

# Atividade nº 04 - Importação de Dados e Gráficos

Lucas Henrique Nogueira

28/05/2024

## Lista de Exercícios

### 1) Importação online e local de data frames

- Conjunto de dados utilizados nessa primeira parte da lista foi extraído do link abaixo:

Census at School

**Contexto:** O conjunto de dados “Census at School-500.csv” provém do projeto internacional Census at School. Este projeto educacional global tem como objetivo envolver alunos em atividades estatísticas usando dados reais coletados por eles mesmos. Os alunos respondem a uma série de perguntas sobre diversos aspectos de suas vidas, como características físicas, hábitos, preferências e atividades. O objetivo é tornar o aprendizado de estatística mais interessante e relevante, utilizando dados coletados pelos próprios alunos.

- a) Utilizando as funções `read.csv()` e `read.table()` para importar os dados diretamente da URL:

```
url <- "https://www.stat.auckland.ac.nz/~wild/d2i/FutureLearn/Census.at.School.500_ages9-15.csv"
```

```
census_csv <- read.csv(url)
```

```
dim(census_csv)      # Dimensão do conjunto (Linha X Coluna)
```

```
## [1] 483  10
```

```
names(census_csv)    # Nomes das variáveis
```

```
## [1] "cellsource" "rightfoot" "travel"      "getlunch"  "height"
## [6] "gender"     "age"         "year"       "armspan"   "cellcost"
```

```
census_table <- read.table(url, sep = ",", header = TRUE)
head(census_table)
```

```
##   cellsource rightfoot travel getlunch height gender age year armspan cellcost
## 1   pocket      20    walk    home    152   male  12   7    150      30
## 2   parent      25   other  friend    153 female  11   6    152      50
## 3   parent      21  motor   home    137   male  10   6    132      55
## 4   pocket      20    walk    home    115   male   9   5    130      60
## 5   pocket      23   other   home    165 female  14  10    160      20
## 6   parent      19  motor   home    137 female  11   7     50      50
```

- b) Utilizando a função `file.choose()` para fazer uma importação local dos dados:

```
census_local <- read.csv(file.choose()) # Importando o arquivo local do diretório do PC.
```

```
dim(census_local) # Dimensão do conjunto (Linha X Coluna)
```

```
## [1] 483 10
```

```
names(census_local) # Nomes das variáveis
```

```
## [1] "cellsource" "rightfoot" "travel" "getlunch" "height"
## [6] "gender" "age" "year" "armspan" "cellcost"
```

```
census_table2 <- read.table(file.choose(), sep = ",", header = TRUE)
head(census_table2)
```

```
##   cellsource rightfoot travel getlunch height gender age year armspan cellcost
## 1   pocket      20   walk    home    152   male  12   7    150      30
## 2   parent      25  other  friend    153 female  11   6    152      50
## 3   parent      21 motor   home    137   male  10   6    132      55
## 4   pocket      20   walk    home    115   male   9   5    130      60
## 5   pocket      23  other   home    165 female  14  10    160      20
## 6   parent      19 motor   home    137 female  11   7     50      50
```

**Observação:** A função `file.choose()` oferece uma maneira simples e direta para os usuários selecionarem um arquivo no sistema, navegando através das pastas do computador de forma interativa. Porém como a seleção do arquivo é feita manualmente pelo usuário, não é possível automatizar esse processo em scripts ou tarefas programadas, o que pode limitar a eficiência em cenários de automação, além de ser mais adequada para uso em ambientes interativos, como o RStudio.

## 2) Operações com o conjunto de dados `skulls{ade4}`.

```
data(skulls, package = "ade4")
head(skulls)
```

```
##      V1  V2  V3 V4
## 1 131 138  89 49
## 2 125 131  92 48
## 3 131 132  99 50
## 4 119 132  96 44
## 5 136 143 100 54
## 6 138 137  89 56
```

- a) Manipulando os dados do conjunto de dados:
  - Renomeando as variáveis: V1 por ACr, V2 por BBr, V3 por BA1 e V4 por ANs.

```
names(skulls) <- c("ACr", "BBr", "BA1", "ANs")
head(skulls)
```

```
##      ACr BBr BA1 ANs
## 1 131 138  89  49
## 2 125 131  92  48
## 3 131 132  99  50
## 4 119 132  96  44
## 5 136 143 100  54
## 6 138 137  89  56
```

- Criando uma variável categórica 'período':

```
período <- factor(rep(1:5, each = 30), labels = c("período pré-dinástico primitivo",
                                                    "período pré-dinástico antigo",
                                                    "12ª e 13ª dinastias",
                                                    "período Ptolemaico",
                                                    "período Romano"))

skulls$período <- período
head(skulls)
```

```
##      ACr BBr BA1 ANs      período
## 1 131 138  89  49 período pré-dinástico primitivo
## 2 125 131  92  48 período pré-dinástico primitivo
## 3 131 132  99  50 período pré-dinástico primitivo
## 4 119 132  96  44 período pré-dinástico primitivo
## 5 136 143 100  54 período pré-dinástico primitivo
## 6 138 137  89  56 período pré-dinástico primitivo
```

