Universidade de Brasília

DEPARTAMENTO DE ESTATÍSTICA

16 julho 2023

Lista 3

Prof. Dr. Donald Matthew Pianto

Aluno: Bruno Gondim Toledo

Matrícula: 15/0167636

Estatística Computacional

1º/2023

```r
if (!require("pacman")) install.packages("pacman")
```

```
## Carregando pacotes exigidos: pacman
```

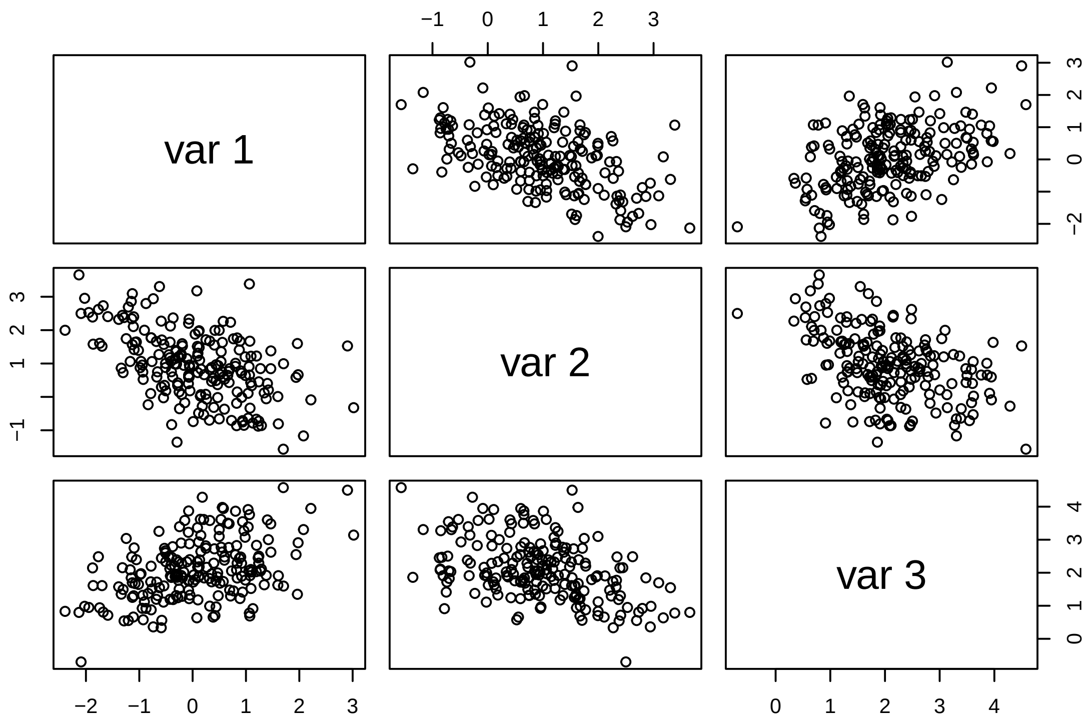
```r
p_load(knitr,tidyverse,car,mvtnorm,e1071,mlbench,caret)
seed <- 150167636
```

1) Rizzo – 3.14

```r
n <- 200
mu <- c(0,1,2)
set.seed(seed)
Sigma <- matrix(c(1, -.5, .5,
                  -.5,1,-.5,
                  .5,-.5,1),
                nrow = 3, ncol = 3)


r <- rmvnorm(n, mean = mu, sigma=Sigma, method="chol")

pairs(r)
```




```r
colMeans(r)
```

```
## [1] 0.06415776 0.95459648 2.08519934
```

```r
cor(r)
```

```
##            [,1]       [,2]       [,3]
## [1,]  1.0000000 -0.5112940  0.4503123
## [2,] -0.5112940  1.0000000 -0.4442163
## [3,]  0.4503123 -0.4442163  1.0000000
```

Notamos que tanto o vetor μ quanto a matriz Σ aparentam convergir para o verdadeiro parâmetro especificado.

