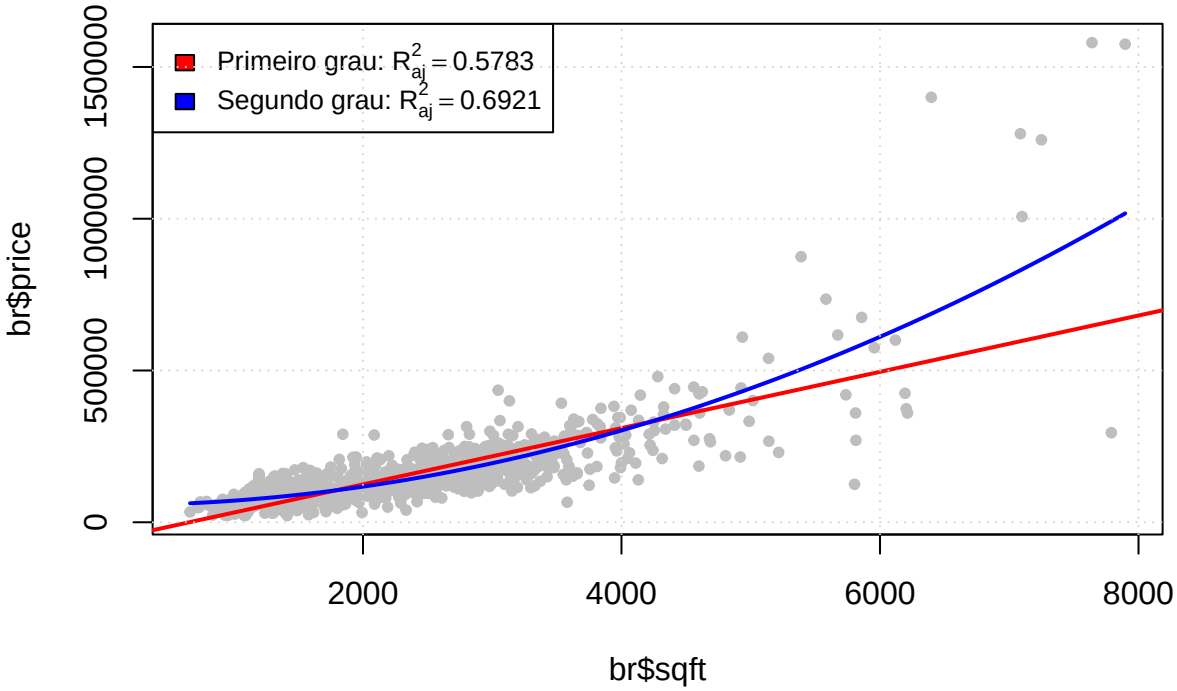
```
library(PoEdata)
data(br)
# testando uma relação quadrática
mod3 <- lm(formula = price ~ I(sqft^2), data = br)
# versus uma do primeiro grau
mod3.a <- lm(formula = price ~ sqft, data = br)
(summary(mod3) -> s3)
```

```
##
## 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
```

```
(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
```