- Criando uma variável quantidade idade:

```
idade <- rep(c(-4000, -3300, -1850, -200, 150), each = 30)

skulls$idade <- idade
head(skulls)
```

```
##   ACr BBr BA1 ANs                periodo idade
## 1 131 138 89 49 período pré-dinástico primitivo -4000
## 2 125 131 92 48 período pré-dinástico primitivo -4000
## 3 131 132 99 50 período pré-dinástico primitivo -4000
## 4 119 132 96 44 período pré-dinástico primitivo -4000
## 5 136 143 100 54 período pré-dinástico primitivo -4000
## 6 138 137 89 56 período pré-dinástico primitivo -4000
```

- b) Ampliando a análise exploratória desses dados:
  - Calculando a média de cada uma das medidas por período:

```
medias_periodo <- aggregate(cbind(ACr, BBr, BA1, ANs) ~ periodo, data = skulls, mean)
rownames(medias_periodo) <- c("Primitivo", "Antigo", "Dinastias", "Ptolemaico", "Romano")

medias_periodo      # Matriz (tabela)
```

```
##                periodo      ACr      BBr      BA1      ANs
## Primitivo período pré-dinástico primitivo 131.3667 133.6000 99.16667 50.53333
## Antigo      período pré-dinástico antigo 132.3667 132.7000 99.06667 50.23333
## Dinastias      12ª e 13ª dinastias 134.4667 133.8000 96.03333 50.56667
## Ptolemaico      período Ptolemaico 135.5000 132.3000 94.53333 51.96667
## Romano                período Romano 136.1667 130.3333 93.50000 51.36667
```

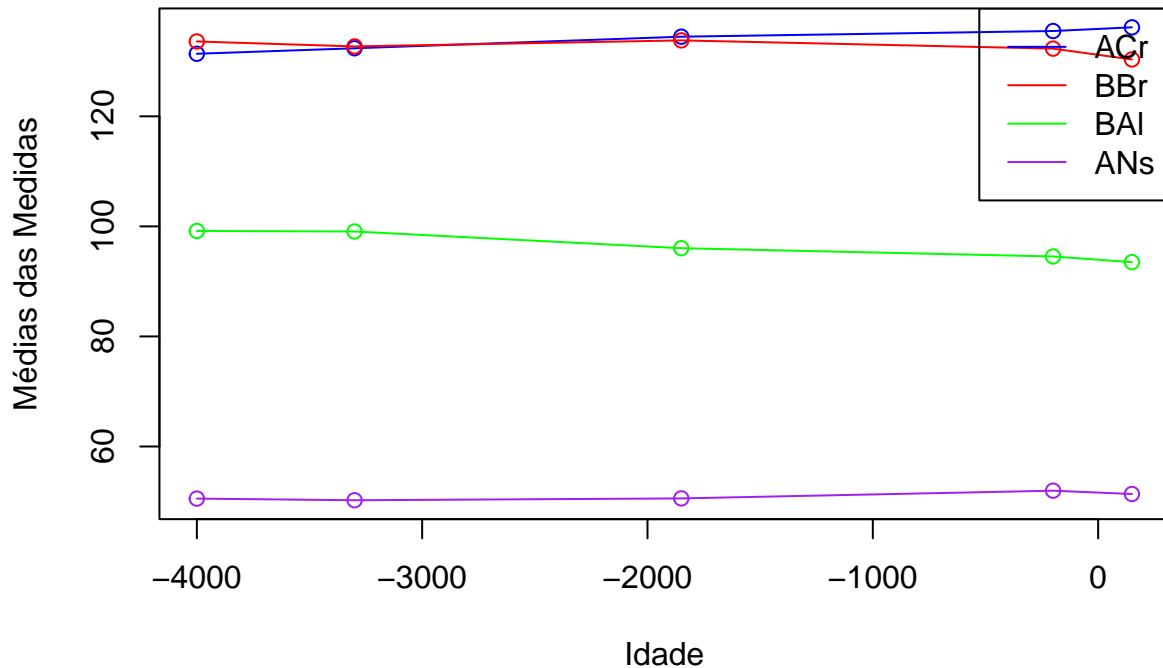
- Gráfico de linhas das médias de cada uma das medidas por idade:

```
medias_idade <- aggregate(cbind(ACr, BBr, BA1, ANs) ~ idade, data = skulls, mean)

plot(medias_idade$idade, medias_idade$ACr, type = "o", col = "blue",
     ylim = range(medias_idade[,2:5]), xlab = "Idade", ylab = "Médias das Medidas",
     main = "Médias das Medidas ao longo do Tempo")
lines(medias_idade$idade, medias_idade$BBr, type = "o", col = "red")
lines(medias_idade$idade, medias_idade$BA1, type = "o", col = "green")
lines(medias_idade$idade, medias_idade$ANs, type = "o", col = "purple")

legend("topright", legend = c("ACr", "BBr", "BA1", "ANs"),
     col = c("blue", "red", "green", "purple"), lty = 1)
```

## Médias das Medidas ao longo do Tempo



- c) Conclusão da análise do gráfico:

Visualizando o gráfico das médias das medidas dos crânios ao longo do tempo, podemos observar as seguintes tendências:

### 1. ACr (Amplitude máxima do crânio):

- A linha correspondente à ACr mostra uma variação ao longo dos diferentes períodos. Podemos ver que, do período pré-dinástico primitivo para os períodos subsequentes, há uma ligeira tendência de aumento e depois uma estabilização.