2) Rizzo – 5.7

```
m <- seed
a <- - 12 + 6 * (exp(1) - 1)
set.seed(seed)
U <- runif(round(m)/1000)
T1 <- exp(U)
T2 <- exp(U) + a * (U - 1/2)

mean(T1)
```

```
## [1] 1.71553
```

```
mean(T2)
```

```
## [1] 1.718276
```

```
(var(T1) - var(T2)) / var(T1)
```

```
## [1] 0.9837556
```

Analisando os outputs, notamos que a redução da variância utilizando variáveis antitéticas foi bastante substantiva.

3) Validação Cruzada

```
data(Glass, package="mlbench")
index <- 1:nrow(Glass)
N <- trunc(length(index)/3)
set.seed(seed)
testindex <- sample(index, N)
testset <- Glass[testindex,]
trainset <- Glass[-testindex,]
svm.model <- svm(Type ~ ., data = trainset, cost = 100, gamma = 0.1)
svm.pred <- predict(svm.model, testset[,-10])
matriz_confusao <- table(pred = svm.pred, true = testset[,10])
(matriz_confusao <- as.matrix(matriz_confusao))
```

```
##     true
## pred  1  2  3  5  6  7
##    1 16  9  0  0  0  1
##    2  7 17  1  3  1  0
##    3  2  1  3  0  0  0
##    5  0  0  0  3  0  0
##    6  0  0  0  0  0  0
##    7  0  0  0  0  0  7
```

a)

```
ctrl <- trainControl(method = "cv", number = 10)

model <- train(Type ~ RI + Na + Mg + Al + Si + K + Ca + Ba + Fe, data = Glass, trControl = ctrl)
print(model)
```

```
## Random Forest
##
```

```
## 214 samples
##   9 predictor
##   6 classes: '1', '2', '3', '5', '6', '7'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 193, 192, 193, 193, 193, 193, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7994052  0.7208959
##   5     0.7571758  0.6648608
##   9     0.7476736  0.6546664
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
model$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 21.03%
## Confusion matrix:
##    1  2 3  5 6  7 class.error
## 1 61  7 2  0 0  0   0.1285714
## 2 11 59 1  2 2  1   0.2236842
## 3  8  3 6  0 0  0   0.6470588
## 5  0  2 0 10 0  1   0.2307692
## 6  0  2 0  0 7  0   0.2222222
## 7  1  2 0  0 0 26   0.1034483
```

Ou seja, o melhor valor para gamma aparenta ser entre 0 e 1; enquanto o erro do teste ficou na casa de 20%.

b)

```r
#predict(model)

svm(Type ~ ., data = Glass, cost = 100, gamma = c(0.01, 0.1 ,1))
```

```
##
## Call:
## svm(formula = Type ~ ., data = Glass, cost = 100, gamma = c(0.01,
##     0.1, 1))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  100
##
## Number of Support Vectors:  148
```

```r
svm(Type ~ ., data = Glass, cost = 100, gamma = c(0.001, 0.05 ,0.15))
```

```
##
## Call:
## svm(formula = Type ~ ., data = Glass, cost = 100, gamma = c(0.001,
##     0.05, 0.15))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  100
##
## Number of Support Vectors:  178
```

```r
# Ou seja, o melhor valor de gama:
svm(Type ~ ., data = Glass, cost = 100, gamma = 0.1)
```

