ESTATÍSTICA AVANÇADA: MODELOS NÃO LINEARES

Modulo 1: Introdução ao R e RStudio

Gilvan Guedes [Cedeplar - UFMG]
Melissa Pinho [Estatística - UFMG]

Escola do Legislativo - ALMG
Belo Horizonte, Minas Gerais

8 de setembro de 2015

Sumário

1 R You Ready? Aspectos Básicos do R

Fonte: De Vries & Meys (2012) - R for Dummies

R é um ambiente de programação que suporta diversos tipos de análises e gerenciamento de dados. A grande vantagem do R é que além de ser gratuito e de código aberto, ele funciona em diversos sistemas operacionais: GNU Linux, Microsoft Windows, Mac OS X e outros.

R é muito mais uma linguagem de programação do que uma aplicação ou um pacote estatístico. Quando, pela primeira vez, você fizer o download do R, ele automaticamente irá instalar um console para o tipo de ambiente (sistema operacional) que mais se adeque ao seu caso. Embora funcional, o console padrão do R é muito básico, menos didático e pode variar em funcionalidades e visual entre ambientes operacionais distintos. A aplicação RStudio tem a vantagem de ser uma aplicação homogênea para todas as plataformas, além de possuir uma aparência de múltiplas janelas (*Source; Console; Workspace and history; Files, plots, package, and help*).

A coumidade R é muito ativa e tem crescido exponencialmente. Alguns exemplos de locais na internet e suas principais utilizações:

1. **Mailing lists**: `www.rproject.org/mail.html`
2. **Perguntas e respostas**:
   (a) **Stack Overflow (P&R de programação)**: `www.stackoverflow.com/questions/tagged/r`
   (b) **CrossValidated (P&R de estatística)**:`http://stats.stackexchange.com/questions/tagged/r`
3. **Redes sociais**: `www.twitter.com/search/rstats`

O programa básico pode ser instalado a partir do website `http://www.r-project.org/`.

O ambiente RStudio pode ser instalado a partir do website `https://www.rstudio.com`.

2 Funções Básicas

Toda vez que você inicia uma sessão de trabalho no R, é importante definir alguns elementos. Esses elementos podem ficar armazenados em um *script*. Isso facilita sua vida para futuras replicações e para documentação do seu trabalho.

A coisa mais importante é saber como obter ajuda. Veja algumas das opções:

1. Para saber como usar uma função específica:

```
help(lm)     # Help da funcao para regressao linear
?lm          # Help da funcao para regressao linear
```

2. Se não conhecemos a função de que precisamos, podemos fazer uma busca de texto mais completa, no nome da função e na sua descrição, através do seguinte comando:

```
help.search("lm")    # Help no CRAN da funcao para regressao linear
??lm               # Help no CRAN da funcao para regressao linear
```

3. Se isso não for suficiente para localizar o comando de que precisamos, uma opção é buscar o comando diretamente na internet. Esse comando só funcionará caso o computador que esteja sendo usado tenha acesso à internet:

```
RSiteSearch("linear model")
```

Vejamos um exemplo de algumas funções básicas:

1. Limpando a memória do programa:

```
ls()
rm(list=ls(all=TRUE))
```

2. Definindo o diretório de trabalho:

```
getwd()

## [1] "/Users/grguedes/APOSTILA"

setwd("/Users/grguedes/APOSTILA/")
getwd()

## [1] "/Users/grguedes/APOSTILA"
```

3. Atualizando versões dos pacotes do R instalados no seu computador:

```
local({r <- getOption("repos")
       r["CRAN"] <- "http://www.vps.fmvz.usp.br/CRAN/"
       options(repos=r)})
#update.packages()
```

4. Verificando todos os pacotes carregados e instalados:

```
library()
installed.packages()
```

5. Verificando quais pacotes estão carregados na sessão:

```
search()

##  [1] ".GlobalEnv"        "package:knitr"     "package:stats"
##  [4] "package:graphics"  "package:grDevices" "package:utils"
##  [7] "package:datasets"  "package:methods"   "Autoloads"
## [10] "package:base"
```

Vejamos mais algumas funções úteis no dia-a-dia do uso do R para manipular dados:

1. A função "sum()"retorna a soma dos elementos do objeto

```
x <- c(2,5,6,3,4,-8,-9,7,6,5,3,4,12,11)
sum(x)

## [1] 51
```

2. A função "prod()"retorna o produto dos elementos do objeto

```r
prod(x)

## [1] 17244057600
```

3. A função “table()”retorna uma tabela de frequência do(s) objeto(s)

```r
table(x) # Frequencia absoluta simples

## x
## -9 -8  2  3  4  5  6  7 11 12
##  1  1  1  2  2  2  2  1  1  1

w <- rep(c(0,5),times=c(10,4))
table(x,w)  # Frequencia absoluta cruzada

##     w
## x    0 5
##   -9 1 0
##   -8 1 0
##   2  1 0
##   3  1 1
##   4  1 1
##   5  2 0
##   6  2 0
##   7  1 0
##   11 0 1
##   12 0 1
```

4. A função “mean()”retorna o produto dos elementos do objeto

```r
mean(x)

## [1] 3.642857
```

5. A função “var()”retorna a variância dos elementos do objeto

```r
var(x)

## [1] 34.55495
```

6. A função “log()”retorna o logaritmo dos elementos do objeto

```r
log(x)          # Retorna o logaritmo natural do objeto x

##  [1] 0.6931472 1.6094379 1.7917595 1.0986123 1.3862944       NaN
##  [7]       NaN 1.9459101 1.7917595 1.6094379 1.0986123 1.3862944
## [13] 2.4849066 2.3978953

log(x, base=10) # Retorna o logaritmo de base 10 do objeto x

##  [1] 0.3010300 0.6989700 0.7781513 0.4771213 0.6020600       NaN
##  [7]       NaN 0.8450980 0.7781513 0.6989700 0.4771213 0.6020600
## [13] 1.0791812 1.0413927

log1p(x)        # Retorna log(1+x) acuradamente tambem para |x| << 1

##  [1] 1.098612 1.791759 1.945910 1.386294 1.609438      NaN      NaN
##  [8] 2.079442 1.945910 1.791759 1.386294 1.609438 2.564949 2.484907
```

7. A função "exp()"retorna o exponencial dos elementos do objeto

```r
exp(x)    # Retorna o exponencial de o objeto x

##  [1] 7.389056e+00 1.484132e+02 4.034288e+02 2.008554e+01 5.459815e+01
##  [6] 3.354626e-04 1.234098e-04 1.096633e+03 4.034288e+02 1.484132e+02
## [11] 2.008554e+01 5.459815e+01 1.627548e+05 5.987414e+04

expm1(x)  # Retorna exp(x) - 1 acuradamente tambem para |x| << 1

##  [1]  6.389056e+00  1.474132e+02  4.024288e+02  1.908554e+01
##  [5]  5.359815e+01 -9.996645e-01 -9.998766e-01  1.095633e+03
##  [9]  4.024288e+02  1.474132e+02  1.908554e+01  5.359815e+01
## [13]  1.627538e+05  5.987314e+04
```

8. A função "abs()"retorna o valor absoluto dos elementos do objeto

```r
abs(x)

##  [1]  2  5  6  3  4  8  9  7  6  5  3  4 12 11
```

2.1 Instalando pacotes

Você pode achar pacotes no site CRAN (*Comprehensive R Archive Network*) sob a aba "Packages".Você pode também baixar direto do CRAN via seu R com o comando abaixo:

```
install.packages("Hmisc", dependencies=TRUE)
```

A função "dependencies=TRUE"diz ao R para instalar quaisquer pacotes que esse pacote dependa ou que o autor sugere que seja útil.

Para carregar um pacote específico, digite:

```
library("Hmisc")
```

```
## Loading required package:  grid
## Loading required package:  lattice
## Loading required package:  survival
## Loading required package:  Formula
## Loading required package:  ggplot2
##
## Attaching package:  'Hmisc'
##
## The following objects are masked from 'package:base':
##
##    format.pval, round.POSIXt, trunc.POSIXt, units
```

```
#ls("package:Hmisc") # listando as funcoes do pacote Hmisc
help("Hmisc")
```

```
## No documentation for 'Hmisc' in specified packages and libraries:
## you could try '??Hmisc'
```

```
install.packages("prettyR")
```

```
##
## The downloaded binary packages are in
##  /var/folders/xl/tmfgg9y1719_9z70r1dydlm00000gn/T//RtmptGBvbq/downloaded_packages
```

```
library("prettyR") # Veja a mensagem que a funcao describe
```

```
##
## Attaching package:  'prettyR'
##
## The following object is masked from 'package:Hmisc':
##
##    describe
```

```
# do pacote "Hmisc" esta sendo mascarada pela funcao
# homonima contida no pacote "prettyR" instalado depois.
# Se quiser evitar esse problema, use o comando prettyR
# e desconecte-o assim que terminar.
search()
```

```
##  [1] ".GlobalEnv"        "package:prettyR"   "package:Hmisc"
##  [4] "package:ggplot2"   "package:Formula"   "package:survival"
##  [7] "package:lattice"   "package:grid"      "package:knitr"
## [10] "package:stats"     "package:graphics"  "package:grDevices"
## [13] "package:utils"     "package:datasets"  "package:methods"
## [16] "Autoloads"         "package:base"
```

```
detach("package:prettyR")
search()

##  [1] ".GlobalEnv"        "package:Hmisc"     "package:ggplot2"
##  [4] "package:Formula"   "package:survival"  "package:lattice"
##  [7] "package:grid"      "package:knitr"     "package:stats"
## [10] "package:graphics"  "package:grDevices" "package:utils"
## [13] "package:datasets"  "package:methods"   "Autoloads"
## [16] "package:base"
```

Quando instalamos uma nova versão do R, todos os *add-ons* têm que ser reinstalados. Um jeito simples é manter um vetor com todos os comandos usualmente utilizados. Esse vetor pode ser salvo em um *script* do R para futuros usos, e basta você indexá-lo em cada sessão nova, usando a função "source(meusPacotes)"

```
meusPacotes <- c("car","foreign","hexbin",
"ggplot2","gmodels","gplots","Hmisc",
"lattice","reshape","prettyR","Rcmdr")

#install.packages(meusPacotes, dependencies=TRUE)
```

Se quiser desinstalar pacotes específicos, basta digitar:

```
detach("package:Hmisc") # Primeiro tem que remover o uso
remove.packages("Hmisc") # Depois desinstala

## Removing package from '/Library/Frameworks/R.framework/Versions/3.1/Resources/lib
## (as 'lib' is unspecified)
```

Caso queira acessar bancos de dados de treinamento:

```
data() # lista todos os bancos de dados de treinamento basicos
```

2.2 Salvando e Carregando seu Trabalho

É muito importante que você salve o seu trabalho com frequência, pois nunca sabemos quando uma oscilação na corrente elétrica ou uma pane no computador pode ocorrer. Vamos ver algumas formas diferentes de salvar seu trabalho.

Caso queira salvar todos um dos objetos específicos criados em seu *workspace*, use a função "save()":

```
x <- c(1,2,3,4,5)
y <- c(6:10)
w <- 2*x+rnorm(5)
z <- matrix(c(1:45),nrow=3,ncol=5,byrow=TRUE)
minhaLista <- c(x,y,w,z)

save(vetorx=x, vetory=y, file="meusvetores.RData")
```

Caso queira remover (apagar) algum objeto, use a função "rm()":

```
rm(x,y)
```

Caso queira salvar todos os objetos criados em seu *workspace*, use a função "save.image()":

```
save.image(file="meuWorkspace.RData")
```

Caso queira carregar o workspace (objetos) salvo, digite:

```
load("meuWorkspace.RData")
```

Para salvar a sua história de trabalho (incluindo comandos), digite:

```
#savehistory(file="meuHistorico.RHistory")
```

Para carregar a história de trabalho salva, digite:

```
#loadhistory(file="meuHistorico.RHistory")
```

3 Objetos no R

O R admite trabalhar com 5 tipos principais de objetos:

- Vetores
- Matrizes
- Arrays
- Listas
- Bases de dados

3.1 Vetores

Um vetor é um conjunto de dados dispostos em uma única dimensão. A dimensão de um vetor é o seu comprimento.

No R, podemos identificar o comprimento de um vetor por meio do seguinte comando:

```
x <- seq(from = -2.7, to = 1.3, length.out = 9)
length(x)

## [1] 9

y <- seq(from = 4.5, to = 2.5, by = -0.5)
length(y)

## [1] 5
```

Vejamos alguns exemplos a seguir:

1. Vetor numérico:

```
x <- c(1,2,3,4,5)
print(x)

## [1] 1 2 3 4 5
```

2. Vetor de texto:

```
nomes <- c("Maria","Joao","Pedro","Ana","Clara")
print(nomes)

## [1] "Maria" "Joao"  "Pedro" "Ana"   "Clara"
```

3. Vetor categórico:

```
qualidade <- factor(c(1,1,3,2,1),
                    levels=c(1,2,3),
                    labels=c("Ruim","Regular","Bom")
)
print(qualidade)

## [1] Ruim    Ruim    Bom     Regular Ruim
## Levels: Ruim Regular Bom
```

4. Vetor sequencial:

```
c(1:20)

##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

seq(from = -2.7, to = 1.3, length.out = 9)

## [1] -2.7 -2.2 -1.7 -1.2 -0.7 -0.2  0.3  0.8  1.3

seq(from = 4.5, to = 2.5, by = -0.5)

## [1] 4.5 4.0 3.5 3.0 2.5
```

5. Vetor repetido:

```r
rep(c(0, 0, 7), times = 3)
```

```
## [1] 0 0 7 0 0 7 0 0 7
```

```r
rep(c(2, 4, 2), each = 3)
```

```
## [1] 2 2 2 4 4 4 2 2 2
```

```r
rep(c(0, 7), times = c(4,2))
```

```
## [1] 0 0 0 0 7 7
```

```r
rep(1:3,length.out=7)
```

```
## [1] 1 2 3 1 2 3 1
```

3.2 Matrizes

Uma matriz é um conjunto de vetores linha e vetores coluna. Podemos também interpretar uma matriz como uma lista em que todos os seus componentes têm o mesmo comprimento. Assim, toda matriz tem duas dimensões. A representação da dimensão de uma matriz é dada por:

$$A_{r \times c}$$

No R, podemos obter a dimensão da matriz com o seguinte comando:

```r
A <- matrix(c(1:45),nrow=3,ncol=5,byrow=TRUE)
dim(z)
```

```
## [1] 3 5
```

Podemos criar matrizes de diferentes formas:

1. Combinando vetores

```r
a1 <- c(1:5)
a2 <- c(6:10)
a3 <- c(11:15)
a4 <- c(16:20)
A <- rbind(a1,a2,a3,a4)
print(A)
```

```
##    [,1] [,2] [,3] [,4] [,5]
## a1    1    2    3    4    5
## a2    6    7    8    9   10
## a3   11   12   13   14   15
## a4   16   17   18   19   20
```

2. Utilizando a função “matrix()”

```
A <- matrix(c(1,2,3,4,5,
              6,7,8,9,10,
              11,12,13,14,15,
              16,17,18,19,20),nrow=4,ncol=5,byrow=TRUE)
print(A)

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    6    7    8    9   10
## [3,]   11   12   13   14   15
## [4,]   16   17   18   19   20
```

Podemos também executar operações com matrizes. Vejamos alguns exemplos:

1. Estimando duas matrizes quadradas

```
A <- matrix(c(1,5,
              3,6),ncol=2,nrow=2,byrow=TRUE)
print(A)

##      [,1] [,2]
## [1,]    1    5
## [2,]    3    6

B <- matrix(c(-3,2,
              12,-4),ncol=2,nrow=2,byrow=TRUE)
print(B)

##      [,1] [,2]
## [1,]   -3    2
## [2,]   12   -4
```

2. Calculando a transposta da matriz A

```
At <- t(A)
print(At)

##      [,1] [,2]
## [1,]    1    3
## [2,]    5    6
```

3. Calculando a inversa da matriz A

```r
Ainv <- solve(A)
print(Ainv)

##             [,1]       [,2]
## [1,] -0.6666667  0.5555556
## [2,]  0.3333333 -0.1111111
```

4. Multiplicando duas matrizes

```r
AAt <- A*At
# Matrizes de mesma dimensao
# cada elemento a(ij) x at(ij)

print(AAt)

##      [,1] [,2]
## [1,]    1   15
## [2,]   15   36

AAt <- A%*%At
# Matrizes de produtos internos identicos
# Somak[a(ik)*at(kj)]

print(AAt)

##      [,1] [,2]
## [1,]   26   33
## [2,]   33   45
```

5. Somando e subtraindo matrizes

```r
AsubB <- A-B  # Retorna matriz A-B (matriz diferenca)
print(AsubB)

##      [,1] [,2]
## [1,]    4    3
## [2,]   -9   10

AplusB <- A+B # Retorna matriz A+B (matriz soma)
print(AplusB)

##      [,1] [,2]
## [1,]   -2    7
## [2,]   15    2
```

3.3 Base de Dados

A base de dados é uma **matriz generalizada**; ou seja, a que pode conter tanto vetores numéricos quanto de texto.