### 2. BBr (Altura basilobregmática do crânio):

- A altura basilobregmática (BBr) também apresenta mudanças ao longo do tempo, com variações menos pronunciadas comparadas à ACr, mas ainda assim com algumas flutuações.

### 3. BAI (Comprimento basiloalveolar do crânio):

- O comprimento basiloalveolar (BAI) dos crânios parece ter uma tendência de aumento gradual ao longo dos períodos históricos, indicando uma possível mudança nas características faciais das populações.

### 4. ANs (Altura nasal do crânio):

- A altura nasal (ANs) apresenta variações significativas, com um aumento visível em alguns períodos e uma leve queda em outros. Esse padrão sugere que as características nasais dos crânios sofreram mudanças ao longo do tempo.

Com base nas observações feitas a partir do gráfico, podemos levantar algumas conjecturas sobre as possíveis razões para as mudanças nas medidas dos crânios ao longo do tempo:

### 1. Influências Ambientais e Culturais:

- As variações nas medidas dos crânios podem ser atribuídas a mudanças nos ambientes e práticas culturais das populações ao longo dos diferentes períodos históricos. Por exemplo, alterações na dieta, práticas de trabalho e condições de vida podem ter impactado o desenvolvimento físico das pessoas.

### 2. Evolução Biológica:

- As mudanças nas características dos crânios também podem refletir um processo de evolução biológica, onde diferentes pressões seletivas ao longo do tempo resultaram em alterações nas características físicas das populações.

### 3. Migrações e Misturas Populacionais:

- A chegada de novas populações e a mistura genética com grupos locais podem ter introduzido novas características físicas, resultando em mudanças observáveis nas medidas dos crânios.

## 3) Operações com os comandos `scan()` e `lower.tri()`.

```
dados <- scan("E9-14.DAT")
dados
```

```
## [1] 1.000 0.505 1.000 0.569 0.422 1.000 0.602 0.467 0.926 1.000 0.621 0.482
## [13] 0.877 0.874 1.000 0.603 0.450 0.878 0.894 0.937 1.000
```

```
n <- 6
R <- matrix(0, n, n)
R[lower.tri(R, diag = TRUE)] <- dados
R <- R + t(R) - diag(diag(R))
R
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,] 1.000 0.505 1.000 0.569 0.422 1.000
## [2,] 0.505 0.602 0.467 0.926 1.000 0.621
## [3,] 1.000 0.467 0.482 0.877 0.874 1.000
## [4,] 0.569 0.926 0.877 0.603 0.450 0.878
## [5,] 0.422 1.000 0.874 0.450 0.894 0.937
## [6,] 1.000 0.621 1.000 0.878 0.937 1.000
```

- a) Calculando o traço e o determinante da matriz de correlações R:

```
traco <- sum(diag(R))
traco                                     # Traço da matriz R.
```

```
## [1] 4.581
```

```
determinante <- det(R)
determinante      # Determinante da matriz R.
```

```
## [1] -0.003203283
```

- b) Calculando os autovetores com o comando `eigen()` e calculando a proporção com o traço da matriz R:

```
eigen_ <- eigen(R)

autovetores <- eigen_$vectors
autovetores      # Autovetores da matriz R.
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] -0.4015955 -0.6371302  0.05483421  0.59510400  0.1226489  0.2461462
## [2,] -0.3585478  0.4943056 -0.32820518  0.44825273  0.3108216 -0.4710056
## [3,] -0.4165466 -0.1895583  0.08716712 -0.12999925 -0.6985779 -0.5273021
## [4,] -0.3783777  0.0773746 -0.71295188 -0.26526987 -0.1989951  0.4822613
## [5,] -0.4033048  0.5265168  0.57722480  0.09536667 -0.1747784  0.4327753
## [6,] -0.4804467 -0.1748942  0.20046849 -0.59038802  0.5746212 -0.1401708
```

```
proporcao_autovetores <- autovetores / sum(autovetores)
proporcao_autovetores      # Proporção de cada autovetor em relação ao traço.
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 0.1707350  0.27087069 -0.02331232 -0.25300361 -0.05214318 -0.10464703
## [2,] 0.1524337 -0.21014999  0.13953376 -0.19057099 -0.13214328  0.20024417
## [3,] 0.1770914  0.08058917 -0.03705839  0.05526812  0.29699472  0.22417819
## [4,] 0.1608642 -0.03289518  0.30310568  0.11277732  0.08460113 -0.20502947
## [5,] 0.1714618 -0.22384434 -0.24540241 -0.04054436  0.07430559 -0.18399090
## [6,] 0.2042580  0.07435483 -0.08522754  0.25099865 -0.24429551  0.05959246
```