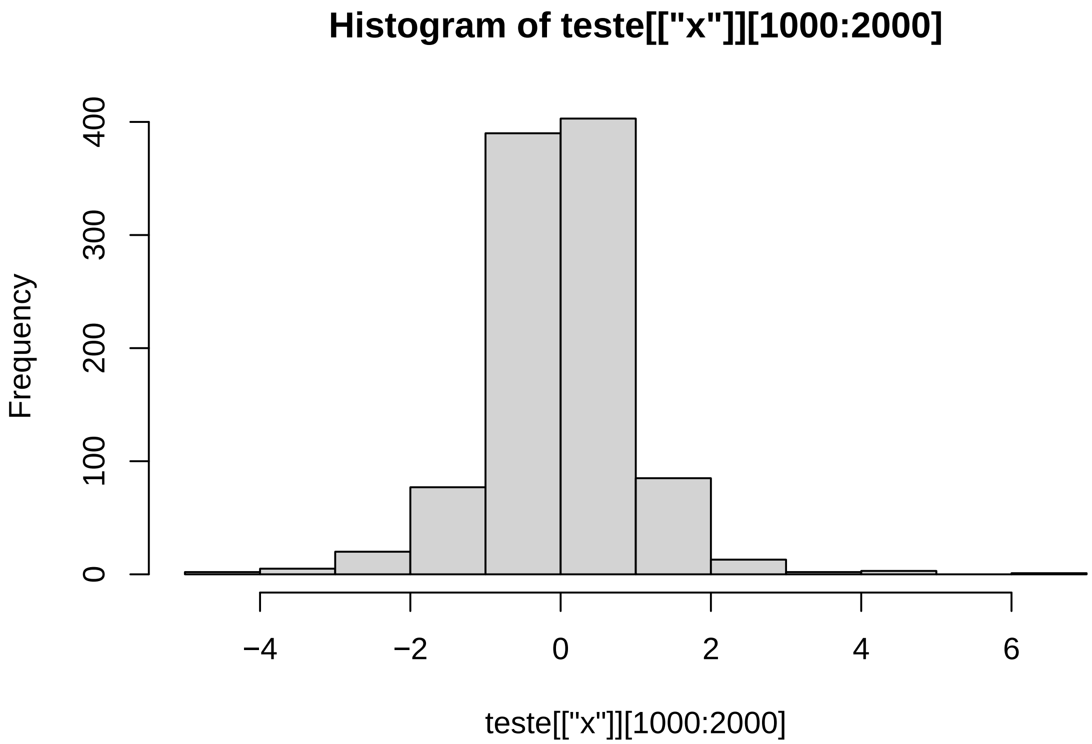
```
##
## Call:
## svm(formula = Type ~ ., data = Glass, cost = 100, gamma = 0.1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  100
##
## Number of Support Vectors:  136
```

4) Rizzo – 9.3 e 9.7

9.3:

```r
rw.Metropolis <- function(x0, N) {
  x <- numeric(N)
  x[1] <- x0
  u <- runif(N)
  k <- 0
  for (i in 2:N) {
    y <- rt(1, 1)
    if (u[i] <= (dcauchy(y) / dcauchy(x[i-1])))
      x[i] <- y else {
        x[i] <- x[i-1]
        k <- k + 1
      }
  }
  return(list(x=x, k=k))
}

teste <- rw.Metropolis(0,2000)
hist(teste[["x"]][1000:2000])
```