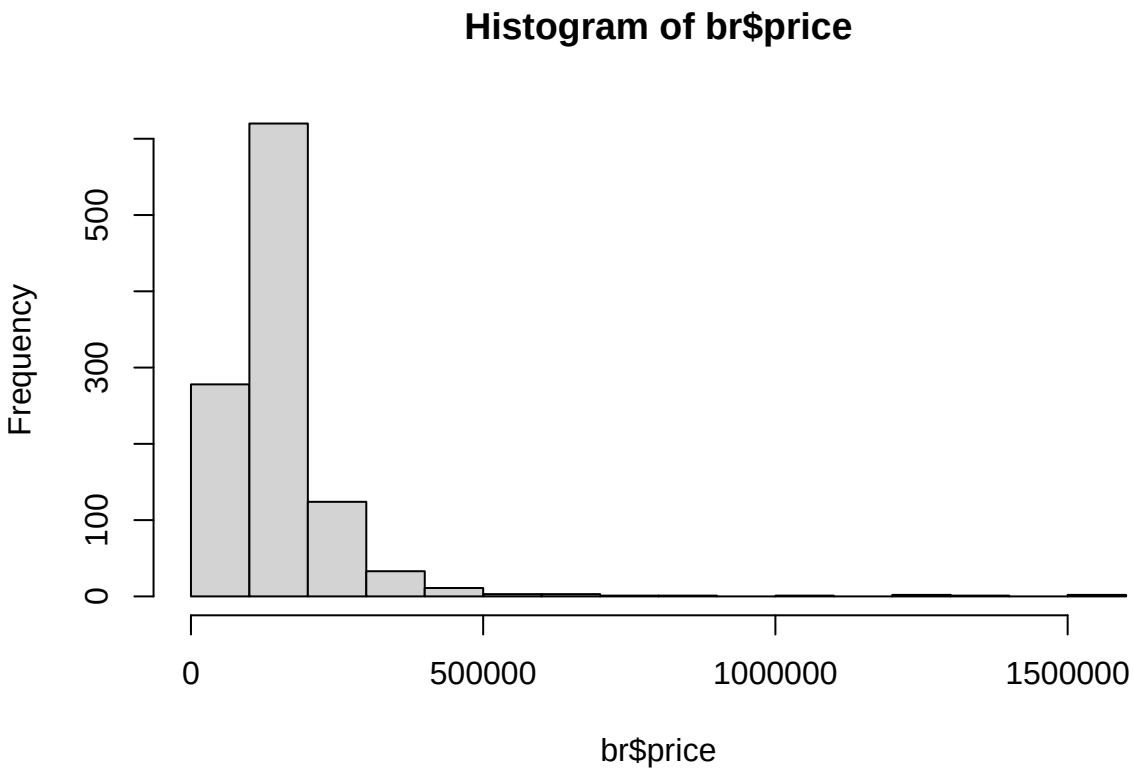
```
elasticity  #prints results
```

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

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

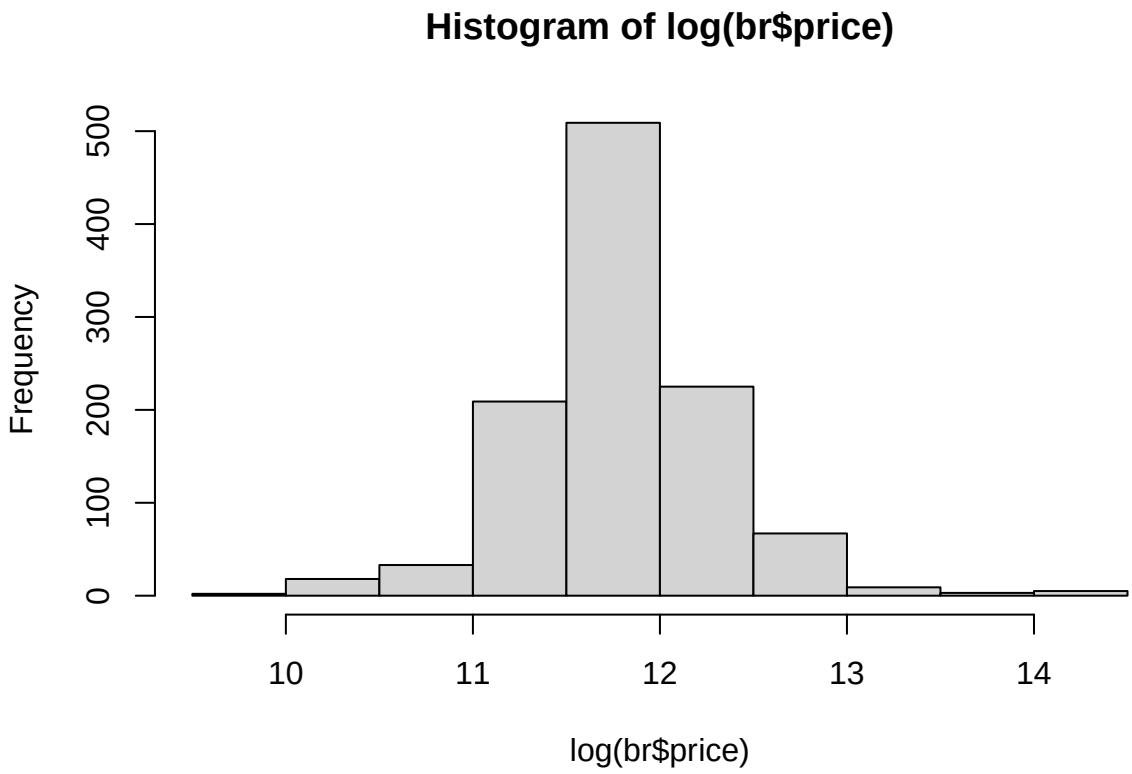


### 6.7.1 Verificando a variável dependente

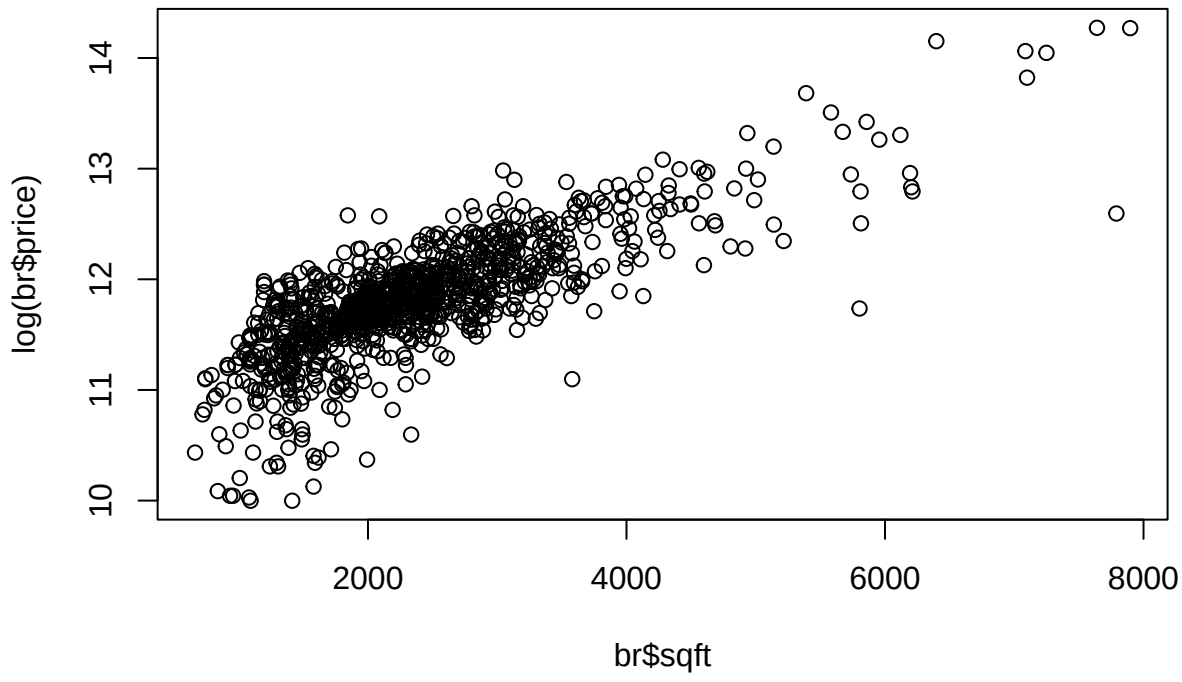



Transformação de variáveis

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




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