- c) Comparando o traço da matriz R com a soma de seus autovalores e o determinante da matriz R com o produto de seus autovalores:

```
autovalores <- eigen_$values
autovalores      # Autovalores da matriz R.
```

```
## [1]  4.64434262  0.76239062  0.26708573 -0.01614753 -0.25542499 -0.82124645
```

```
soma_autovalores <- sum(autovalores)

soma_autovalores      # Soma dos autovalores da matriz R.
```

```
## [1] 4.581
```

```
traco      # Traço.
```

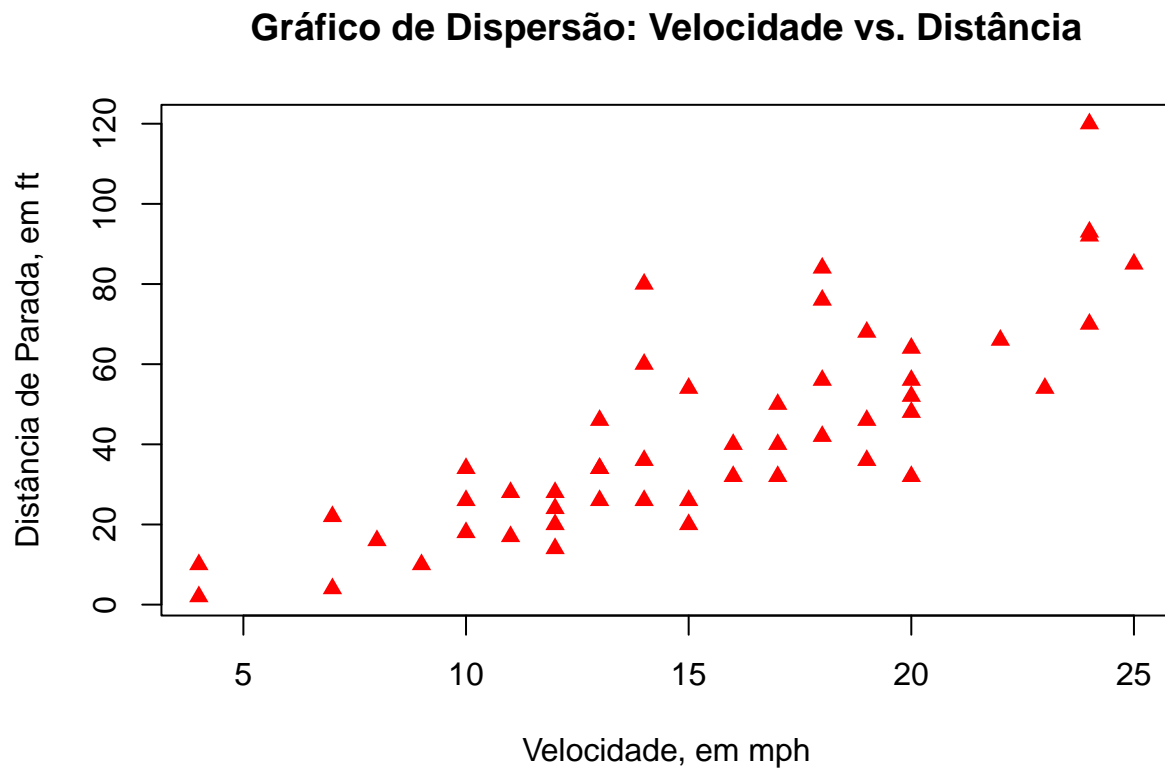
```
## [1] 4.581
```

#### 4) Conjunto de dados `cars{datasets}`.

```
data(cars)
```

- a), b) e c) Construindo um gráfico de dispersão de `speed` por `dist`:

```
plot(cars$speed, cars$dist, main = "Gráfico de Dispersão: Velocidade vs. Distância",  
      xlab = "Velocidade, em mph", ylab = "Distância de Parada, em ft", col = "red", pch = 17)
```