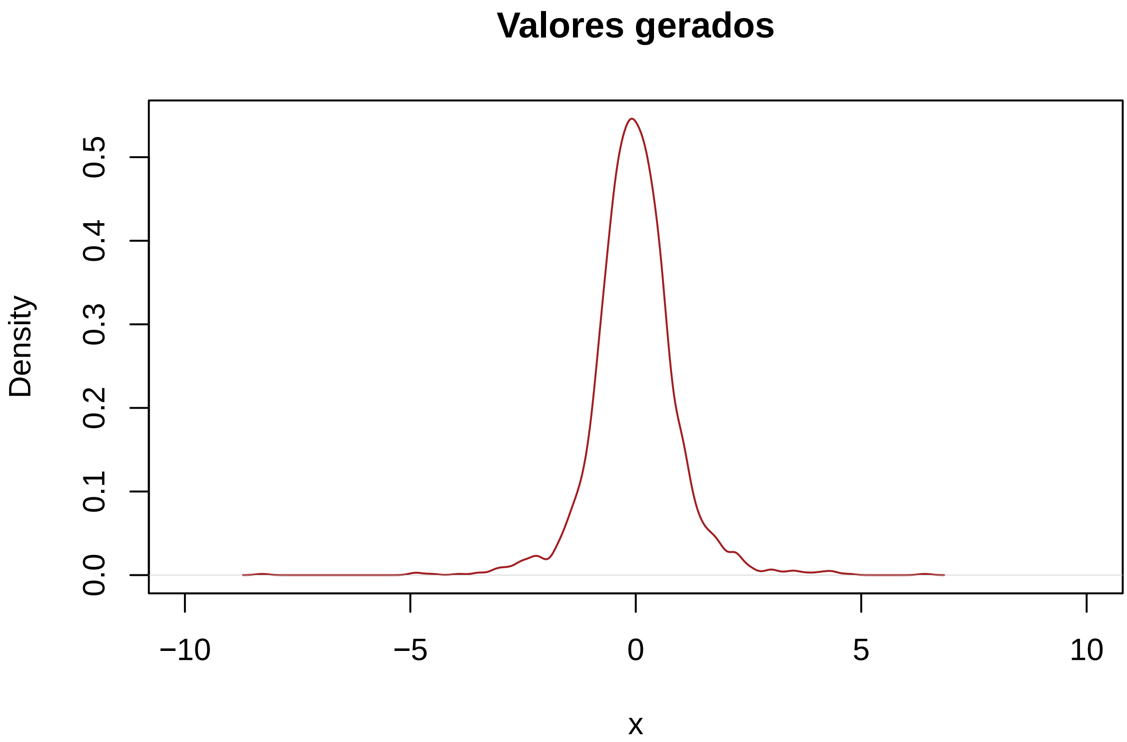

```r
quantile(teste[["x"]][1000:2000])
```

```
##          0%         25%         50%         75%        100%
## -4.90245448 -0.52696605  0.01728719  0.49204551  6.41100533
```

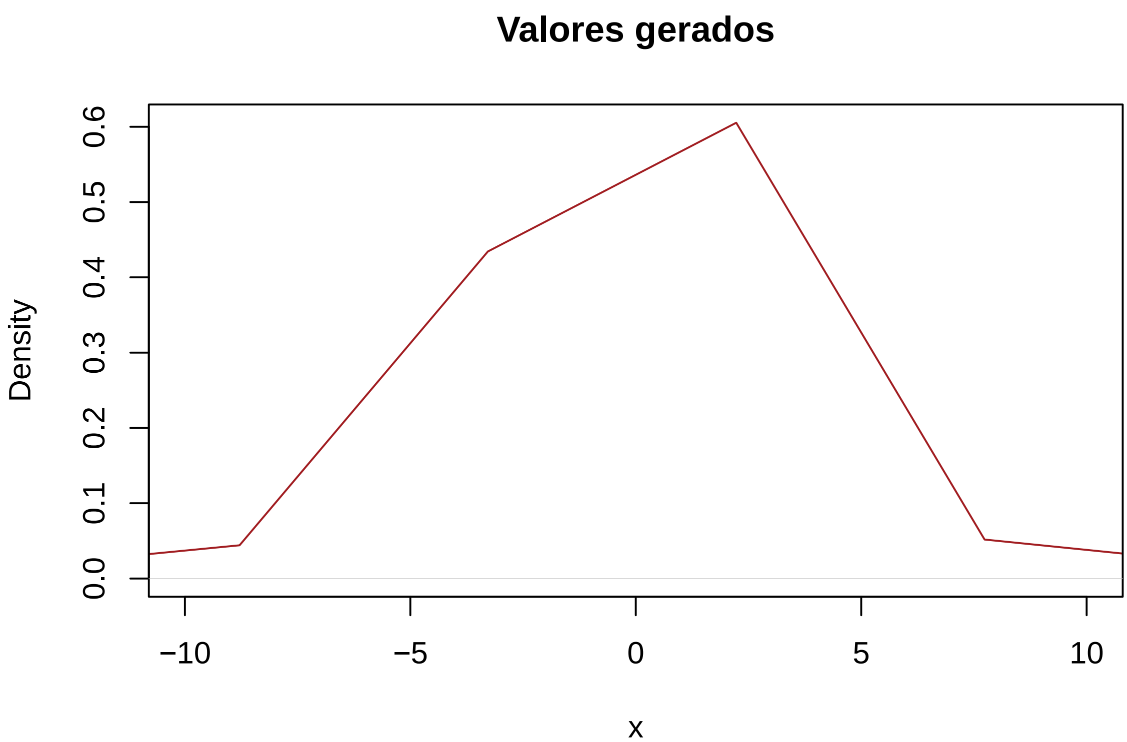
```r
cauchy <- rcauchy(1000)
quantile(cauchy)
```

```
##            0%           25%           50%           75%          100%
## -1.194745e+03 -8.830585e-01  1.060605e-02  9.093564e-01  1.960833e+02
```