## 6.8 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 |
|---|---|---|---|---|---|
| 205.452 | 23.46 | 6 | 0 | 0 | 1 |
| 185.328 | 20.03 | 5 | 0 | 0 | 1 |
| 248.422 | 27.77 | 6 | 0 | 0 | 0 |
| 154.690 | 20.17 | 1 | 0 | 0 | 0 |
| 221.801 | 26.45 | 0 | 0 | 0 | 1 |
| 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
```

## 6.9 Monte Carlo

Vamos ver depois

# Capítulo 7

# Chapter 3 Interval Estimation and Hypothesis Testing

## 7.1 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 | 5.972053 | 14.44723 |

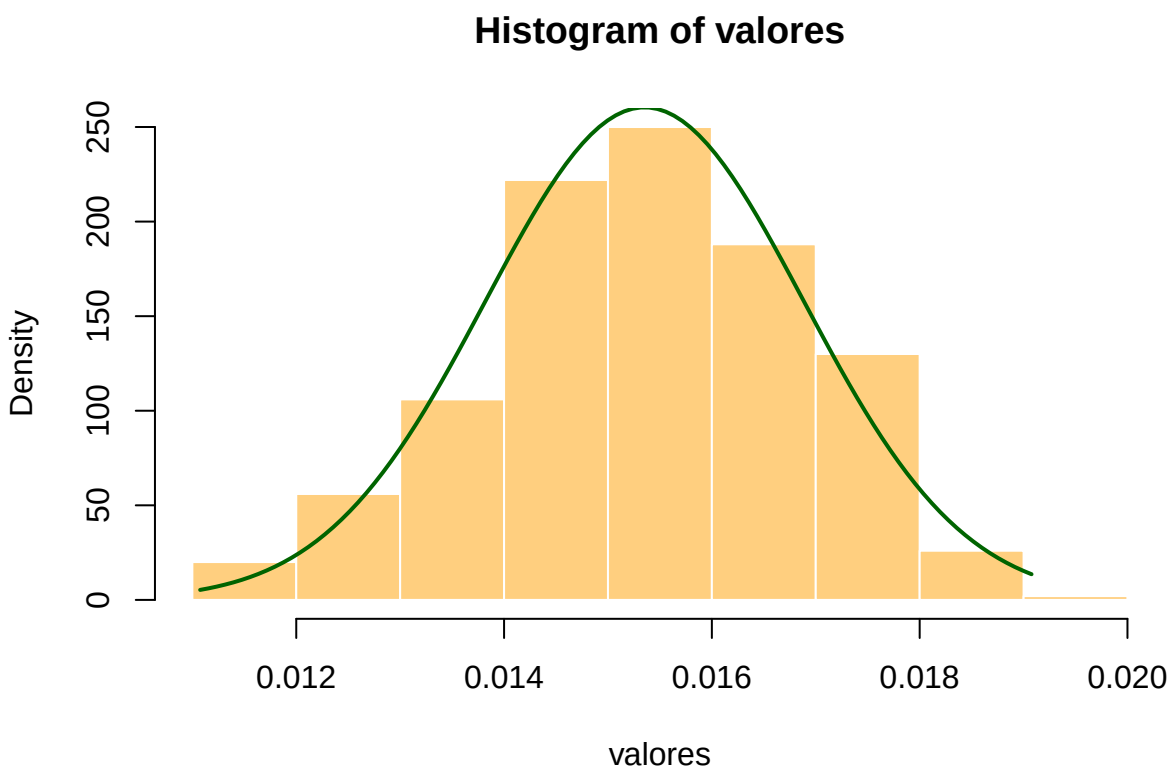
## 7.2 Bootstrap

Reamostragem

```
# Travo o gerador de números pseudo-aleatórios
set.seed(1)
# Número de simulações
N = 500  # coloquei 500 para rodar mais rápido, na prática usa-se 2000 ou mais
# Número de elementos na amostra
```

```
nrow(br) -> n
# 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])
}
```