- d), e), f), g) e h) Construindo um gráfico e aplicando os modelos linear e quadrático:

```
modelo.linear <- lm(dist ~ speed, data = cars)

plot(cars$speed, cars$dist,
     xlab = "Velocidade, em mph",
     ylab = "Distância de parada, em ft",
     main = "Ajuste Linear e Quadrático: Velocidade vs. Distância de parada")

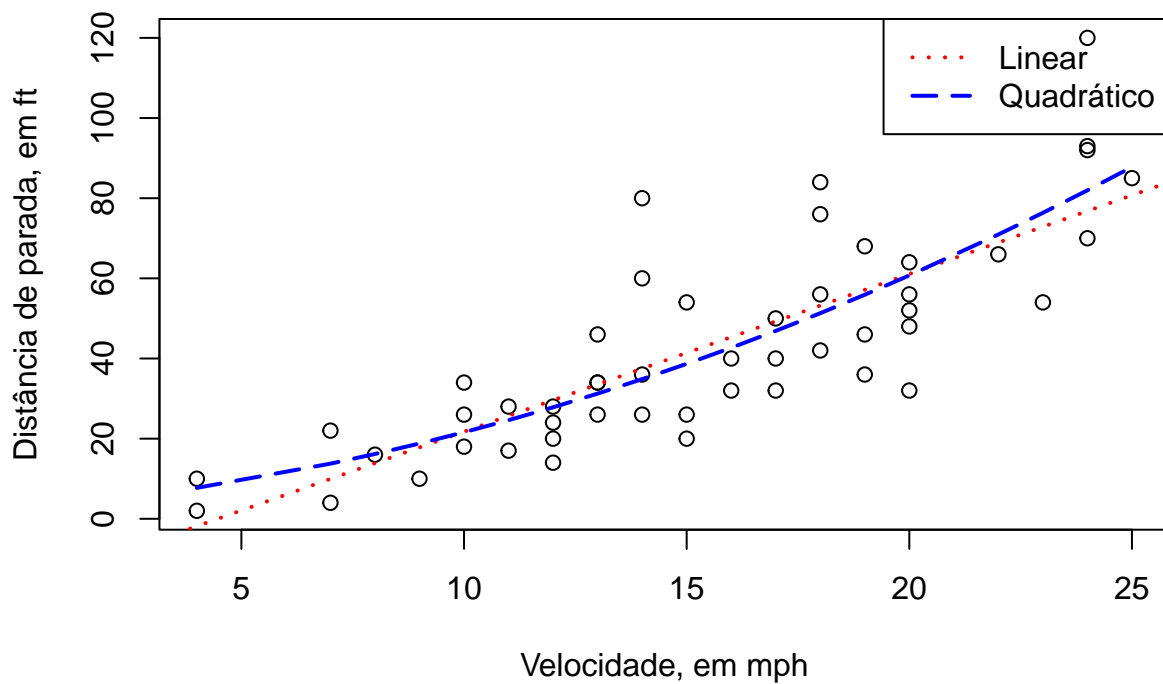
abline(modelo.linear, col = "red", lty = "dotted", lwd = 2)

modelo.quadratico <- lm(dist ~ speed + I(speed^2), data = cars)

lines(cars$speed, predict(modelo.quadratico), col = "blue", lty = "longdash", lwd = 2)

legend("topright", legend = c("Linear", "Quadrático"),
      col = c("red", "blue"), lty = c("dotted", "longdash"), lwd = 2)
```