A base de dados também é um tipo de *lista*, porém é uma lista em que todos os seus componentes são forçados a ter o *mesmo comprimento*. A maioria dos dados trabalhados pelos cientistas sociais são disponibilizados em formato de base de dados. Portanto, você irá lidar constantemente com esse tipo de estrutura de dado no R.

Vamos agora criar uma base de dados a partir de vetores.

```
workshop <- factor(c(1,2,1,2,1,2,1,2),
                   levels <- c(1,2,3,4),
                   labels <- c("R","Stata","SPSS","SAS"))
gender <- factor(c("f","f","f",NA,"m","m","m","m"),
                 levels <- c("f","m"),
                 labels <- c("Mulher","Homem"))
q1 <- c(1,2,2,3,4,5,5,4)
q2 <- c(1,1,2,1,5,4,3,5)
q3 <- c(5,4,4,NA,2,5,4,5)
q4 <- c(1,1,3,3,4,5,4,5)
pretest <- rnorm(n=8, mean=70, sd=5)
posttest <- rnorm(n=8, mean=80, sd=5)

meusdados <- data.frame(workshop,
                        sex=gender,q1,q2,q3,q4,
                        pretest,posttest)
meusdados
```

```
##   workshop    sex q1 q2 q3 q4  pretest posttest
## 1        R Mulher  1  1  5  1 73.45549 87.58443
## 2    Stata Mulher  2  1  4  1 67.95146 74.80625
## 3        R Mulher  2  2  4  3 74.59316 79.75358
## 4    Stata   <NA>  3  1 NA  3 72.44981 78.16843
## 5        R  Homem  4  5  2  4 65.59845 86.63922
## 6    Stata  Homem  5  4  5  5 82.09950 88.73906
## 7        R  Homem  5  3  4  4 72.12604 73.68634
## 8    Stata  Homem  4  5  5  5 75.95329 77.95451
```

Caso você queira visualizar sua base de dados criada, uma janela de visualização aparecerá na aba de programação do RStudio. Para tanto, basta digitar:

```
View(meusdados)
```

Caso esteja interessado em conhecer a estrutura do seu objeto, digite:

```
str(x)
```

```
##  num [1:5] 1 2 3 4 5
```

```
str(A)
```

```
##  num [1:2, 1:2] 1 3 5 6

str(meusdados)

## 'data.frame': 8 obs. of  8 variables:
##  $ workshop: Factor w/ 4 levels "R","Stata","SPSS",..: 1 2 1 2 1 2 1 2
##  $ sex     : Factor w/ 2 levels "Mulher","Homem": 1 1 1 NA 2 2 2 2
##  $ q1      : num  1 2 2 3 4 5 5 4
##  $ q2      : num  1 1 2 1 5 4 3 5
##  $ q3      : num  5 4 4 NA 2 5 4 5
##  $ q4      : num  1 1 3 3 4 5 4 5
##  $ pretest : num  73.5 68 74.6 72.4 65.6 ...
##  $ posttest: num  87.6 74.8 79.8 78.2 86.6 ...
```

Veja que no exemplo acima, haviamos transformado o objeto "gender"em um vetor categórico (*factor*). No entanto, a função "data.frame()"sempre força todos os vetores de texto serem transformados em variáveis categóricas.

Nem sempre desejamos que isso aconteça. Imagine que temos um vetor adicional, de texto, com o nome dos participantes de um *workshop* para treinamento de pacotes estatísticos. Esse vetor deve conservar seu caráter de texto após ser incluido no banco de dados. Para tanto, devemos acionar a opção "stringsAsFactors=FALSE"na função "data.frame()". Veja como podemos fazer isso:

```
nomes <- c("Aldo", "Magda", "Joana",
           "Pedro", "Marcelo", "Jose", "Maria", "Ilda")

meusdados <- data.frame(nomes,workshop,
                        sex=gender,q1,q2,q3,q4,
                        pretest,posttest,
                        stringsAsFactors=FALSE,
                        row.names=NULL)

print(meusdados)

##     nomes workshop    sex q1 q2 q3 q4  pretest posttest
## 1    Aldo        R Mulher  1  1  5  1 73.45549 87.58443
## 2   Magda    Stata Mulher  2  1  4  1 67.95146 74.80625
## 3   Joana        R Mulher  2  2  4  3 74.59316 79.75358
## 4   Pedro    Stata   <NA>  3  1 NA  3 72.44981 78.16843
## 5 Marcelo        R  Homem  4  5  2  4 65.59845 86.63922
## 6    Jose    Stata  Homem  5  4  5  5 82.09950 88.73906
## 7   Maria        R  Homem  5  3  4  4 72.12604 73.68634
## 8    Ilda    Stata  Homem  4  5  5  5 75.95329 77.95451

str(meusdados)

## 'data.frame': 8 obs. of  9 variables:
##  $ nomes   : chr  "Aldo" "Magda" "Joana" "Pedro" ...
##  $ workshop: Factor w/ 4 levels "R","Stata","SPSS",..: 1 2 1 2 1 2 1 2
```

```
##  $ sex     : Factor w/ 2 levels "Mulher","Homem": 1 1 1 NA 2 2 2 2
##  $ q1      : num  1 2 2 3 4 5 5 4
##  $ q2      : num  1 1 2 1 5 4 3 5
##  $ q3      : num  5 4 4 NA 2 5 4 5
##  $ q4      : num  1 1 3 3 4 5 4 5
##  $ pretest : num  73.5 68 74.6 72.4 65.6 ...
##  $ posttest: num  87.6 74.8 79.8 78.2 86.6 ...
```

Caso queira ver o nome das linhas do banco de dados, digitar:

```
row.names(meusdados)

## [1] "1" "2" "3" "4" "5" "6" "7" "8"
```

Caso queira alterar esse nome das linhas por alguma variável do seu banco de dados, digitar:

```
row.names(meusdados) <- meusdados$nomes
print(meusdados)

##           nomes workshop    sex q1 q2 q3 q4  pretest posttest
## Aldo       Aldo        R Mulher  1  1  5  1 73.45549 87.58443
## Magda     Magda    Stata Mulher  2  1  4  1 67.95146 74.80625
## Joana     Joana        R Mulher  2  2  4  3 74.59316 79.75358
## Pedro     Pedro    Stata   <NA>  3  1 NA  3 72.44981 78.16843
## Marcelo Marcelo        R  Homem  4  5  2  4 65.59845 86.63922
## Jose       Jose    Stata  Homem  5  4  5  5 82.09950 88.73906
## Maria     Maria        R  Homem  5  3  4  4 72.12604 73.68634
## Ilda       Ilda    Stata  Homem  4  5  5  5 75.95329 77.95451
```

Caso queira alterar o nome das colunas do seu banco de dados, digitar:

```
# Alterando apenas um nome
colnames(meusdados)[2] <- "genero"
print(meusdados)

##           nomes genero    sex q1 q2 q3 q4  pretest posttest
## Aldo       Aldo      R Mulher  1  1  5  1 73.45549 87.58443
## Magda     Magda  Stata Mulher  2  1  4  1 67.95146 74.80625
## Joana     Joana      R Mulher  2  2  4  3 74.59316 79.75358
## Pedro     Pedro  Stata   <NA>  3  1 NA  3 72.44981 78.16843
## Marcelo Marcelo      R  Homem  4  5  2  4 65.59845 86.63922
## Jose       Jose  Stata  Homem  5  4  5  5 82.09950 88.73906
## Maria     Maria      R  Homem  5  3  4  4 72.12604 73.68634
## Ilda       Ilda  Stata  Homem  4  5  5  5 75.95329 77.95451

# Alterando todos os nomes de uma vez
colnames(meusdados) <- c("workshop","sex",
```

```
                          "q1","q2","q3","q4",
                          "pretest","posttest")
names(meusdados)

## [1] "workshop" "sex"      "q1"       "q2"       "q3"       "q4"
## [7] "pretest"  "posttest" NA
```

3.4 Listas

As linhas são os objetos mais flexíveis do R. Elas podem combinar diferentes tipos de objetos, preservando suas estruturas originais.

Quando rodamos uma regressão, diferentes resultados ficam armazenados em listas, o que facilita na hora de acessarmos partes específicas desses resultados. Diferentemente dos outros objetos, as listas têm um indexador a mais, o qual identifica o objeto na lista, e é representado pelo símbolo "[[]]".

Vamos agora criar uma lista, combinando vetores, uma matriz e uma base de dados:

```
minhalista <- list(nomes, workshop, gender,
                   q1, q2, q3, q4, pretest, posttest, A, meusdados)
minhalista

## [[1]]
## [1] "Aldo"    "Magda"   "Joana"   "Pedro"   "Marcelo" "Jose"
## [7] "Maria"   "Ilda"
##
## [[2]]
## [1] R     Stata R     Stata R     Stata R     Stata
## Levels: R Stata SPSS SAS
##
## [[3]]
## [1] Mulher Mulher Mulher <NA>   Homem  Homem  Homem  Homem
## Levels: Mulher Homem
##
## [[4]]
## [1] 1 2 2 3 4 5 5 4
##
## [[5]]
## [1] 1 1 2 1 5 4 3 5
##
## [[6]]
## [1]  5  4  4 NA  2  5  4  5
##
## [[7]]
## [1] 1 1 3 3 4 5 4 5
##
## [[8]]
```

```
## [1] 73.45549 67.95146 74.59316 72.44981 65.59845 82.09950 72.12604
## [8] 75.95329
##
## [[9]]
## [1] 87.58443 74.80625 79.75358 78.16843 86.63922 88.73906 73.68634
## [8] 77.95451
##
## [[10]]
##      [,1] [,2]
## [1,]    1    5
## [2,]    3    6
##
## [[11]]
##         workshop   sex     q1 q2 q3 q4 pretest posttest       NA
## Aldo        Aldo     R Mulher  1  1  5       1 73.45549 87.58443
## Magda      Magda Stata Mulher  2  1  4       1 67.95146 74.80625
## Joana      Joana     R Mulher  2  2  4       3 74.59316 79.75358
## Pedro      Pedro Stata   <NA>  3  1 NA       3 72.44981 78.16843
## Marcelo  Marcelo     R  Homem  4  5  2       4 65.59845 86.63922
## Jose        Jose Stata  Homem  5  4  5       5 82.09950 88.73906
## Maria      Maria     R  Homem  5  3  4       4 72.12604 73.68634
## Ilda        Ilda Stata  Homem  4  5  5       5 75.95329 77.95451
```

Se quisermos acessar apenas um objeto específico dentro da lista, podemos acessá-lo facilmente. Vamos ver quais são as respostas dos participantes 2, 3 e 4 do *workshop*:

```
minhalista[[11]][c(2,3,4),]

##       workshop   sex     q1 q2 q3 q4 pretest posttest       NA
## Magda    Magda Stata Mulher  2  1  4       1 67.95146 74.80625
## Joana    Joana     R Mulher  2  2  4       3 74.59316 79.75358
## Pedro    Pedro Stata   <NA>  3  1 NA       3 72.44981 78.16843
```

4 Importando e Exportando dados com o R

O R consegue importar dados de diversos formatos. Nesta seção, iremos verificar alguns comandos e os principais pacotes utilizados para importar e exportar dados do R.