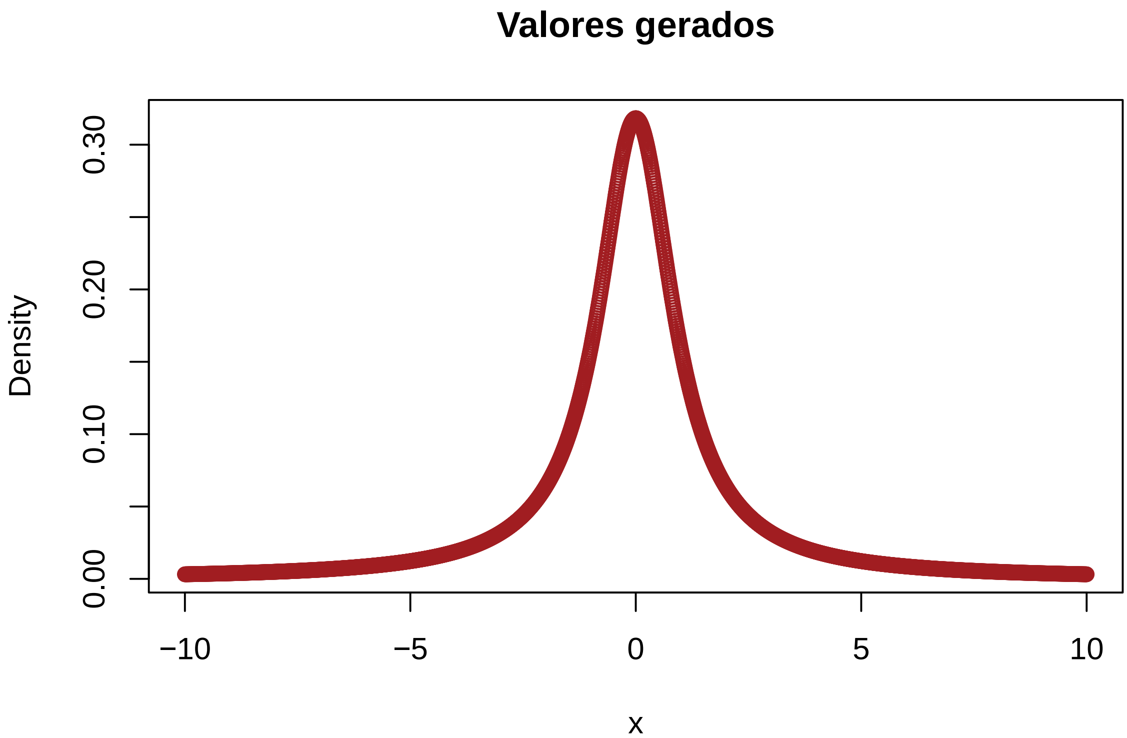
```r
plot(density(teste$x), main = "Valores gerados", xlab = "x", ylab = "Density", col = "#a11d21",
     xlim = c(-10,10))
```