```
# desenho um histograma com uma curva normal superimposta
# Este esquema de cores é somente um exemplo, adote um
# padrão para todos os gráficos para não ficar um
# 'carnaval'
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)
```




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

```
##  I(sqft^2)
## 0.01542130 0.01535264
```

```
# Teste de normalidade (veremos em um futuro próximo)
shapiro.test(sample(valores, min(500, length(valores))))
```

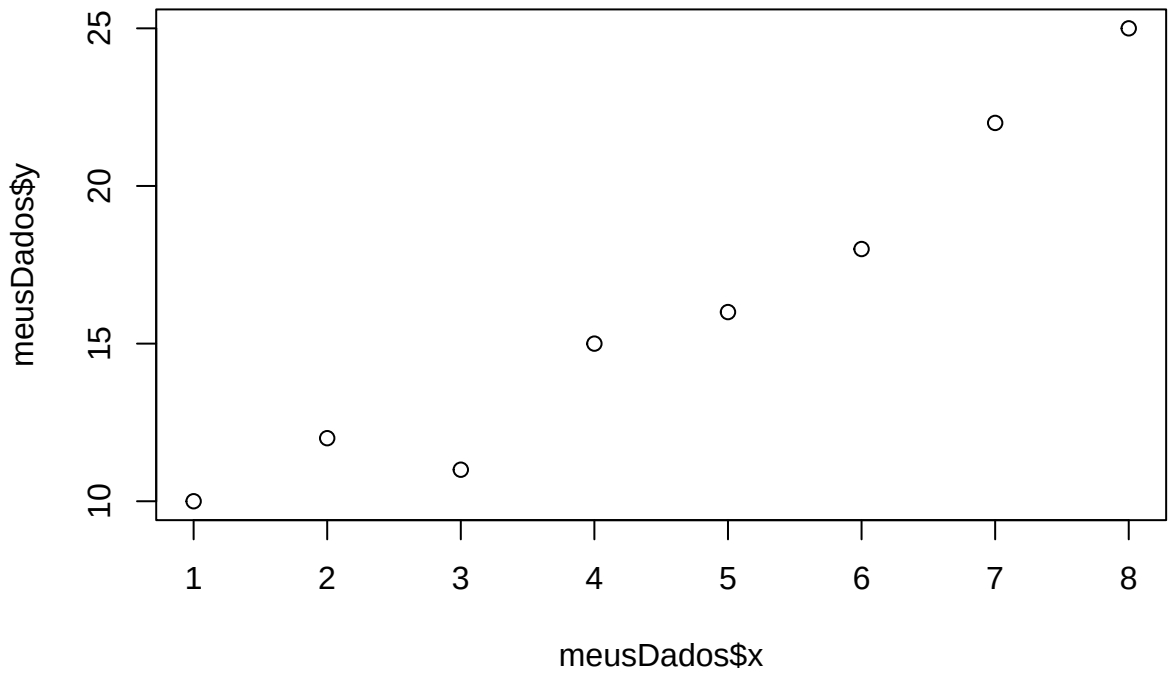
```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  sample(valores, min(500, length(valores)))
## W = 0.99399, p-value = 0.04537
```

# Capítulo 8

# Adendo - ler dados do EXCEL

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




```
lm(y ~ x, meusDados) -> m
m %>%
    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 9

# Adendo — Regressão OLS em Python

#“‘

```
#
# ATENÇÃO: para rodar este trecho do código é necessário ter o Python instalado e configurado
#
# o comando import do Python é similar ao comando library do R
# statsmodels.formula.api é a interface para os modelos estatísticos
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
```

```
# rodo o modelo OLS
model = smf.ols(formula="y~x", data=r.meusDados)
# inspeciono os resultados
print(model.fit().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:                Tue, 25 Mar 2025   Prob (F-statistic):           7.75e-05
## Time:                        16:22:12   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:41
##   return hypotest_fun_in(*args, **kwds)
```

#“‘