O comando "setwd()"é utilizado para definir o diretório de trabalho do R.

```
# Mostra o diretorio atual:
getwd()

## [1] "/Users/grguedes/APOSTILA"

# Define o dire\orio de trabalho:
setwd("/Users/grguedes/APOSTILA/")
```

Vamos apresentar a importação e exportação de dados nos seguintes formatos:

- Arquivos ".txt"(texto por tabulação ou vírgula)
- Arquivos ".csv"(texto separado por vírgula)
- Arquivos ".xls"e ".xlsx"(arquivos do Microsoft Excel)
- Arquivos ".dta"(bases de dados do Stata)
- Arquivos ".sav"(bases de dados do SPSS)
- Arquivos ".dbf"(bases de dados de base para mapas)

4.1 Arquivos txt

Para importar os dados, usamos a função "read.table()". Para exportar dados do R para o formato ".txt", utiliza-se a função "write.table()". Após este comando, será criado um arquivo ".txt"no diretório de trabalho definido anteriormente. Tal arquivo contém os dados do objeto "dados_txt".

```
# Importando dados do formato .txt
dados_txt <- read.table("mydata2.txt",  # Nome do arquivo txt
                        sep=",")
# sep="," colunas sao separadas por virgula

dados_txt

##   V1 V2 V3 V4 V5 V6
## 1  1  1  1  1  5  1
## 2  2  1  2  1  4  1
## 3  1  1  2  2  4  3
## 4  2 NA  3  1 NA  3
## 5  1  2  4  5  2  4
## 6  2  2  5  4  5  5
## 7  1  2  5  3  4  4
## 8  2  2  4  5  5  5

# Exportando dados no formato .txt:

write.table(x=dados_txt,  # Nome do objeto que sera exportado
            file="dados_exportacao.txt")
# Nome do arquivo que sera criado no diretorio de trabalho
```

4.2 Arquivos csv

A função "read.table"também pode ser utilizada para ler arquivos do formato ".csv", conforme o exemplo abaixo. Analogamente, a função "write.csv()"é utilizada para exportar dados do R para o formato ".csv".

```r
# Importando dados do formato .csv

dados_csv <- read.table("mydata.csv",  # Nome do arquivo csv
                        sep=",")
# sep="," indica que as colunas sao separadas por virgula
# sep="\t" indica que colunas sao separadas por tabulacao

# Exportando dados no formato .csv:

write.csv(x=dados_csv,
          file="dados_exportacao.csv")
```

4.3 Arquivos xls e xlsx

```r
install.packages("xlsx")

require(xlsx)

## Loading required package:  xlsx
## Loading required package:  rJava
## Loading required package:  xlsxjars

dados_xls <- read.xlsx("mydata100.xls",  # Nome do arquivo txt
                       sheetIndex=1)
# sheetIndex indica em qual planilha se encontram os dados
# poderia ter sido alterado para sheetName="mydata"
# ou seja, nome que aparece na aba.

# Exportando dados no formato .xlsx:

write.xlsx(x=dados_xls,
           file="dados_exportacao.xlsx")
```

Além da função "read.xlsx()", o pacote "xlsx"contém a função "read.xlsx2()". Essa última é utilizada quando o banco de dados original é muito grande. No entanto, cada variável é importada como uma variável categórica (fator na linguagem do R). Nesse caso, cada variável numérica deve ser retransformada, utilizando os seguintes comandos:

```r
with(dados_xls, as.numeric(as.character(q1)))
```

4.4 Arquivos dta, sav e dbf

Para importar e exportar dados nos formatos ".dta", ".sav", e ".dbf", deve-se utilizar o pacote "foreign".

```r
install.packages("foreign")

require(foreign)

## Loading required package:  foreign

#=====================================================================
# Importando dados do formato .dta

dados_dta <- read.dta("mydata.dta")

# Exportando dados no formato .dta:

write.dta(dataframe=dados_txt,
          file="dados_exporta??o.dta")

#=====================================================================
# Importando dados do formato .sav

dados_sav <- read.spss("mydata.sav",
                       to.data.frame=TRUE)

# Exportando dados no formato .sav:

write.foreign(df=dados_sav,
              datafile="dados_exportacao",
              codefile="dados_exportacao",
              package="SPSS")

#=====================================================================
# Importando dados do formato .dbf

dados_dbf <- read.dbf("SP_MUNIC2013_NOVO.dbf")

# Exportando dados no formato .dbf:

write.dbf(dataframe=dados_dbf,
          file="dados_exportacao.dbf")
```

5 Manipulação de dados no R

Nesta sessão vamos aprender algumas operações importantes com bancos de dados. Veremos como fazer as seguintes operações:

- Criar um banco de dados a partir dos vetores
- Criar um banco de dados que é subconjunto de outro

- Criar uma nova variável num banco já existente
- Recategorizar variáveis
- Dar nome às variáveis e às categorias das variáveis
- Agregar valores de unidades menores em maiores
- Junção de banco de dados

Vamos começar carregando dois bancos de dados para trabalharmos com essas operações. Trabalharemos com dois bancos:

- O banco "auto.dta"é constituído de 74 observações de carros no mercado americano em 1978 com as variáveis:
  - make: Make and Model
  - price: Price
  - mpg: Mileage (mpg)
  - rep78: Repair Record 1978
  - headroom: Headroom (in.)
  - trunk: Trunk space (cu. ft.)
  - weight: Weight (lbs.)
  - length: Length (in.)
  - turn: Turn Circle (ft.)
  - displacement: Displacement (cu. in.)
  - gear_ratio: Gear Ratio
  - foreign: Car type
- O banco "binlfp2.dta"é constituído de 753 observações de mulheres casadas no mercado de trabalho americano em 1976, com as variáveis:
  - lfp: Paid Labor Force: 1=yes 0=no
  - k5: # kids < 6
  - k618: # kids 6-18
  - age: Wife's age in years
  - wc: Wife College: 1=yes 0=no
  - hc: Husband College: 1=yes 0=no
  - lwg: Log of wife's estimated wages
  - inc: Family income excluding wife's

Vamos começar carregando esses dois bancos de dados no R. Lembre-se que precisamos, nesse caso, usar a conversão do formato ".dta"para ".RData"utilizando a função "read.dta()"que aprendemos na sessão anterior.

```r
# Carrega o banco de dados auto.dta:
auto <- read.dta("auto.dta")

# Carrega o banco de dados binlfp2.dta:
binlfp2 <- read.dta("binlfp2.dta")

head(auto)  # Apresenta as 6 primeiras obs do banco de dados
```

```
##             make price mpg rep78 headroom trunk weight length turn
## 1   AMC Concord  4099  22     3      2.5    11   2930    186   40
## 2     AMC Pacer  4749  17     3      3.0    11   3350    173   40
## 3    AMC Spirit  3799  22    NA      3.0    12   2640    168   35
## 4 Buick Century  4816  20     3      4.5    16   3250    196   40
## 5 Buick Electra  7827  15     4      4.0    20   4080    222   43
## 6 Buick LeSabre  5788  18     3      4.0    21   3670    218   43
##   displacement gear_ratio  foreign
## 1          121       3.58 Domestic
## 2          258       2.53 Domestic
## 3          121       3.08 Domestic
## 4          196       2.93 Domestic
## 5          350       2.41 Domestic
## 6          231       2.73 Domestic
```

```r
head(auto, 10) # Apresenta as 10 primeiras
```

```
##             make price mpg rep78 headroom trunk weight length turn
## 1    AMC Concord  4099  22     3      2.5    11   2930    186   40
## 2      AMC Pacer  4749  17     3      3.0    11   3350    173   40
## 3     AMC Spirit  3799  22    NA      3.0    12   2640    168   35
## 4  Buick Century  4816  20     3      4.5    16   3250    196   40
## 5  Buick Electra  7827  15     4      4.0    20   4080    222   43
## 6  Buick LeSabre  5788  18     3      4.0    21   3670    218   43
## 7     Buick Opel  4453  26    NA      3.0    10   2230    170   34
## 8    Buick Regal  5189  20     3      2.0    16   3280    200   42
## 9  Buick Riviera 10372  16     3      3.5    17   3880    207   43
## 10 Buick Skylark  4082  19     3      3.5    13   3400    200   42
##    displacement gear_ratio  foreign
## 1           121       3.58 Domestic
## 2           258       2.53 Domestic
## 3           121       3.08 Domestic
## 4           196       2.93 Domestic
## 5           350       2.41 Domestic
## 6           231       2.73 Domestic
## 7           304       2.87 Domestic
## 8           196       2.93 Domestic
## 9           231       2.93 Domestic
## 10          231       3.08 Domestic
```

5.1 Criando um banco de dados a partir de vetores

Vamos criar um banco de dados de duas formas, primeiro combinando vetores em uma matriz. Depois vamos transformar essa matriz em um banco de dados de fato.

Isso é importante, porque algumas operações com banco de dados podem ser feitas no formato de um objeto matricial, mas outras somente quando a matriz é declarada como banco de dados. Vemos como fazer isso utilizando as seguintes funções: "cbind()"e "data.frame()".

```
# Criando variaveis em forma de vetores:
workshop <- c(1,2,1,2,1,2,1,2)
gender <- c("f","f","f",NA,"m","m","m","m")

q1 <- c(1,2,2,3,4,5,5,4)
q2 <- c(1,1,2,1,5,4,3,5)
q3 <- c(5,4,4,NA,2,5,4,5)
q4 <- c(1,1,3,3,4,5,4,5)

# Criando labels para um fator:
workshop <- factor(workshop,
                   levels=c(1,2,3,4),
                   labels=c("R","Stata","SPSS","SAS"))

# Criando um data frame com os vetores
# workshop,gender,q1,q2,q3 e q4:

# Forma de matriz
mydata <- cbind(workshop,gender,q1,q2,q3,q4)

# Forma de banco de dados:
mydata <- data.frame(mydata)

# Alternativamente:
mydata <- data.frame(workshop,gender,q1,q2,q3,q4)
```

5.2 Criando um banco de dados como subconjunto de outro

Quando você quer trabalhar com partes do banco de dados, pode criar um banco novo, como um subconjunto do original. Isso facilita muito a vida do usuário, especialmente quando o banco original é muito grande. Para tanto, vamos utilizar uma função bastante útil, chamada de "subset()". A funcao "subset()"é utilizada para selecionar subconjuntos do banco de dados. Pode-se utilizar esta função para selecionar casos ou variáveis do banco de dados.

```
# Seleciona apenas as observacoes tais que foreign==1
df1 <- auto[auto$foreign=="Foreign",]

# Alternativamente:
```

```r
df1 <- subset(auto,foreign=="Foreign")

# Seleciona apenas os carros importados com mpg>20:
f1mpg <- subset(auto,foreign=="Foreign" & mpg>20)

# Seleciona apenas as variaveis price e mpg:
f1mpg <- subset(auto,foreign=="Foreign" & mpg>20,select=c(price,mpg))

# Exclui apenas a variavel weight:
b2 <- subset(auto,select=-weight)

head(b2)
```

```
##            make price mpg rep78 headroom trunk length turn
## 1   AMC Concord  4099  22     3      2.5    11    186   40
## 2     AMC Pacer  4749  17     3      3.0    11    173   40
## 3    AMC Spirit  3799  22    NA      3.0    12    168   35
## 4 Buick Century  4816  20     3      4.5    16    196   40
## 5 Buick Electra  7827  15     4      4.0    20    222   43
## 6 Buick LeSabre  5788  18     3      4.0    21    218   43
##   displacement gear_ratio  foreign
## 1          121       3.58 Domestic
## 2          258       2.53 Domestic
## 3          121       3.08 Domestic
## 4          196       2.93 Domestic
## 5          350       2.41 Domestic
## 6          231       2.73 Domestic
```

5.3 Criando uma nova variável num banco já existente

Agora vamos aprender a criar uma nova variável num banco de dados que já existe. Isso é útil quando queremos gerar, por exemplo, uma variável a partir da variável original, com diferente estrutura. Vejamos um exemplo.

```r
# Criar  nova coluna

# Variavel que indique se o carro
# custa mais que 5000 (vari\'avel dummy):
auto$pc <- ifelse(auto$price > 5000, 1, 0)

# Criando uma nova coluna em branco:
auto$vazio <- NA

# Criando uma variavel continua ao quadrado:
auto$price2 <- with(auto,as.numeric(price*price))

# Criando uma variavel log:
```

```
auto$lnprice <-with(auto,as.numeric(log(price)))
```

5.4 Dando nome às variáveis e às suas categorias

Para que você e outras pessoas entendam o seu banco de dados, é importante ter uma etapa de metadados. Essa etapa pode ser feita parcialmente dentro do próprio banco de dados. Para tanto, utilizamos dois passos: dar nomes (rótulos) ás variáveis e dar rótulos às categorias das variáveis quando essas são categóricas. Vamos aos exemplos práticos:

```
# Criando rotulos para as variaveis:
install.packages("Hmisc")
library(Hmisc)

##
## Attaching package:  'Hmisc'
##
## The following objects are masked from 'package:base':
##
##    format.pval, round.POSIXt, trunc.POSIXt, units
```

Carregado o pacote "Hmisc", vamos agora atribuir os rótulos:

```
# Rotulando individualmente cada variavel
label(mydata$workshop) <- "Workshop que participou"
label(mydata$gender) <- "Genero"
label(mydata$q1) <- "Questao 1"
label(mydata$q2) <- "Questao 2"
label(mydata$q3) <- "Questao 3"
label(mydata$q4) <- "Questao 4"

# Para verificar os nomes das variaveis
names(mydata)

## [1] "workshop" "gender"   "q1"       "q2"       "q3"       "q4"

# Para ver os rotulos atribuidos
describe(mydata)

## mydata
##
##  6  Variables      8  Observations
## ---------------------------------------------------------------------------
## workshop : Workshop que participou
##       n missing  unique
##       8       0       2
##
## R (4, 50%), Stata (4, 50%)
```

```
## ---------------------------------------------------------------------
## gender : Genero
##       n missing  unique
##       7       1       2
##
## f (3, 43%), m (4, 57%)
## ---------------------------------------------------------------------
## q1 : Questao 1
##       n missing  unique    Info    Mean
##       8       0       5    0.96    3.25
##
##            1  2  3  4  5
## Frequency  1  2  1  2  2
## %         12 25 12 25 25
## ---------------------------------------------------------------------
## q2 : Questao 2
##       n missing  unique    Info    Mean
##       8       0       5    0.94    2.75
##
##            1  2  3  4  5
## Frequency  3  1  1  1  2
## %         38 12 12 12 25
## ---------------------------------------------------------------------
## q3 : Questao 3
##       n missing  unique    Info    Mean
##       7       1       3    0.86   4.143
##
## 2 (1, 14%), 4 (3, 43%), 5 (3, 43%)
## ---------------------------------------------------------------------
## q4 : Questao 4
##       n missing  unique    Info    Mean
##       8       0       4    0.95    3.25
##
## 1 (2, 25%), 3 (2, 25%), 4 (2, 25%), 5 (2, 25%)
## ---------------------------------------------------------------------

# Criando rotulos para as categorias das variaveis:
workshop <- factor(workshop,
                   levels=c(1,2,3,4),
                   labels=c("R","Stata","SPSS","SAS"))
```

5.5 Agregando valores de unidades menores em maiores

Se quisermos criar um banco de dados agregado, por exemplo, com média ou soma dos valores do banco desagregado, basta utilizarmos a função "aggregate()".