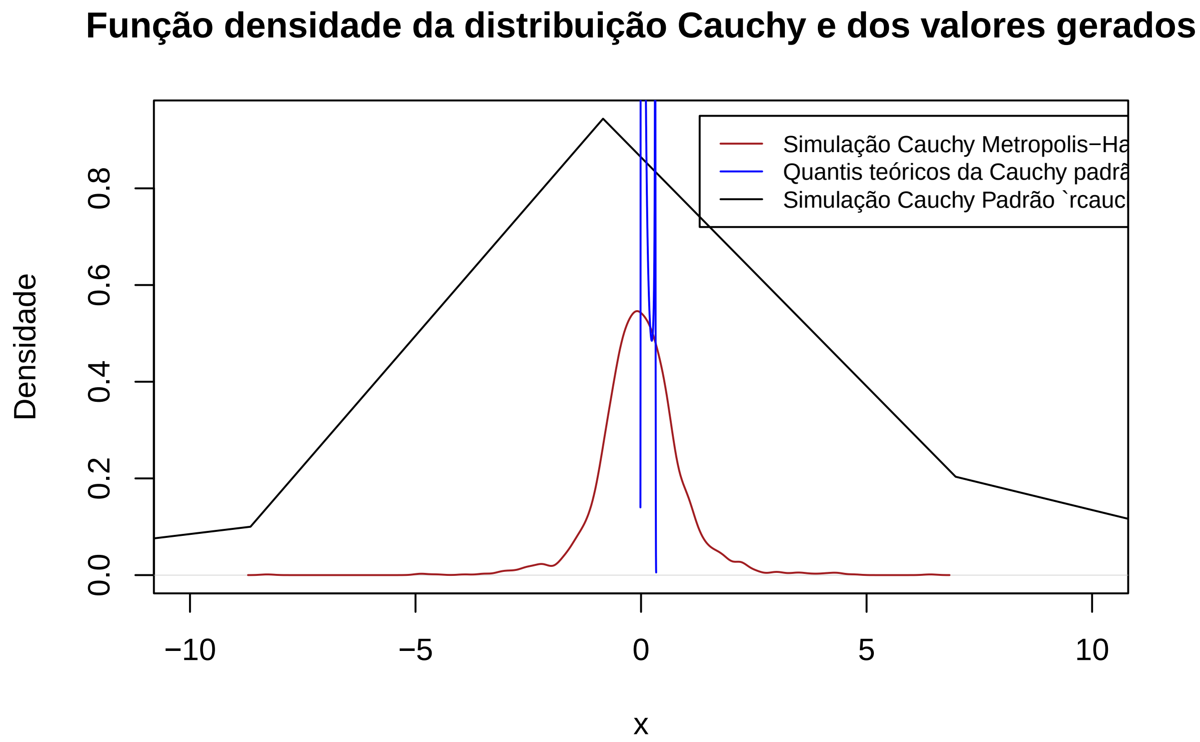

```
set.seed(seed)
plot(density(rcauchy(1000, location = 0, scale = 1)), main = "Valores gerados", xlab = "x", ylab = "
     xlim = c(-10,10))
```



```r
x <- seq(-10, 10, length.out = 2000)
plot(x,dcauchy(x, location = 0, scale = 1), main = "Valores gerados", xlab = "x", ylab = "Density",
     xlim = c(-10,10))
```




```r
set.seed(seed)
plot(density(rcauchy(2000, location = 0, scale = 1)), type = "l",
     xlab = "x", ylab = "Densidade",
     main = "Função densidade da distribuição Cauchy e dos valores gerados",
     xlim = c(-10,10))
lines(density(teste$x), col = "#a11d21")
lines(density(dcauchy(x, location = 0, scale = 1)), col = "blue")
legend( legend = c("Simulação Cauchy Metropolis-Hasting",
                   "Quantis teóricos da Cauchy padrão", "Simulação Cauchy Padrão `rcauchy`"),
        col = c("#a11d21","blue", "black"), lty = 1, cex = .75, x = 1.3, y = .95)
```