## Ajuste Linear e Quadrático: Velocidade vs. Distância de parada



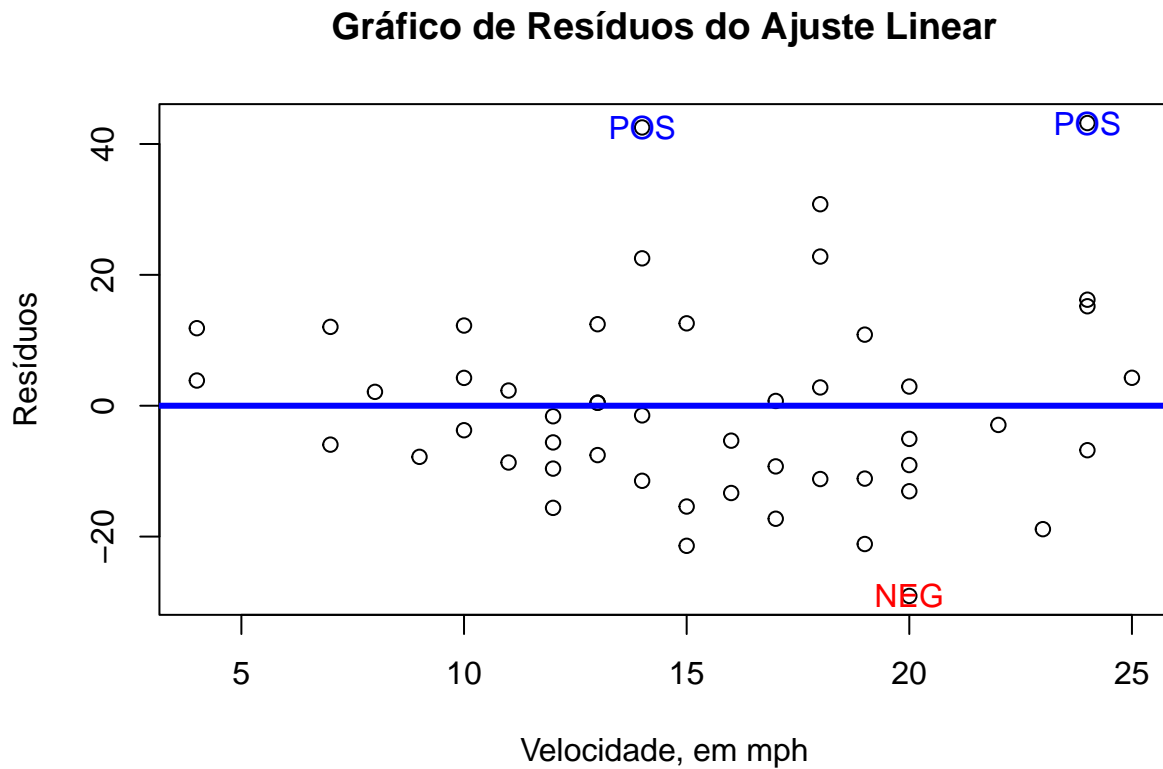
- i), j), k) e l) Construindo o gráfico de resíduos:

```
plot(cars$speed, modelo.linear$residuals, xlab = "Velocidade, em mph",
     ylab = "Resíduos", main = "Gráfico de Resíduos do Ajuste Linear")

abline(h = 0, col = "blue", lwd = 3)

extremos_positivos <- order(modelo.linear$residuals, decreasing = TRUE)[1:2]
text(cars$speed[extremos_positivos], modelo.linear$residuals[extremos_positivos],
     labels = "POS", col = "blue")

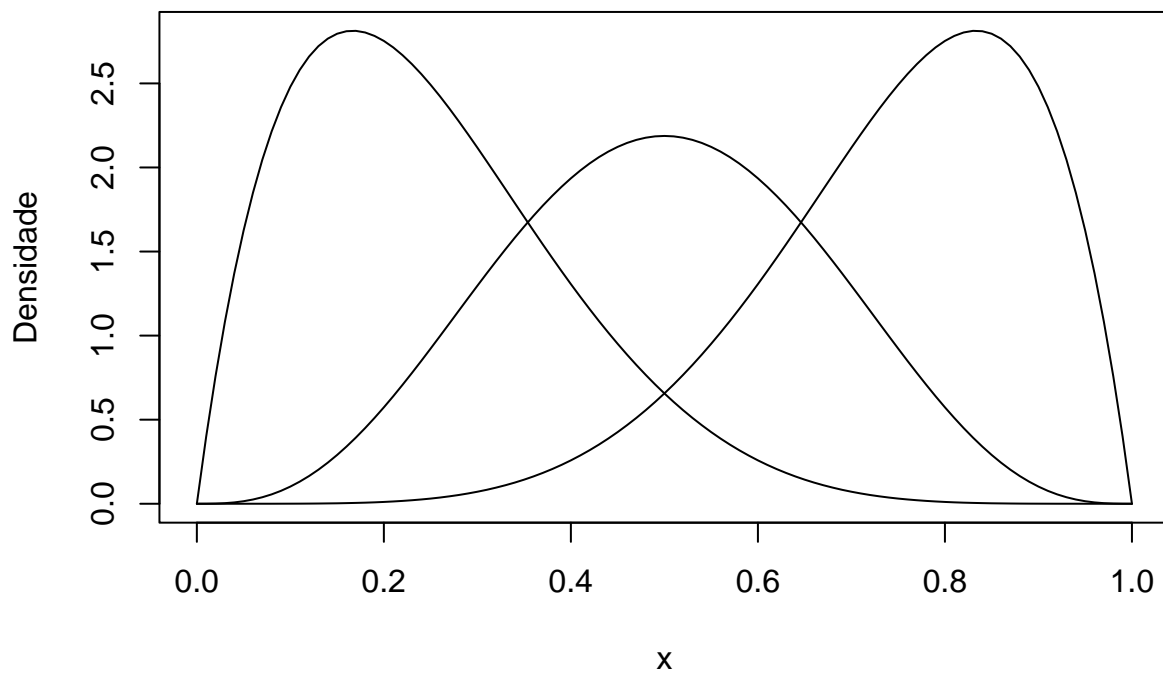