Imagine, por exemplo, que você esteja trabalhando com o banco de dados em que cada linha corresponde a um município, com informações no nível municipal. No en-

tanto, você quer transformar esse banco em um banco com informações por Estado ou Micrroregião. Um caso típico seria um banco com renda municipal média por estado da Federação. Vejamos como isso é feito utilizando o banco de dados de automóveis que estamos trabalhando:

```
# Agrega os valores de
# price, mpg, trunk, displacement, length, weight, foreign
# por rep78

# Agregacao por soma dos valores
x <- with(auto,
          cbind(price,mpg,trunk,displacement,length,weight,foreign)
          )
rep78sum <- with(auto,
                 aggregate(x,by=list(rep78=rep78),FUN="sum")
                 )

rep78sum

##   rep78  price mpg trunk displacement length weight foreign
## 1     1   9129  42    17          382    378   6200       2
## 2     2  47741 153   117         1938   1595  26830       8
## 3     3 192877 583   458         6901   5820  98970      33
## 4     4 109287 390   243         3219   3327  51660      27
## 5     5  65043 301   126         1222   1872  25550      20

# Agregacao por media (proporcao) dos valores
rep78mean <- with(auto,
                  aggregate(x,by=list(rep78=rep78),FUN="mean")
                  )

rep78mean

##   rep78    price      mpg    trunk displacement   length   weight
## 1     1 4564.500 21.00000  8.50000     191.0000 189.0000 3100.000
## 2     2 5967.625 19.12500 14.62500     242.2500 199.3750 3353.750
## 3     3 6429.233 19.43333 15.26667     230.0333 194.0000 3299.000
## 4     4 6071.500 21.66667 13.50000     178.8333 184.8333 2870.000
## 5     5 5913.000 27.36364 11.45455     111.0909 170.1818 2322.727
##    foreign
## 1 1.000000
## 2 1.000000
## 3 1.100000
## 4 1.500000
## 5 1.818182
```

5.6 Juntanto banco de dados

A junção de banco de dados pode ocorrer de várias formas. Veja a figura abaixo:

Fonte: De Vries & Meys (2012) - R for Dummies

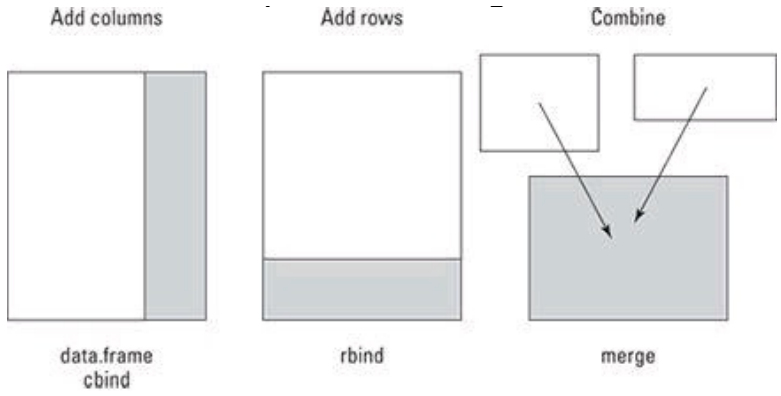


Figura 1: Modos Principais de Juntar Bancos no R

Vamos nos concentrar em dois casos principais. O terceiro, listado abaixo, já foi visto acima:

- Junção 1 por 1 (exemplo: pessoa para pessoa)
- Junção 1 para muitos (exemplo: domicilio para pessoa)
- Junção muitos para 1 (função "agregate()"vista acima)

5.6.1 Junção 1 para 1

Vamos fazer primeiramente o caso 1 por 1. Esse é o caso mais simples. Imagine que você tem um banco com 50 indivíduos, com as seguintes informações:

- id: identificador único do indivíduo
- nconsultas: número de consultas médicas no último ano
- sexo: sexo do indivíduo

Agora você quer agregar mais informações desses indivíduos que você deixou de fora do banco. Vamos juntar mais duas informações: idade e consultas ao dentista. Veja que esse último banco possui informação apenas para 13 daqueles 50 indivíduos. Assim, podemos juntar os dois bancos, mas idade e consulta ao dentista apresentará valor NA para os demais 37 casos. Vejamos agora como isso é feito:

```
# Cria um data frame para 50 pessoas
pessoas_a <- data.frame(id=1:50,nconsultas=rpois(50, lambda=3),
                        sexo=rbinom(50,1,0.4))

# Cria um data frame para 13 pessoas
pessoas_b <- data.frame(pessoa=3:15,ndentista=rpois(13, lambda=1),
                        idade=round(rnorm(13,35,10),digits=0))

# Mescla os dois bancos atravas das variaveis:
# id para banco: pessoas_a
```

```r
# pessoa para banco: pessoas_b
pessoas <- merge(x=pessoas_a, # banco 1
           y=pessoas_b, # banco 2
           by.x="id", # variavel que esta nos 2 bancos
           by.y="pessoa",
           all=TRUE) # Todas, mesmo que nao paream

head(pessoas, 10)

##    id nconsultas sexo ndentista idade
## 1   1          3    1        NA    NA
## 2   2          4    0        NA    NA
## 3   3          5    0         1    27
## 4   4          5    0         0    37
## 5   5          1    0         1    26
## 6   6          5    0         2    40
## 7   7          7    1         1    41
## 8   8          4    0         1    33
## 9   9          2    1         0    38
## 10 10          2    0         1    32
```

5.6.2 Junção Muitos para 1

Vamos imaginar agora que queremos acrescentar características dos domicílios desses indivíduos. Por exemplo, podemos incorporar informações no nível domiciliar do tipo:

- lixo: se tem coleta regular
- esgoto: se tem rede geral
- npess: número de pessoas no domicílio

Nesse caso, indivíduos que moram no mesmo domicílio teria o identificador do domicílio repetido. Essa é a situação de muitos para 1. Vejamos como fazer isso:

```r
# Criando um indicador de domicilio no banco pessoas:
pessoas$dom <- rep(1:15, times=c(1,5,3,7,1,2,8,3,5,4,2,4,2,1,2))

# Criando o banco de dados do domicilio

domicilios <- data.frame(dom=1:15,
                         lixo=rbinom(15,1,0.7),
                         esgoto=rbinom(15,1,0.8),
                         renddom=rnorm(15,1000))

# Trazendo informacoes do domicilio
# para o nivel de pessoas
pessoas <- merge(x=pessoas, # banco 1
           y=domicilios, # banco 2
```

```
            by.x="dom", # variavel que esta nos 2 bancos
            by.y="dom",
            all.x=TRUE) # Todas as pessoas, mesmo que nao paream
```

6 Estatística Descritiva

Nesta sessão você aprenderá a descrever os seus dados com três instrumentos principais:

- Tabelas simples e cruzadas
- Gráficos
- Medidas descritivas

Vejamos cada um deles a seguir. Para maiores detalhes, veja De Vries e Meyers (2012) ou Muenchen e Hilbe (2010).

6.1 Tabelas

Vamos apresentar as seguintes tabelas:

- Frequência simples
- Frequência cruzada

6.1.1 Tabela de Frequência Simples

```
# Tabela de frequencia absoluta de rep78
table(auto$rep78)

##
##  1  2  3  4  5
##  2  8 30 18 11

# Tabela de frequencia absoluta,
# relativa e acumulada de rep78
tabs <- function(x){
  x <- na.omit(x)
  return(cbind(Nome=unique(sort(x)),
               Fabs=table(x),
               Frel=round(prop.table(table(x)),digits=2),
               FAcum=cumsum(table(x)),
               FrelAcum=round(cumsum(prop.table(table(x))),digits=2)))
}

with(auto,tabs(rep78))
```

```
##   Nome Fabs Frel FAcum FrelAcum
## 1    1    2 0.03     2     0.03
## 2    2    8 0.12    10     0.14
## 3    3   30 0.43    40     0.58
## 4    4   18 0.26    58     0.84
## 5    5   11 0.16    69     1.00
```

6.1.2 Tabela de Frequência Cruzada

Apresentamos os seguintes tipos de tabela:

- Frequência absoluta cruzada
- Frequência absoluta cruzada com marginais
- Frequência (absoluta e relativa) cruzada com soma na diagonal
- Frequência (absoluta e relativa) cruzada com soma na linha
- Frequência (absoluta e relativa) cruzada com soma na coluna

Frequência absoluta cruzada

```
# Tabela de frequencia de workshop por genero
with(auto,table(rep78,foreign))
```

```
##      foreign
## rep78 Domestic Foreign
##     1        2       0
##     2        8       0
##     3       27       3
##     4        9       9
##     5        2       9
```

Frequência absoluta cruzada com marginais

```
# Tabela Cruzada de Workshop e Gender com marginais:
with(auto,addmargins(table(rep78,foreign)))
```

```
##      foreign
## rep78 Domestic Foreign Sum
##   1          2       0   2
##   2          8       0   8
##   3         27       3  30
##   4          9       9  18
##   5          2       9  11
##   Sum       48      21  69
```

Frequência (absoluta e relativa) cruzada com soma na diagonal

```r
library("gmodels")

with(auto,CrossTable(rep78,foreign,format="SAS",prop.t=TRUE))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  69
##
##
##              | foreign
##        rep78 |  Domestic |   Foreign | Row Total |
## -------------|-----------|-----------|-----------|
##            1 |         2 |         0 |         2 |
##              |     0.266 |     0.609 |           |
##              |     1.000 |     0.000 |     0.029 |
##              |     0.042 |     0.000 |           |
##              |     0.029 |     0.000 |           |
## -------------|-----------|-----------|-----------|
##            2 |         8 |         0 |         8 |
##              |     1.065 |     2.435 |           |
##              |     1.000 |     0.000 |     0.116 |
##              |     0.167 |     0.000 |           |
##              |     0.116 |     0.000 |           |
## -------------|-----------|-----------|-----------|
##            3 |        27 |         3 |        30 |
##              |     1.801 |     4.116 |           |
##              |     0.900 |     0.100 |     0.435 |
##              |     0.562 |     0.143 |           |
##              |     0.391 |     0.043 |           |
## -------------|-----------|-----------|-----------|
##            4 |         9 |         9 |        18 |
##              |     0.990 |     2.264 |           |
##              |     0.500 |     0.500 |     0.261 |
##              |     0.188 |     0.429 |           |
##              |     0.130 |     0.130 |           |
## -------------|-----------|-----------|-----------|
##            5 |         2 |         9 |        11 |
##              |     4.175 |     9.543 |           |
```

```
##              |     0.182 |     0.818 |     0.159 |
##              |     0.042 |     0.429 |           |
##              |     0.029 |     0.130 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        48 |        21 |        69 |
##              |     0.696 |     0.304 |           |
## -------------|-----------|-----------|-----------|
##
##
```

Frequência (absoluta e relativa) cruzada com soma na linha

```
with(auto,CrossTable(rep78,foreign,format="SAS",prop.r=TRUE))

##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  69
##
##
##              | foreign
##       rep78 |  Domestic |   Foreign | Row Total |
## -------------|-----------|-----------|-----------|
##            1 |         2 |         0 |         2 |
##              |     0.266 |     0.609 |           |
##              |     1.000 |     0.000 |     0.029 |
##              |     0.042 |     0.000 |           |
##              |     0.029 |     0.000 |           |
## -------------|-----------|-----------|-----------|
##            2 |         8 |         0 |         8 |
##              |     1.065 |     2.435 |           |
##              |     1.000 |     0.000 |     0.116 |
##              |     0.167 |     0.000 |           |
##              |     0.116 |     0.000 |           |
## -------------|-----------|-----------|-----------|
##            3 |        27 |         3 |        30 |
##              |     1.801 |     4.116 |           |
##              |     0.900 |     0.100 |     0.435 |
##              |     0.562 |     0.143 |           |
```

```
##              |     0.391 |     0.043 |           |
## -------------|-----------|-----------|-----------|
##            4 |         9 |         9 |        18 |
##              |     0.990 |     2.264 |           |
##              |     0.500 |     0.500 |     0.261 |
##              |     0.188 |     0.429 |           |
##              |     0.130 |     0.130 |           |
## -------------|-----------|-----------|-----------|
##            5 |         2 |         9 |        11 |
##              |     4.175 |     9.543 |           |
##              |     0.182 |     0.818 |     0.159 |
##              |     0.042 |     0.429 |           |
##              |     0.029 |     0.130 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        48 |        21 |        69 |
##              |     0.696 |     0.304 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```
# Por linha, sem valores abolutos:
with(auto,round(prop.table(table(rep78,foreign),margin=1),digits=2))
```

```
##      foreign
## rep78 Domestic Foreign
##     1     1.00    0.00
##     2     1.00    0.00
##     3     0.90    0.10
##     4     0.50    0.50
##     5     0.18    0.82
```

Frequência (absoluta e relativa) cruzada com soma na coluna

```
with(auto,CrossTable(rep78,foreign,format="SAS",prop.c=TRUE))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  69
```

```
##
##
##              | foreign
##       rep78 |  Domestic |   Foreign | Row Total |
## -------------|-----------|-----------|-----------|
##            1 |         2 |         0 |         2 |
##              |     0.266 |     0.609 |           |
##              |     1.000 |     0.000 |     0.029 |
##              |     0.042 |     0.000 |           |
##              |     0.029 |     0.000 |           |
## -------------|-----------|-----------|-----------|
##            2 |         8 |         0 |         8 |
##              |     1.065 |     2.435 |           |
##              |     1.000 |     0.000 |     0.116 |
##              |     0.167 |     0.000 |           |
##              |     0.116 |     0.000 |           |
## -------------|-----------|-----------|-----------|
##            3 |        27 |         3 |        30 |
##              |     1.801 |     4.116 |           |
##              |     0.900 |     0.100 |     0.435 |
##              |     0.562 |     0.143 |           |
##              |     0.391 |     0.043 |           |
## -------------|-----------|-----------|-----------|
##            4 |         9 |         9 |        18 |
##              |     0.990 |     2.264 |           |
##              |     0.500 |     0.500 |     0.261 |
##              |     0.188 |     0.429 |           |
##              |     0.130 |     0.130 |           |
## -------------|-----------|-----------|-----------|
##            5 |         2 |         9 |        11 |
##              |     4.175 |     9.543 |           |
##              |     0.182 |     0.818 |     0.159 |
##              |     0.042 |     0.429 |           |
##              |     0.029 |     0.130 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        48 |        21 |        69 |
##              |     0.696 |     0.304 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```r
# Por coluna, sem valores absolutos:
with(auto,round(prop.table(table(rep78,foreign),margin=2),digits=2))
```

```
##      foreign
## rep78 Domestic Foreign
##     1     0.04    0.00
##     2     0.17    0.00
```

```
##      3     0.56    0.14
##      4     0.19    0.43
##      5     0.04    0.43
```

6.2 Gráficos

O R possui muitos pacotes para que o usuário possa produzir gráficos, incluindo gráficos dinâmicos (como no pacote "iplots"), tridimensionais (como no pacote "rgoobi"), para confecção de mapas (como no pacote "maps") e para representação de dados categóricos (como no pacote "vcd"). Esses pacotes estão fora do escopo dessa apostila.

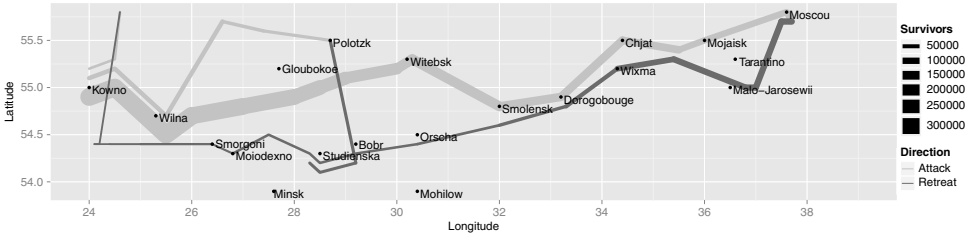
Aqui iremos nos concentrar em apenas um dos três principais tipos de gráficos do R, que são:

- tradicional
- lattice
- ggplot2

Os gráficos tradicionais do R incluem funções de alto e baixo nível para desenhar atributos como pontos, linhas e eixos. Essas funções de baixo nível provêem flexibilidade e controle para gerar quase qualquer tipo de gráfico que não esteja pré-definido.

O pacote "ggplot2", escrito por Hadley Wickham, oferece um ótimo balanço entre poder e facilidade de uso. Através de sua utilização, o famoso gráfico do avanço e retração do exército de Napoleão para Moscow pode ser desenhado. As linhas mais claras representam o avanço, as mais escuras a retração, e a espessura das linhas representa o contingente do exército a cada ponto.