9.7:

```r
N <- 5000
burn <- 1000
X <- matrix(0, N, 2)
rho <- -.9
mu1 <- 0
mu2 <- 0
sigma1 <- 1
sigma2 <- 1
s1 <- sqrt(1-rho^2)*sigma1
s2 <- sqrt(1-rho^2)*sigma2
X[1, ] <- c(mu1, mu2)
for (i in 2:N) {
  x2 <- X[i-1, 2]
  m1 <- mu1 + rho * (x2 - mu2) * sigma1/sigma2
  X[i, 1] <- rnorm(1, m1, s1)
  x1 <- X[i, 1]
  m2 <- mu2 + rho * (x1 - mu1) * sigma2/sigma1
  X[i, 2] <- rnorm(1, m2, s2)
}
b <- burn + 1
X <- X[b:N, ]

df <- data.frame(X)
fit <- lm(df$X2 ~ df$X1)
summary(fit)
```

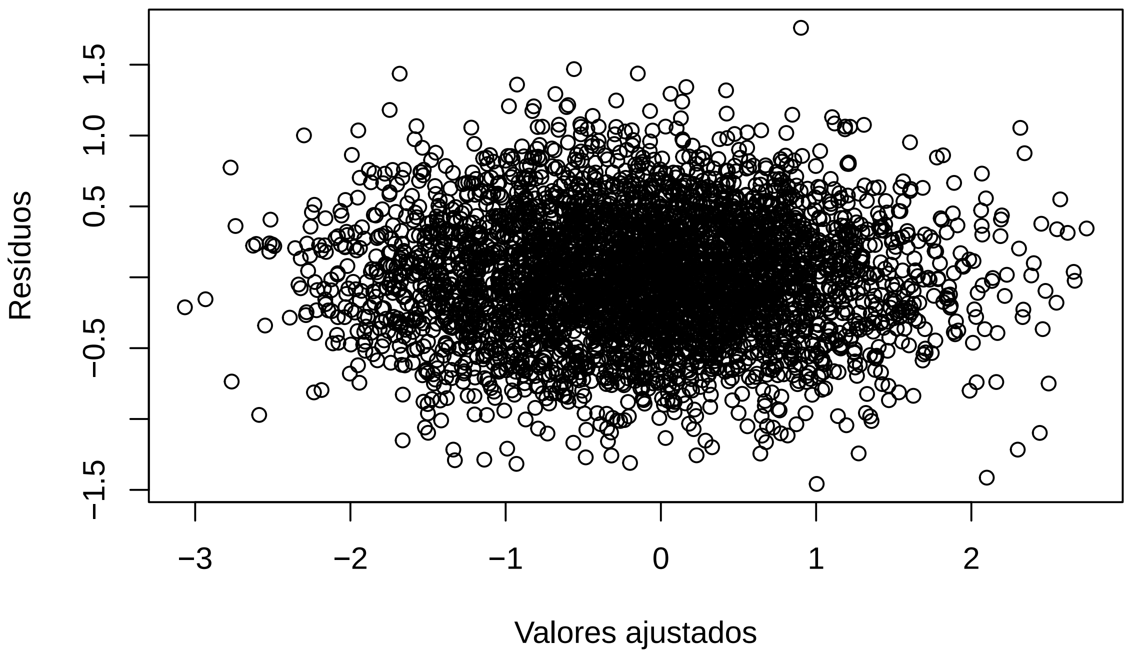
```
##
## Call:
## lm(formula = df$X2 ~ df$X1)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.45791 -0.28646 -0.00584  0.29403  1.76016
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -0.009756   0.006908   -1.412    0.158
## df$X1       -0.899590   0.006911 -130.163   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4333 on 3998 degrees of freedom
## Multiple R-squared:  0.8091, Adjusted R-squared:  0.809
## F-statistic: 1.694e+04 on 1 and 3998 DF,  p-value: < 2.2e-16
```