extremo_negativo <- which.min(modelo.linear$residuals)
text(cars$speed[extremo_negativo], modelo.linear$residuals[extremo_negativo],
     labels = "NEG", col = "red")
```



## 5) Gráfico com função de probabilidade

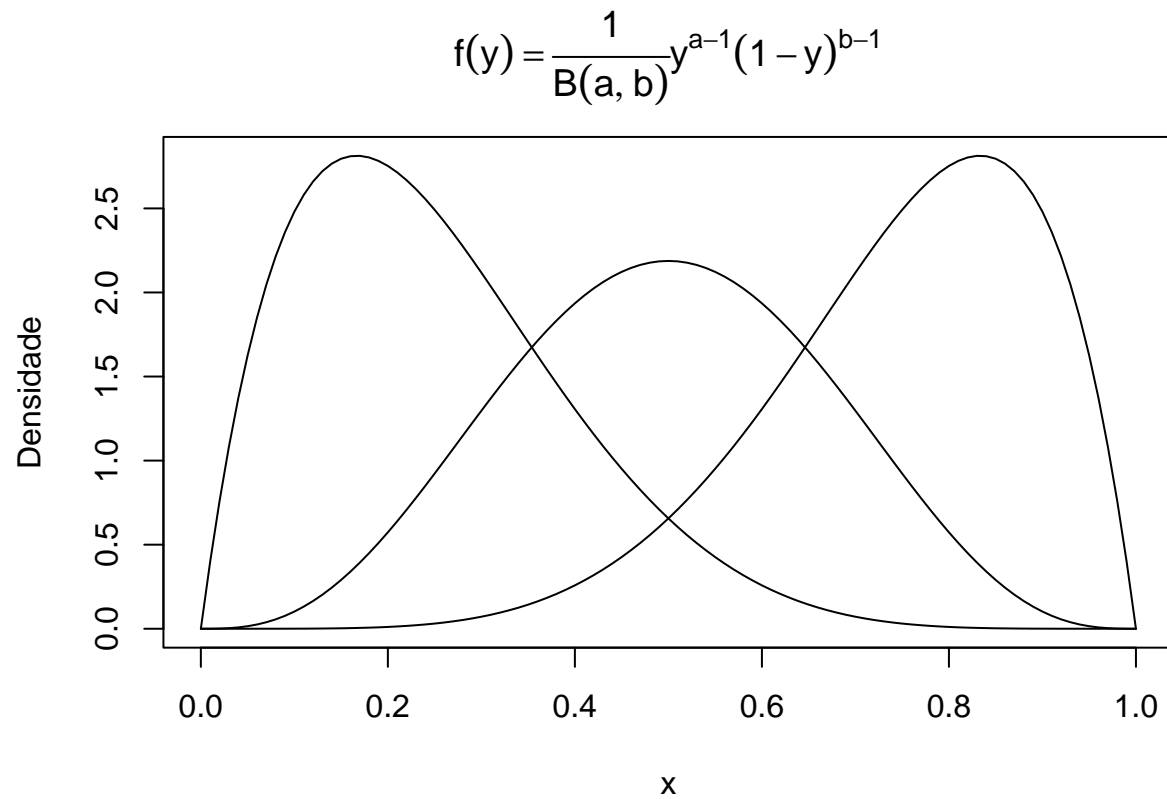
- a) Aplicando as função `curve()` para apresentar as densidades  $\text{Beta}(2,6)$ ,  $\text{Beta}(4,4)$  e  $\text{Beta}(6,2)$ :

```
curve(dbeta(x, 2, 6), from = 0, to = 1, ylab = "Densidade", xlab = "x")  
curve(dbeta(x, 4, 4), from = 0, to = 1, add = TRUE)  
curve(dbeta(x, 6, 2), from = 0, to = 1, add = TRUE)
```