Fonte: Muenchen e Hilbe (2010): R for Stata Users



A Tabela 6.2, apresentada abaixo, mostra as potencialidades e limitações de cada um dos tipos de pacotes gráficos:

Fonte: Muenchen e Hilbe (2010): R for Stata Users

	Traditional (or base)	`lattice`	`ggplot2`
Automatic output for different objects	Yes	No	No
Automatic legends	No	Sometimes	Yes
Easily repeats plots for different groups	No	Yes	Yes
Easy to use with multiple data sources	Yes	No	Yes
Allows you to build plots piece-by-piece	Yes	No	Yes
Allows you to replace pieces after creation	No	No	Yes
Consistent functions	No	No	Yes
Attractiveness of default settings	Good	Good	Excellent
Can do mosaic plots	Yes	Yes	No
Control extends beyond data graphics	Yes	No	No
Underlying graphics system	Traditional	Grid	Grid

6.2.1 Procedimentos e Sistemas Gráficos

Os gráficos possuem funcionalidades de baixo nível, chamados de *sistemas gráficos*, os quais permitem controlar e alterar ajustes que afetam todos os tipos de gráficos, como:

- fontes de texto
- padrões de preenchimento
- tipos de linhas ou pontos

No nível mais básico do gerenciamento de gráficos estão os mecanismos gráficos. A seguir apresentamos algums tipos de mecanismos mais comuns dos gráficos em R:

1. Ativando produção de gráficos salvos em formato "Encapsulated PostSCript File":

```
postscript(file="meugrafico.eps",
           paper="special",
           width=4,
           height=3.5)    # Abre o mecanismo

plot(meusdados$pretest,meusdados$posttest)    # Produz o grafico

dev.off()                 # Fecha o mecanismo

## pdf
##   2
```

2. Ativando produção de gráficos salvos em formato "Portable Network Graphics":

```
# PNG permite transparencia, pois eps nao permite

png(file="transparencyDemo.png",
    res=600,         # resolucao em dpi (dots per inch)
    width=2400,      # 2400 dpi a 600 dpi --> 2400/600 = 4 inches
    height=2100)     # 2100 dpi a 600 dpi --> 2100/600 = 3.5 inches
```

```
plot(meusdados$pretest,meusdados$posttest)   # Faz o grafico de dispersao

dev.off()                    # Fecha o device

## pdf
##   2
```

3. Ativando produção de gráficos salvos em formato "Windows Meta File":

```
# Sistema grafico para Windows Meta File (.wmf)

#win.metafile(file="myPlot%03d.wmf")

#barplot( table(meusdados£workshop)) #Graf1 para myPlot 1.wmf

#hist(meusdados£posttest) #Graf2 para myPlot 2.wmf

#plot(meusdados£pretest,meusdados£posttest) #Graf3 para myPlot 3.wmf

#dev.off()
```

4. Ativando produção de gráficos salvos em formato "Portable Document File":

```
x <- 1:100
y <- 0.029*x + rnorm(100)

pdf("sample.pdf",
    width=7,
    height=5)
plot(x, y, pch=19, col=rgb(0.5, 0.5, 0.5, 0.5), cex=1.5)
abline(lm(y ~ x))
dev.off()

## pdf
##   2
```

A seguir vamos apresentar uma série de gráficos que serão utilizados para representar variáveis de diferentes naturezas (categóricas, ordinais e contínuas). Utilizaremos os gráficos básicos do R, pois apesar de mais difíceis, eles são mais flexíveis.

6.2.2 Gráficos Tradicionais

A maioria dos gráficos tradicionais do R precisa que você sumarize as informações primeiro. No Stata, fazer um gráfico de barras é so digitar "graph bar *var*". O Stata já entende que os dados precisam ser sumarizados. No R é o oposto. Vamos aos exemplos:

- Gráficos de Barras (utilizado para variáveis categóricas):

```
# Grafico de barras de contagens
par(mfrow=c(1,2))
barplot( c(40,60), main="Grafico de Barras" )

# Grafico de barras de cada observacao da variavel q4
load("/Volumes/NO NAME/ALMG/mydata100.RData")
#barplot(mydata100£q4)
# Nao e um bom grafico, porque nao me diz nada

# Para fazer o grafico, usar a funcao table() que gera a contagem
table(mydata100$q4)


##
##  1  2  3  4  5
##  6 14 34 26 20

barplot(table(mydata100$q4),
        main="Distribuicao de q4")

# Grafico de barras horizontal
par(mfrow=c(1,2))
# Simples, incluindo todas as categorias de workshop
barplot(table(mydata100$workshop), horiz=T,
        main="Participantes\npor Workshop")

# Simples, sumindo uma das categorias de workshop
barplot(
  table(mydata100$workshop[mydata100$workshop!="Stata"])
  [table(mydata100$workshop[mydata100$workshop!="Stata"])!=0],
  horiz=TRUE,
  main="Participantes\npor Workshop\n(sem Stata)")

# Grafico de Barras de contagem por subgroupo
par(mfrow=c(1,1))
barplot( table(mydata100$gender,mydata100$workshop),
         main="Frequencia no Workshop\npor sexo",
         ylab="Numero de pessoas",
         xlab="Workshop",
         col=c("gray90","gray60"))
legend("topright",
        c("Mulher","Homem"),
        fill=c("gray90","gray60"))

# Distribuicao Media de Q1 por Workshop e Genero
myMeans <- tapply(mydata100$q1,
                  list(mydata100$gender,mydata100$workshop),
```

```
                  mean,na.rm=T)
par(mfrow=c(2,2))
barplot(myMeans,
        main="Media de Q1\npor Workshop e Genero",
        ylab="Media de Q1",
        xlab="Workshop",
        col=c("gray90","gray60"),
        ylim=c(0,10))
legend("topright",
        c("Mulher","Homem"),
        fill=c("gray90","gray60"))

barplot(myMeans,
        main="Media de Q1\npor Workshop e Genero",
        ylab="Media de Q1",
        xlab="Workshop",
        col=c("gray90","gray60"),
        ylim=c(0,10),
        beside=T)
legend("topright",
        c("Mulher","Homem"),
        fill=c("gray90","gray60"))

# Distribuicao Media de Q1 por Genero e Workshop
myMeans <- tapply(mydata100$q1,
                  list(mydata100$workshop,mydata100$gender),
                  mean,na.rm=T)
barplot(myMeans,
        main="Media de Q1\npor Genero e Workshop",
        ylab="Media de Q1",
        xlab="Workshop",
        col=c("gray90","gray70","gray50","gray30"),
        ylim=c(0,20),
        beside=F)
legend("topright",
        c("R","SAS","SPSS","Stata"),
        fill=c("gray90","gray70","gray50","gray30"))

barplot(myMeans,
        main="Media de Q1\npor Genero e Workshop",
        ylab="Media de Q1",
        xlab="Workshop",
        col=c("gray90","gray70","gray50","gray30"),
        ylim=c(0,10),
        beside=T)
legend("topright",
        c("R","SAS","SPSS","Stata"),
        fill=c("gray90","gray70","gray50","gray30"))
```

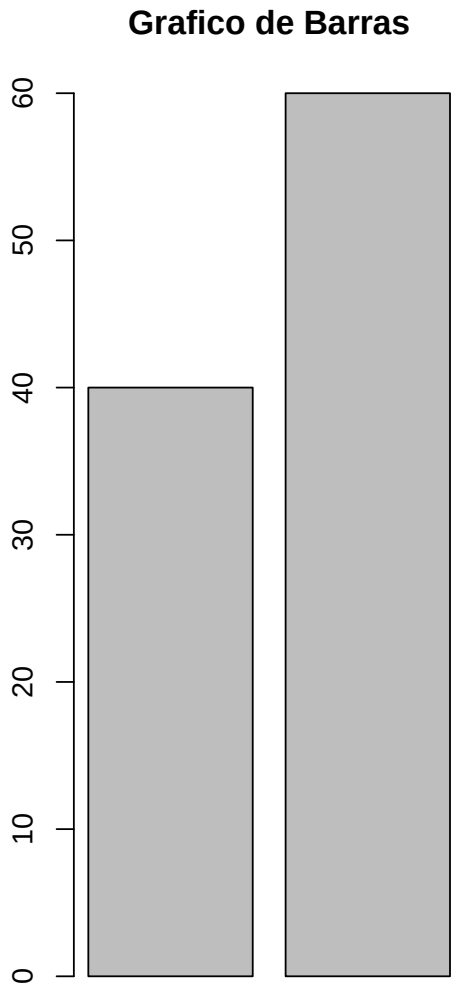
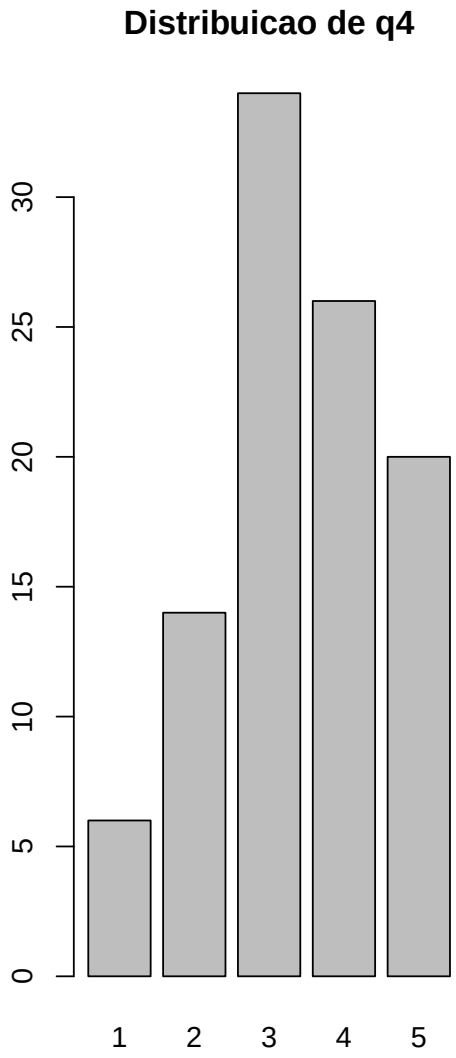
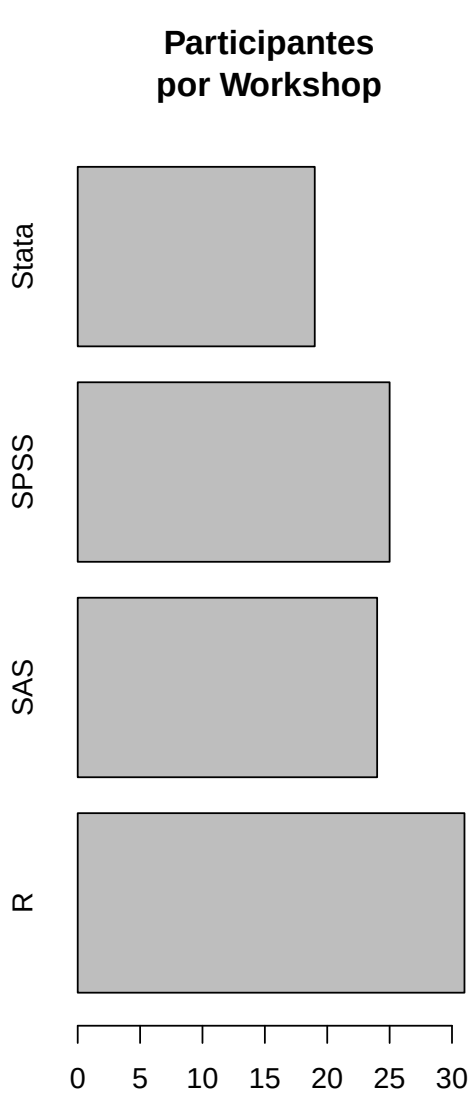
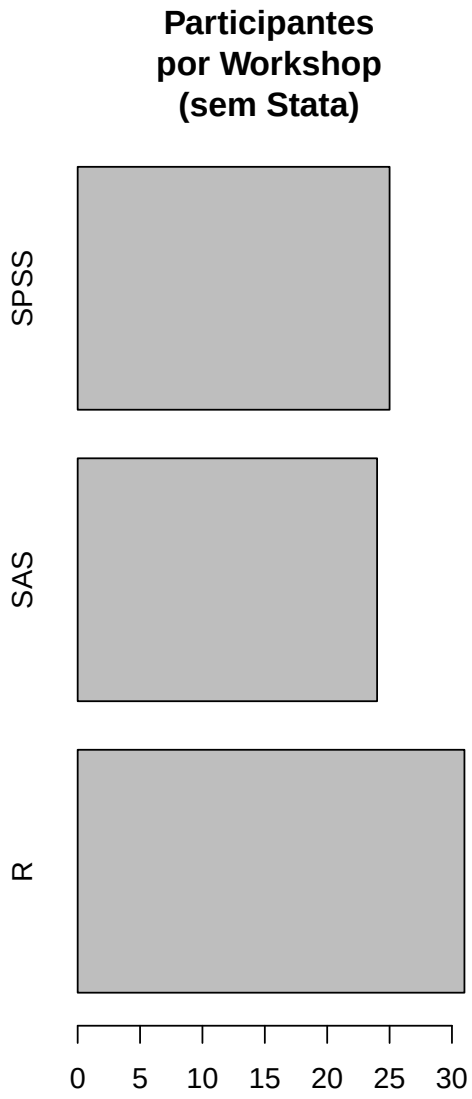
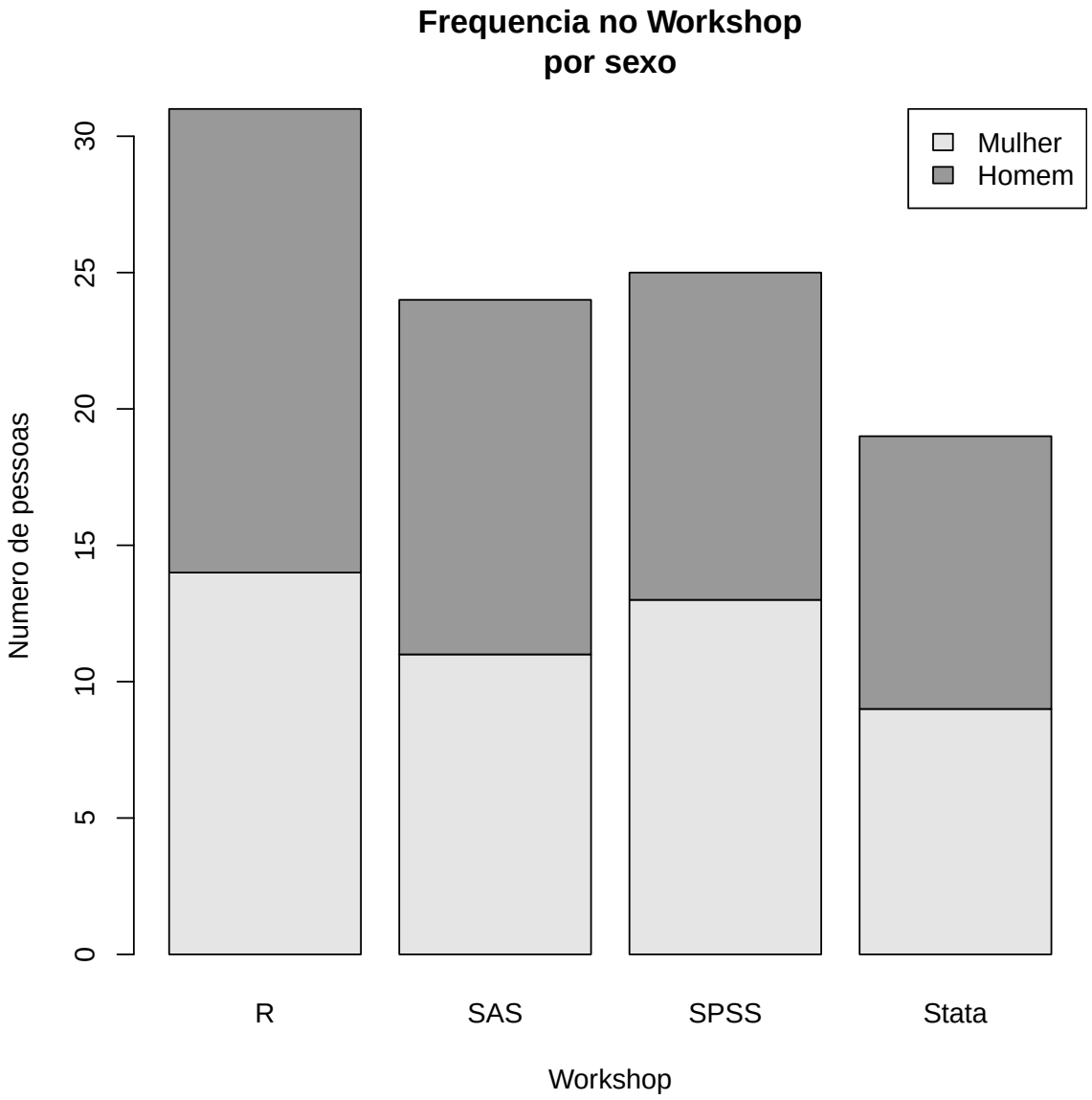