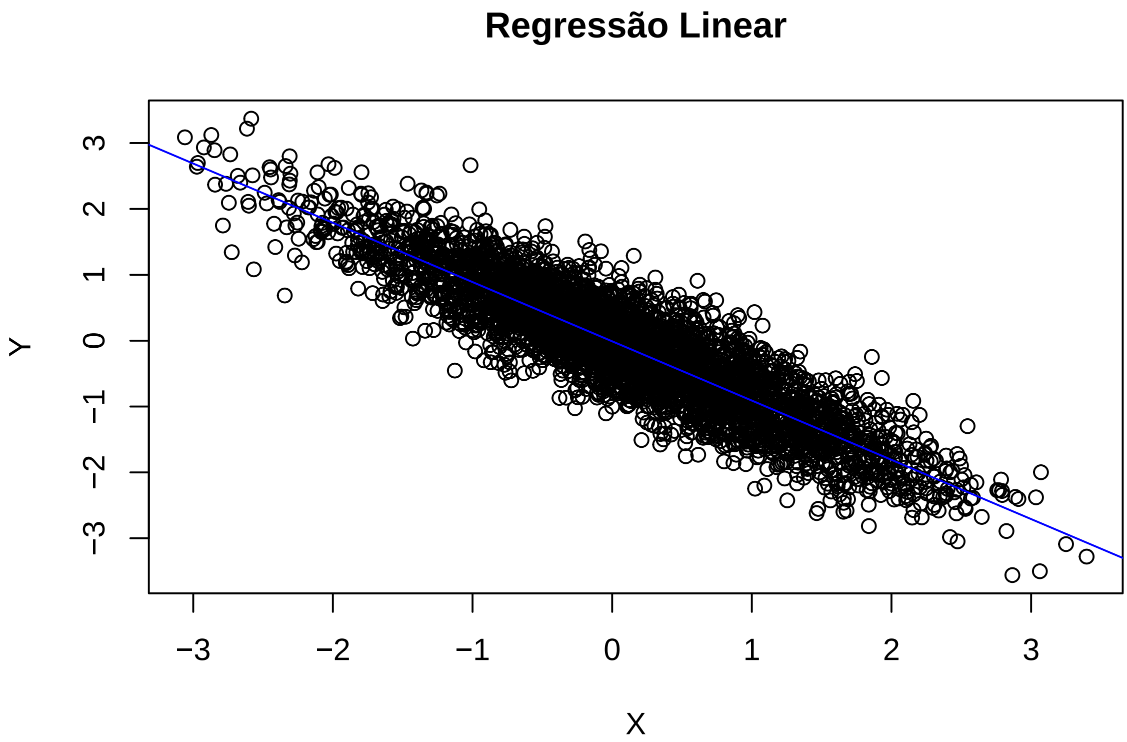
```r
shapiro.test(fit$residuals) # normalidade ok
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit$residuals
## W = 0.99966, p-value = 0.7625
```

```r
plot(fit$fitted.values, fit$residuals,
     xlab = "Valores ajustados", ylab = "Resíduos") # variância ok
```




```r
# Gráfico de dispersão + linha de mínimos quadrados (regressão)
plot(df$X1, df$X2, main = "Regressão Linear", xlab = "X", ylab = "Y")
abline(fit, col = "blue")
```


Regressão Linear
Y
X
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3