- b) Adicionando título no gráfico:

```
curve(dbeta(x, 2, 6), from = 0, to = 1, ylab = "Densidade", xlab = "x")
curve(dbeta(x, 4, 4), from = 0, to = 1, add = TRUE)
curve(dbeta(x, 6, 2), from = 0, to = 1, add = TRUE)
title(expression(f(y) == frac(1, B(a, b)) * y^{a-1} * (1-y)^{b-1}))
```

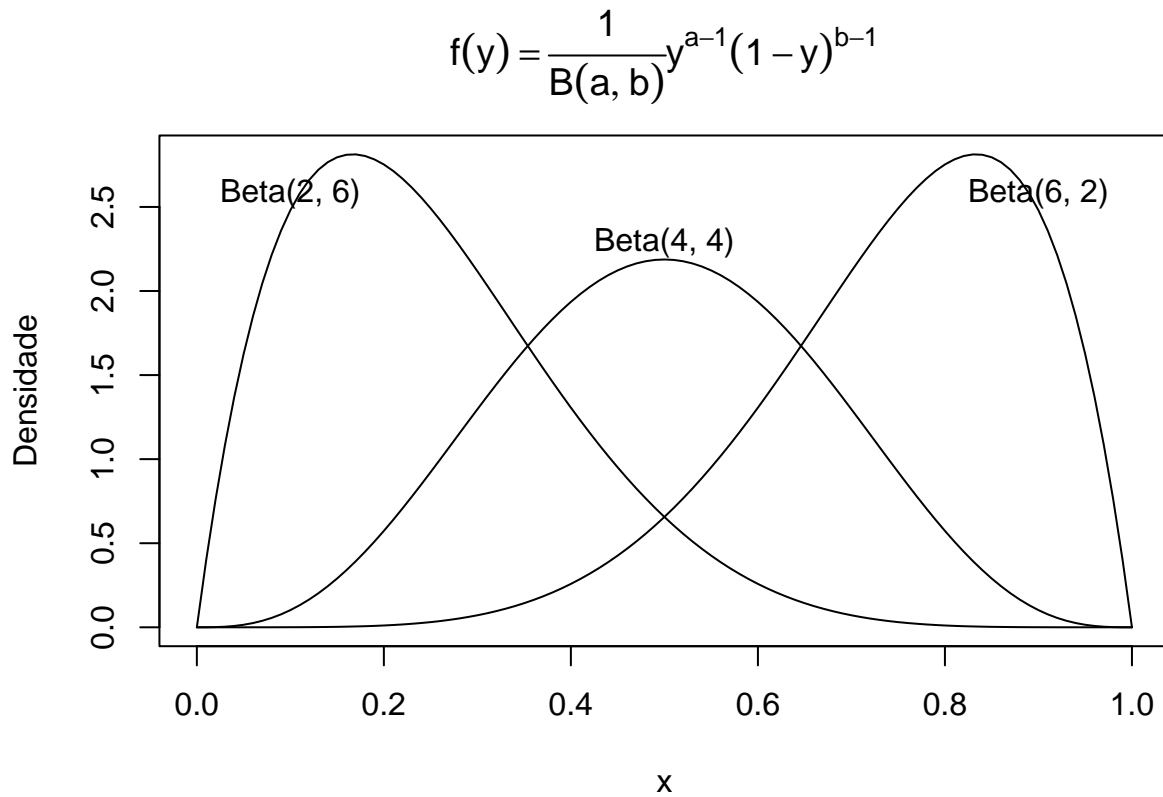


- c) Adicionando rótulos a cada uma das curvas com a função `text()`:

```
curve(dbeta(x, 2, 6), from = 0, to = 1, ylab = "Densidade", xlab = "x")
curve(dbeta(x, 4, 4), from = 0, to = 1, add = TRUE)
curve(dbeta(x, 6, 2), from = 0, to = 1, add = TRUE)

title(expression(f(y) == frac(1, B(a, b)) * y^{a-1} * (1-y)^{b-1}))

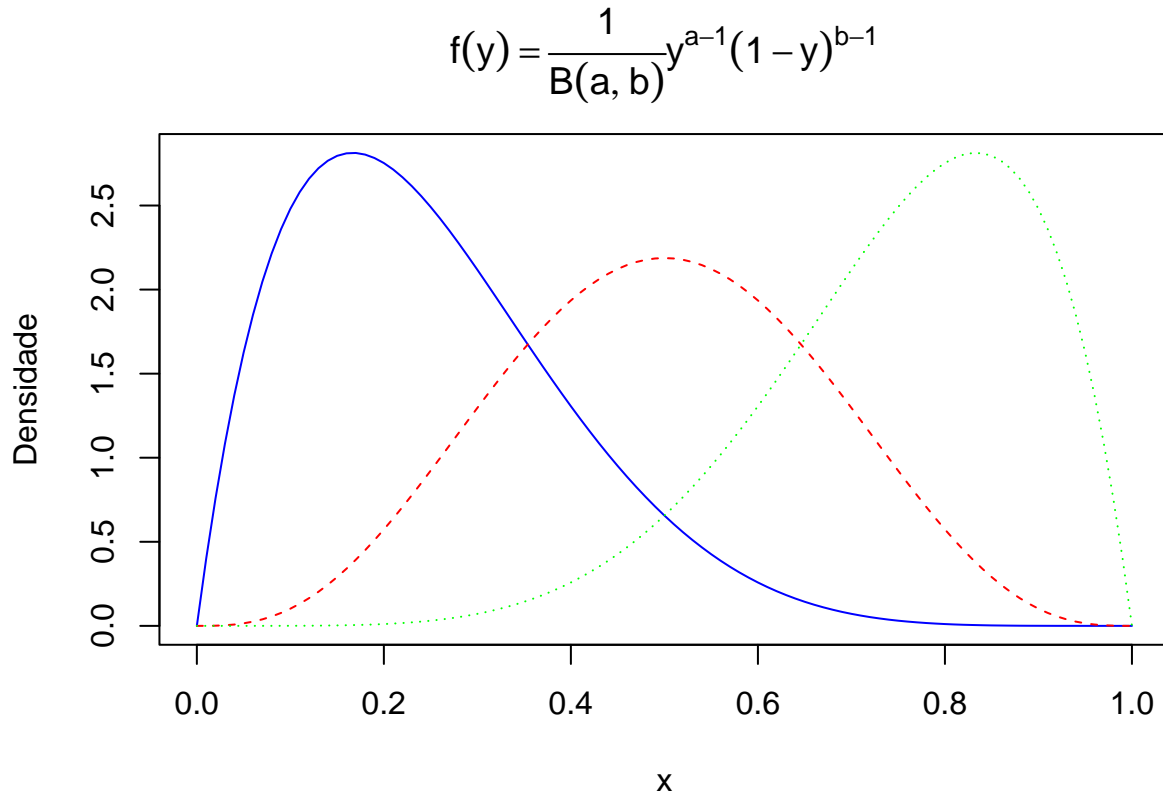
text(0.1, dbeta(0.1, 2, 6) + 0.1, "Beta(2, 6)")
text(0.5, dbeta(0.5, 4, 4) + 0.1, "Beta(4, 4)")
text(0.9, dbeta(0.9, 6, 2) + 0.1, "Beta(6, 2)")
```



- d) Refazendo o gráfico do item (a), adicionando cores e tipos de linha diferentes para cada uma das três curvas:

```
curve(dbeta(x, 2, 6), from = 0, to = 1, ylab = "Densidade", xlab = "x", col = "blue", lty = 1)
curve(dbeta(x, 4, 4), from = 0, to = 1, add = TRUE, col = "red", lty = 2)
curve(dbeta(x, 6, 2), from = 0, to = 1, add = TRUE, col = "green", lty = 3)

title(expression(f(y) == frac(1, B(a, b)) * y^{a-1} * (1-y)^{b-1}))
```



- e) Adicionando legenda sem usar a função `text()`:

```
curve(dbeta(x, 2, 6), from = 0, to = 1, ylab = "Densidade", xlab = "x", col = "blue", lty = 1)
curve(dbeta(x, 4, 4), from = 0, to = 1, add = TRUE, col = "red", lty = 2)
curve(dbeta(x, 6, 2), from = 0, to = 1, add = TRUE, col = "green", lty = 3)

title(expression(f(y) == frac(1, B(a, b)) * y^{a-1} * (1-y)^{b-1}))

legend("topright", legend = c("Beta(2, 6)", "Beta(4, 4)", "Beta(6, 2)"),
      col = c("blue", "red", "green"), lty = c(1, 2, 3))
```