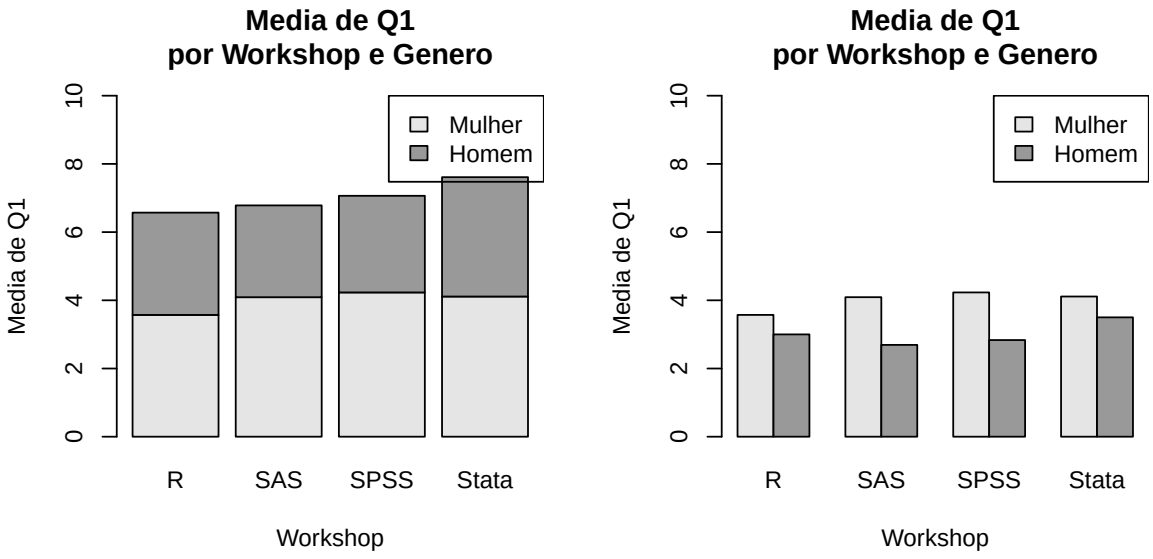
Grafico de Barras
0
10
20
30
40
50
60

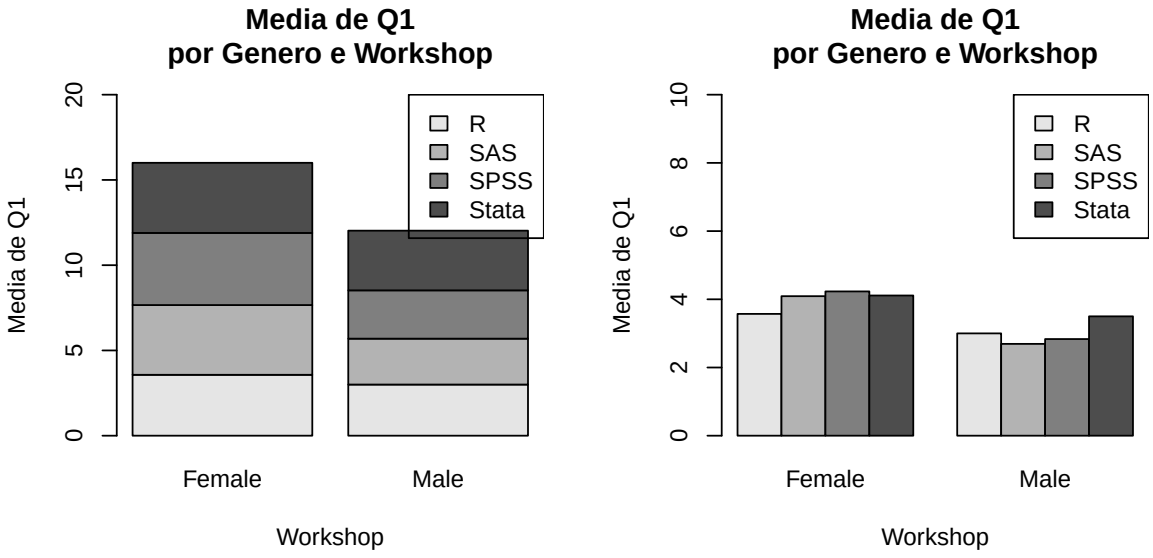

Distribuicao de q4
0
5
10
15
20
25
30
1
2
3
4
5


Participantes
por Workshop
Stata
SPSS
SAS
R
0
5
10
15
20
25
30


Participantes
por Workshop
(sem Stata)
SPSS
SAS
R
0
5
10
15
20
25
30

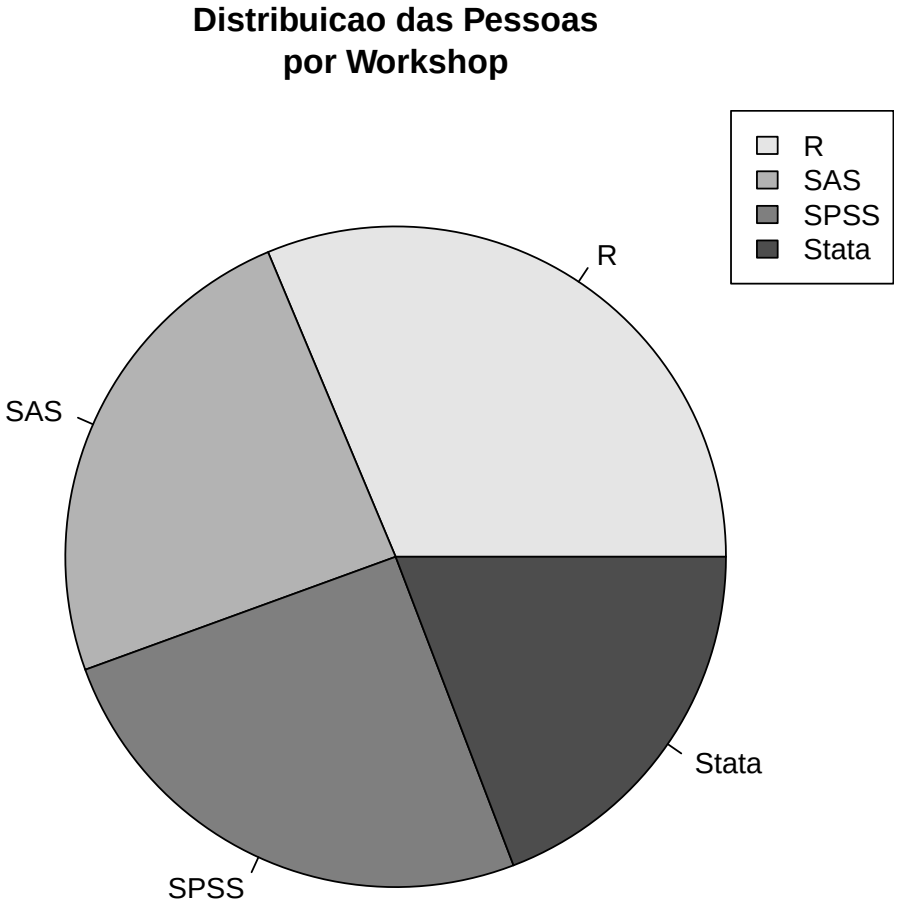

Frequencia no Workshop
por sexo
Mulher
Homem
Numero de pessoas
0
5
10
15
20
25
30
R
SAS
SPSS
Stata
Workshop





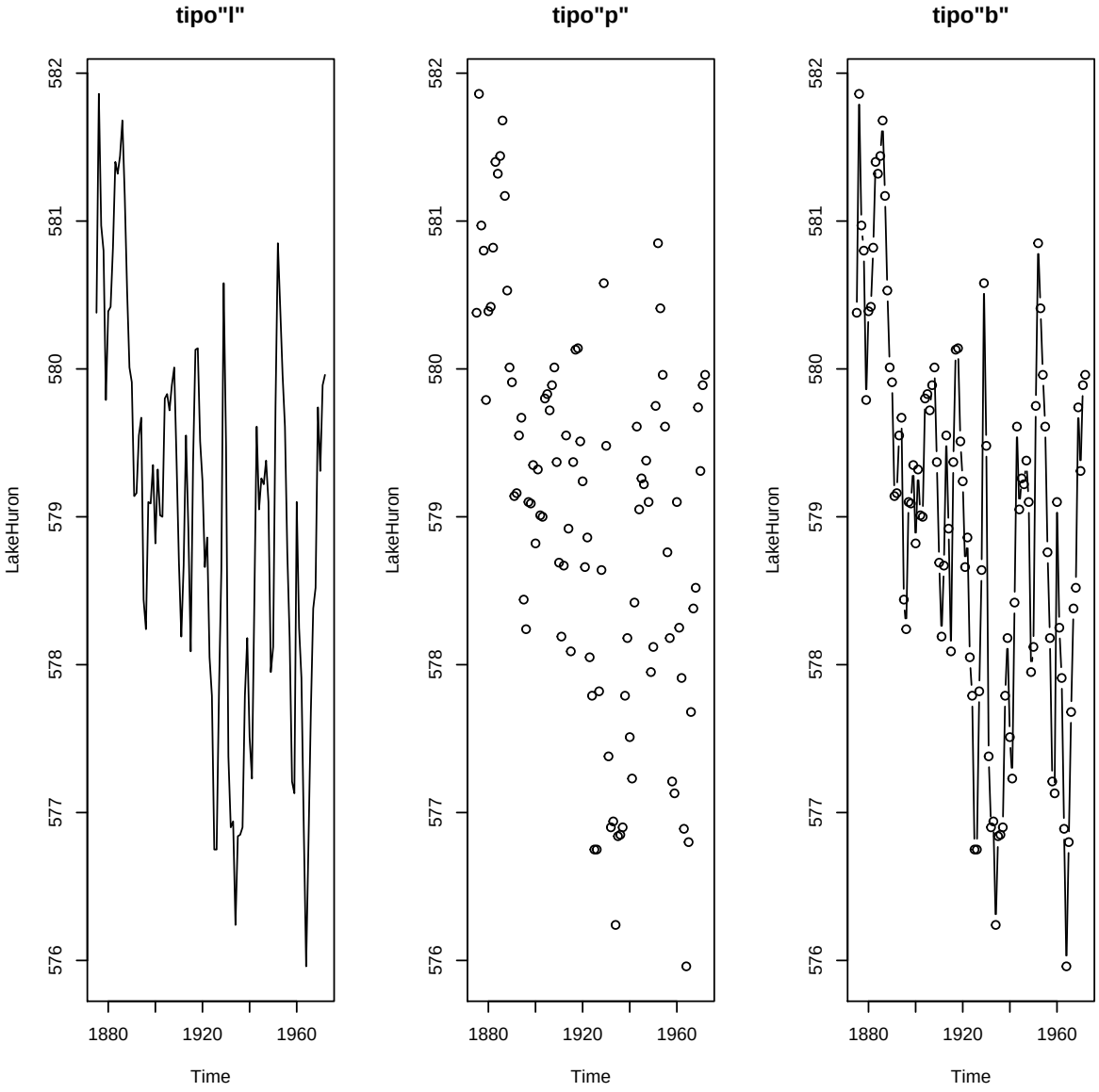
- Gráficos de Pizza (utilizado para variáveis categóricas):

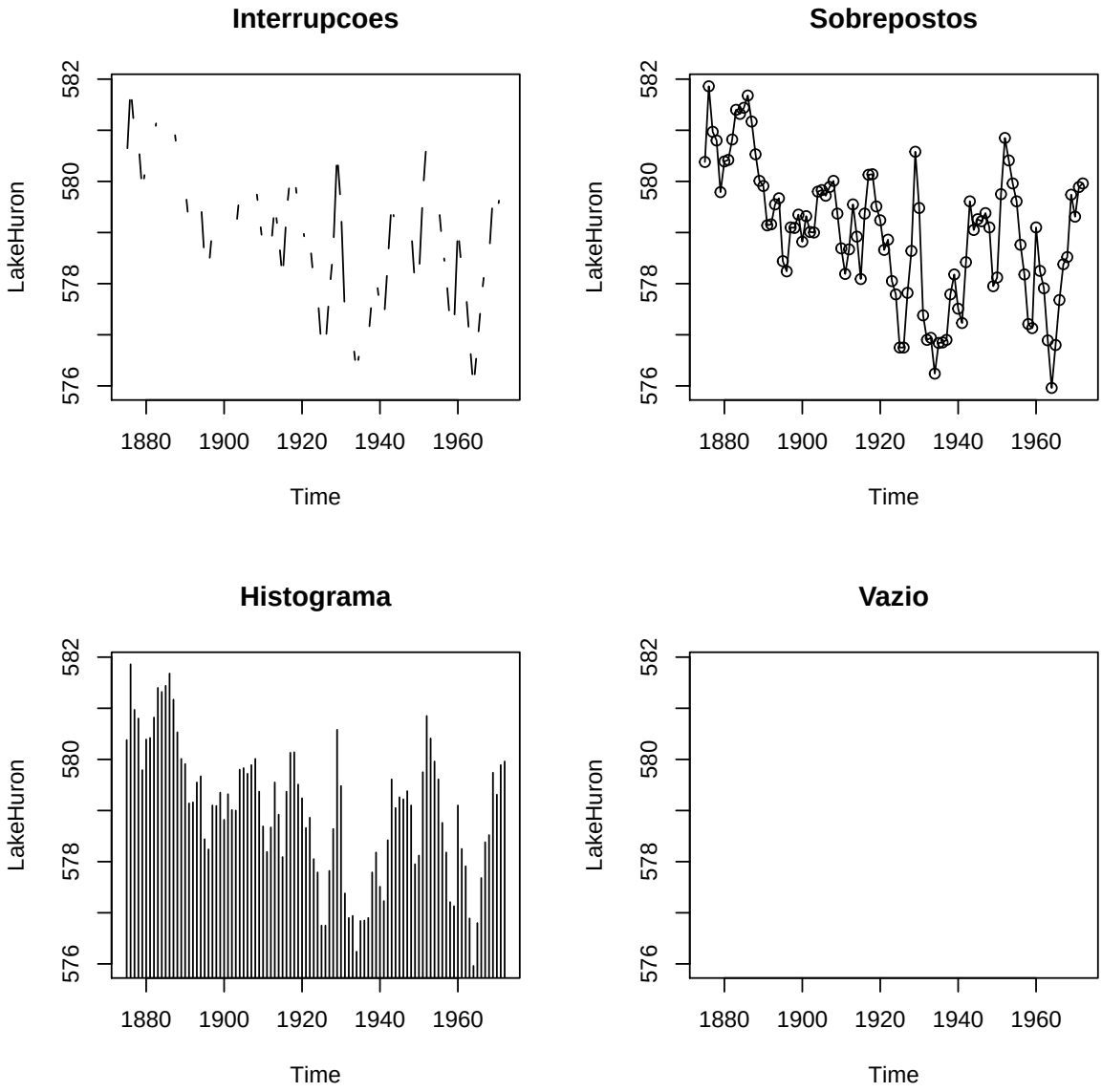
```
# Graficos de Pizza
par(mfrow=c(1,1))
pie(table(mydata100$workshop),
    col=c("gray90","gray70","gray50","gray30"),
    main="Distribuicao das Pessoas\npor Workshop")
legend("topright",
       c("R","SAS","SPSS","Stata"),
       fill=c("gray90","gray70","gray50","gray30"))
```



- Gráficos de Linhas (utilizados para variáveis contínuas, especialmente para séries temporais):

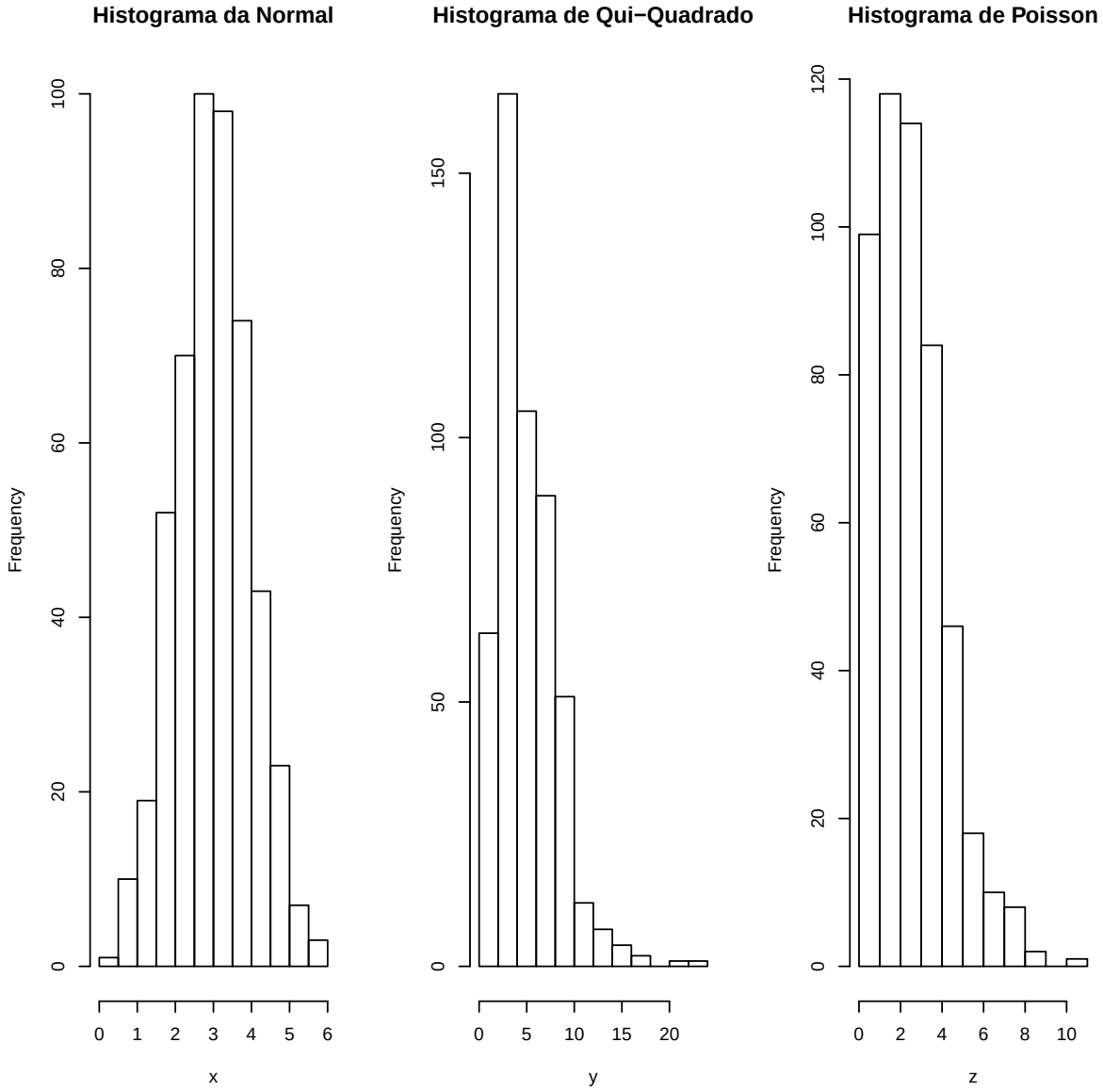
```
# Graficos de Linhas
par(mfrow=c(1,3))
plot(LakeHuron, type="l", main='tipo"l"') # l --> linha
plot(LakeHuron, type="p", main='tipo"p"') # p --> pontos
plot(LakeHuron, type="b", main='tipo"b"') # b --> ambos

par(mfrow=c(2,2))
plot(LakeHuron,type="c",main="Interrupcoes") # Linha parte do ponto
plot(LakeHuron,type="o",main="Sobrepostos") # Ambos sobrepostos
plot(LakeHuron,type="h",main="Histograma") # Tipo histograma
plot(LakeHuron,type="n",main="Vazio") # Grafico vazio
```


tipo"l"
tipo"p"
tipo"b"
LakeHuron
Time
576
577
578
579
580
581
582
1880
1920
1960



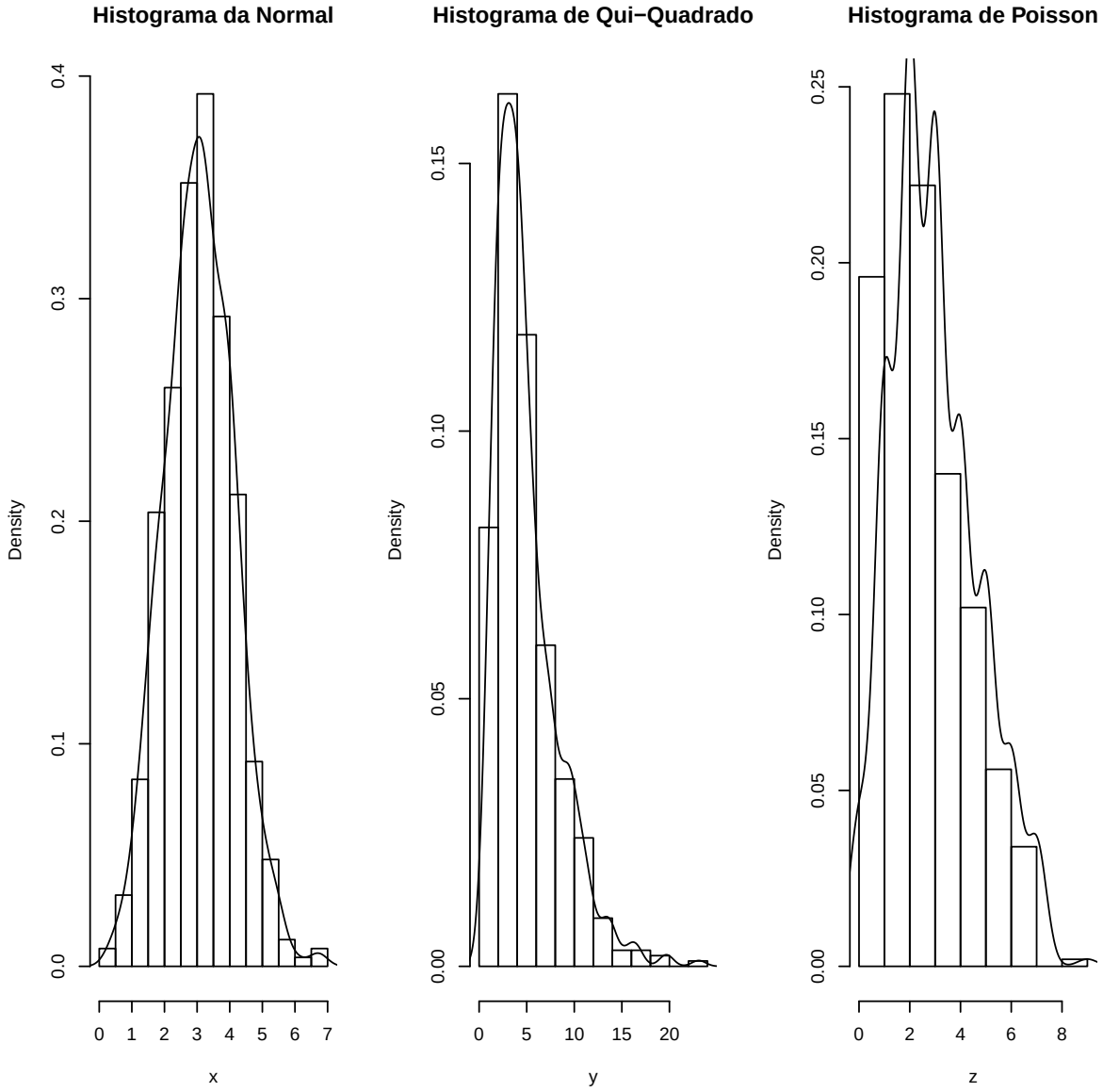
- Histogramas (utilizados para variáveis contínuas):

```
# Histogramas de Distribuicoes (counts)
x <- rnorm(500,mean=3,sd=1)
y <- rchisq(500,df=5)
z <- rpois(500,lambda=3)
par(mfrow=c(1,3))
hist(x, main="Histograma da Normal", breaks=10, probability=F)
hist(y, main="Histograma de Qui-Quadrado", breaks=10, probability=F)
hist(z, main="Histograma de Poisson", breaks=10, probability=F)
```



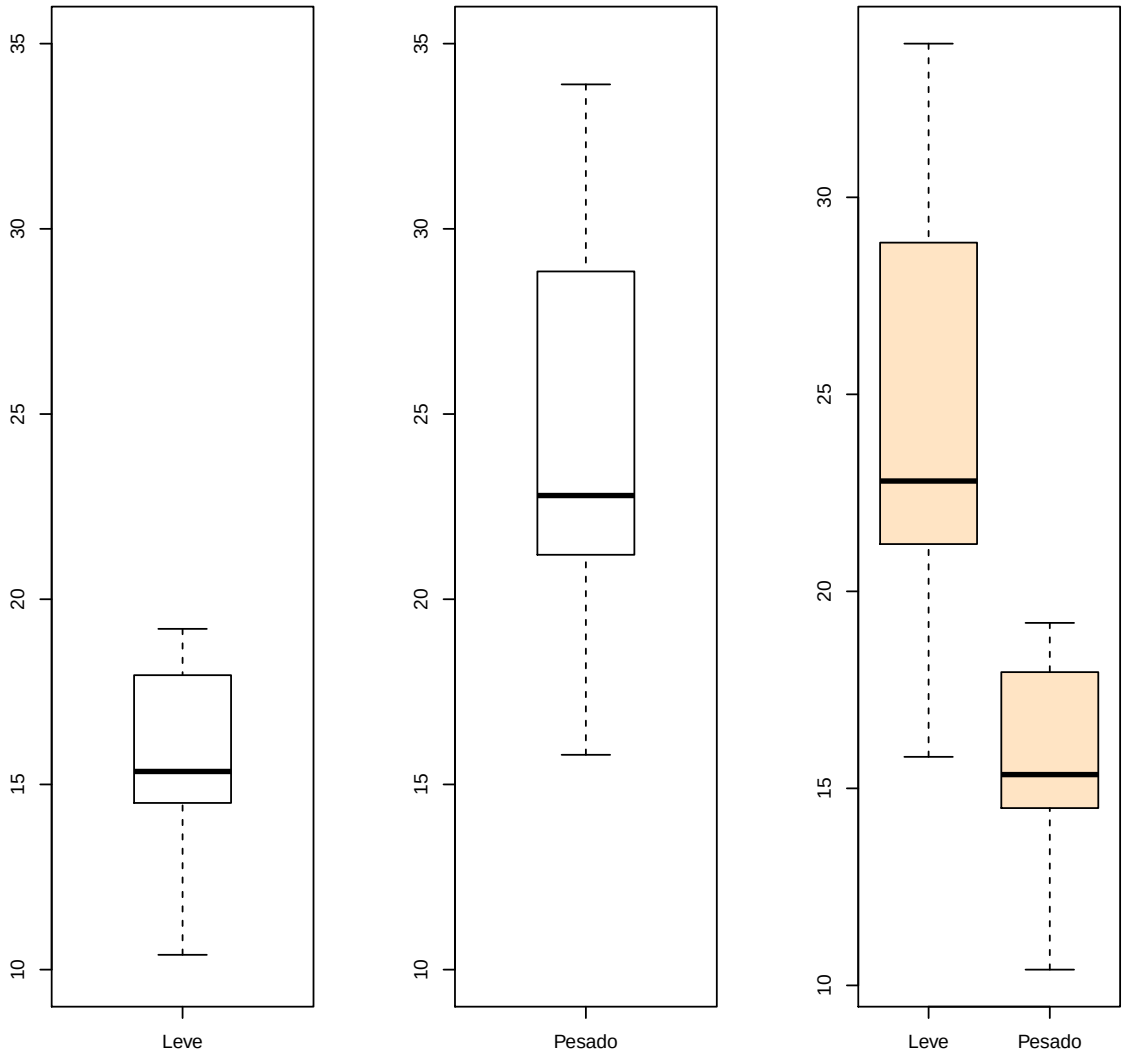
- Gráficos de Densidade (utilizados para variáveis contínuas):

```
# Histogramas de Distribuicoes
# (probabilidades + Kernel)
x <- rnorm(500,mean=3,sd=1)
y <- rchisq(500,df=5)
z <- rpois(500,lambda=3)
par(mfrow=c(1,3))
hist(x, main="Histograma da Normal", breaks=10, probability=T)
lines( density(x))
hist(y, main="Histograma de Qui-Quadrado", breaks=10, probability=T)
lines( density(y))
hist(z, main="Histograma de Poisson", breaks=10, probability=T)
lines( density(z))
```



- Box-Plot (utilizados para variáveis contínuas):

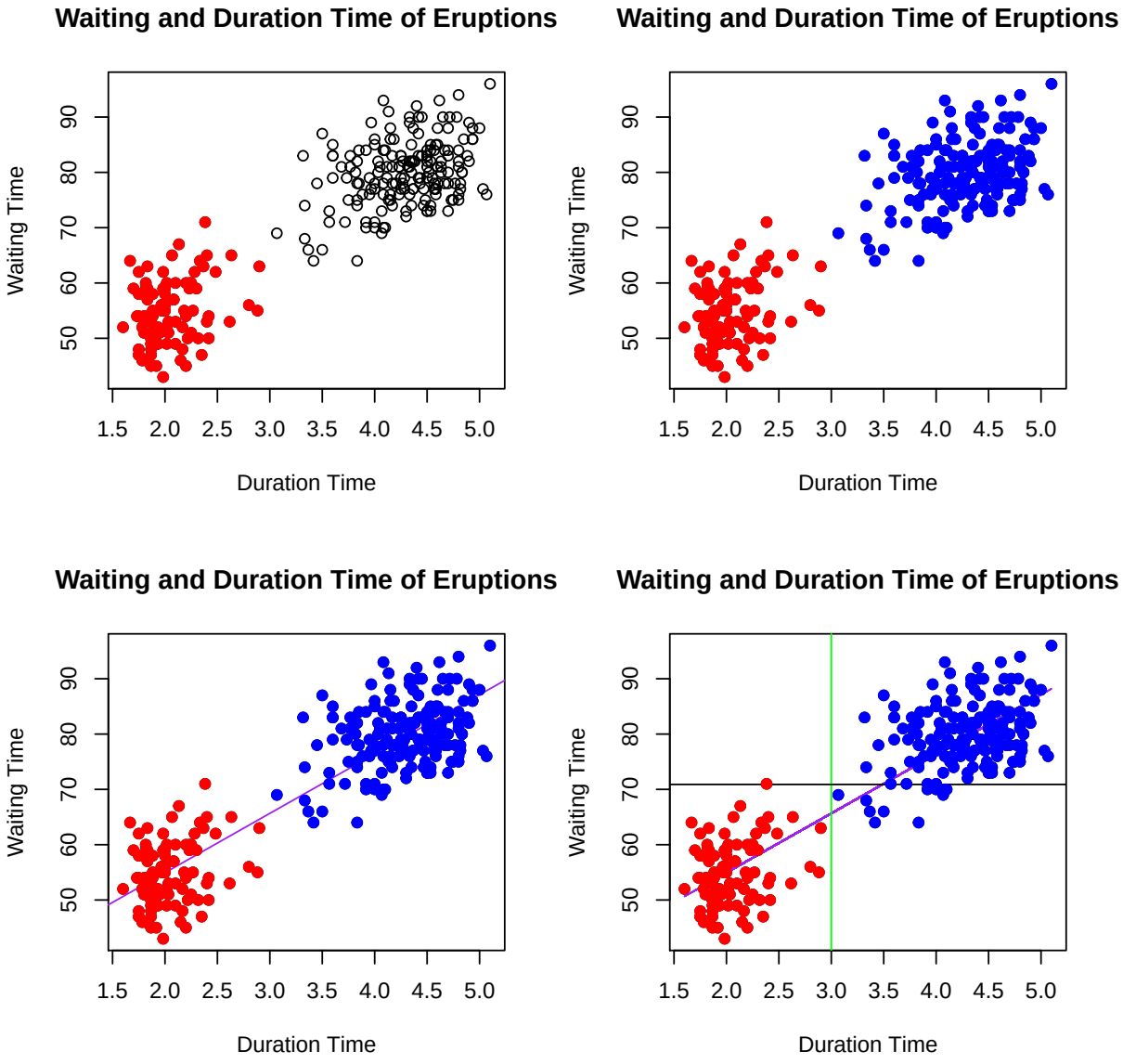
```
# Box plot de Desempenho de combustivel por Peso
par(mfrow=c(1,3))
with(mtcars, boxplot(mpg[wt>mean(wt)],ylim=c(10,35)))
axis(1,1,labels=c("Leve"))
with(mtcars, boxplot(mpg[wt<=mean(wt)],ylim=c(10,35)))
axis(1,1,labels=c("Pesado"))

# Box plot de Desempenho de combustivel por Peso
mtcars$new <- ifelse(mtcars$wt<mean(mtcars$wt),0,1)
mtcars$new <- factor(mtcars$new,
                     levels=c(0,1),
                     labels=c("Leve","Pesado"))
boxplot(mpg ~ new, data = mtcars, col = "bisque")
```



- Gráficos de Dispersão (utilizados para variáveis contínuas):

```
par(mfrow=c(2,2))
# Dispersao com titulo e cor dos pontos erupcoes curtas
short.eruptions<-with(faithful,faithful[eruptions<3,])
plot(faithful$eruptions,faithful$waiting,
main="Waiting and Duration Time of Eruptions",
ylab="Waiting Time",
xlab="Duration Time")
points(short.eruptions, col="red", pch=19)

# Dispersao com titulo e cor dos pontos
# por tempo de erupcoes
long.eruptions <- with(faithful,faithful[eruptions>=3,])
plot(faithful$eruptions,faithful$waiting,
main="Waiting and Duration Time of Eruptions",
```

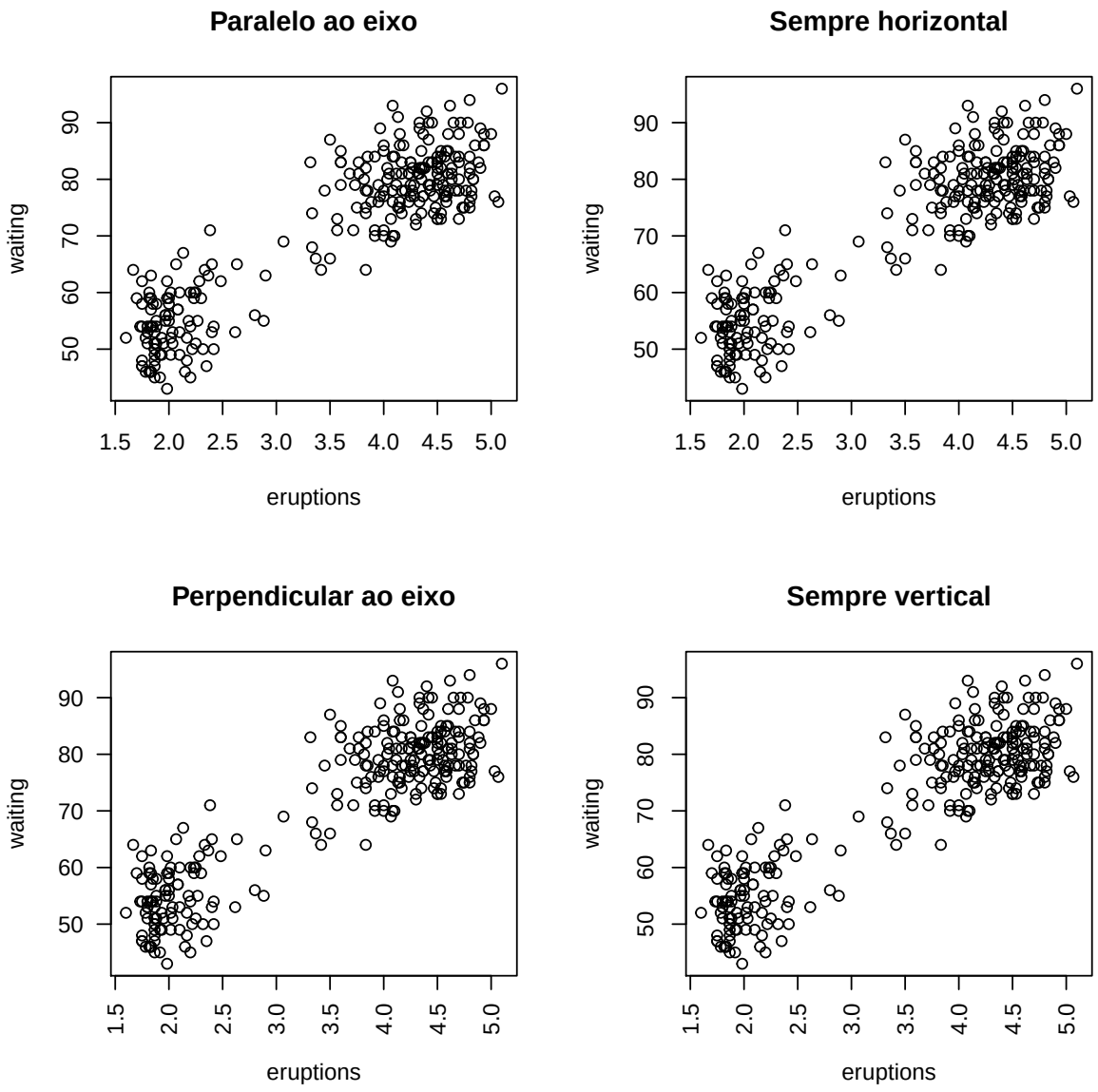
```
ylab="Waiting Time",
xlab="Duration Time")
points(short.eruptions, col="red", pch=19)
points(long.eruptions, col="blue", pch=19)

# Dispersao com titulo e cor dos pontos
# por tempo de erupcoes e regressao
fit<-lm(waiting~eruptions,data=faithful)
plot(faithful$eruptions,faithful$waiting,
main="Waiting and Duration Time of Eruptions",
ylab="Waiting Time",
xlab="Duration Time")
abline(fit, col="purple")
points(short.eruptions, col="red", pch=19)
points(long.eruptions, col="blue", pch=19)

# Dispersao com titulo e cor dos pontos
# por tempo de erupcoes e regressao
fit<-lm(waiting~eruptions,data=faithful) # regressao linear
plot(faithful$eruptions,faithful$waiting,
main="Waiting and Duration Time of Eruptions",
ylab="Waiting Time",
xlab="Duration Time")
lines(faithful$eruptions, fitted(fit), col="purple") # regressao
abline(v=3, col="green") # linha vertical
abline(h=mean(faithful$waiting)) # linha horizontal (media de y)
points(short.eruptions, col="red", pch=19)
points(long.eruptions, col="blue", pch=19)
```



6.2.3 Recursos Adicionais para Formatação de Gráficos

O comando abaixo permite que você controle a orientação dos valores que aparecem no eixo dos gráficos.

```
# Mudando a orientacao dos valores dos eixos
par(mfrow=c(2,2))
plot(faithful,las=0,main="Paralelo ao eixo")
plot(faithful,las=1,main="Sempre horizontal")
plot(faithful,las=2,main="Perpendicular ao eixo")
plot(faithful,las=3,main="Sempre vertical")
```



6.3 Medidas Descritivas

Nesta sessão apresentaremos as seguintes medidas descritivas:

- sumarização (média, amplitude, quartis e mediana)
- média
- desvio-padrão
- número de observações na amostra
- correlação pareada (Pearson e Spearman)
- intervalo de confiança para a média

Os comandos estão descritos com exemplos a seguir:

```r
# Algumas estatisticas descritivas:

desc <- function(x) {
  summary <- summary(x)
  media <- mean(x)
  sd <- sd(x)
  length <- length(x)
  IClow <- (mean(x)-qnorm(0.975)*sqrt(var(x)/length(x)))
  IChigh <- (mean(x)+qnorm(0.975)*sqrt(var(x)/length(x)))
  return(list(Sumario=summary,
              Teste=c(N=length,Media=media,sd=sd,
                      IC_low=IClow,IC_high=IChigh)))
}

with(auto,desc(price))

## $Sumario
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3291    4220    5006    6165    6332   15910
##
## $Teste
##        N    Media       sd   IC_low  IC_high
##   74.000 6165.257 2949.496 5493.240 6837.273

# Correlacao pareada
library(Hmisc)

myQs <- with(auto,cbind(price,mpg,headroom,trunk,weight,length))

# Matriz de correlacao de Pearson:
rcorr(myQs,type="pearson")

##          price   mpg headroom trunk weight length
## price     1.00 -0.47     0.11  0.31   0.54   0.43
## mpg      -0.47  1.00    -0.41 -0.58  -0.81  -0.80
## headroom  0.11 -0.41     1.00  0.66   0.48   0.52
## trunk     0.31 -0.58     0.66  1.00   0.67   0.73
## weight    0.54 -0.81     0.48  0.67   1.00   0.95
## length    0.43 -0.80     0.52  0.73   0.95   1.00
##
## n= 74
##
##
## P
##          price  mpg    headroom trunk  weight length
## price           0.0000 0.3313   0.0064 0.0000 0.0001
## mpg      0.0000        0.0002   0.0000 0.0000 0.0000
## headroom 0.3313 0.0002          0.0000 0.0000 0.0000
```

```
## trunk    0.0064 0.0000 0.0000          0.0000 0.0000
## weight   0.0000 0.0000 0.0000   0.0000        0.0000
## length   0.0001 0.0000 0.0000   0.0000 0.0000

# Matriz de correlacao de Spearman:
rcorr(myQs,type="spearman")

##          price   mpg headroom trunk weight length
## price     1.00 -0.54     0.10  0.40   0.49   0.49
## mpg      -0.54  1.00    -0.49 -0.65  -0.86  -0.83
## headroom  0.10 -0.49     1.00  0.68   0.53   0.53
## trunk     0.40 -0.65     0.68  1.00   0.66   0.72
## weight    0.49 -0.86     0.53  0.66   1.00   0.95
## length    0.49 -0.83     0.53  0.72   0.95   1.00
##
## n= 74
##
##
## P
##          price  mpg    headroom trunk  weight length
## price           0.0000 0.4113   0.0004 0.0000 0.0000
## mpg      0.0000        0.0000   0.0000 0.0000 0.0000
## headroom 0.4113 0.0000          0.0000 0.0000 0.0000
## trunk    0.0004 0.0000 0.0000          0.0000 0.0000
## weight   0.0000 0.0000 0.0000   0.0000        0.0000
## length   0.0000 0.0000 0.0000   0.0000 0.0000
```

7 Testes de Hipótese

7.1 Testes Não Paramétricos

Os testes não paramétricos são testes simples, requerem poucos pressupostos sobre a distribuição dos dados e são mais eficientes que os testes paramétricos quando as suposições dos testes paramétricos não são atendidas. Casos as sejam, os testes paramétricos são mais poderosos.

Em geral, os testes não paramétricos trabalham com estatísticas de ordem, e portanto ocorre perda de informações, uma vez que não consideram a magnitude dos dados. Além disso, a utilização das tabelas dos testes é mais complicada.

7.1.1 Teste de Shapiro-Wilk

O teste de Shapiro-Wilk é utilizado para verificar se os dados de uma amostra possui distribuição Normal. Este é um **teste para pequenas amostras** ($4 \leq X \leq 50$).

Seja $X_1, X_2, \ldots, X_n$ uma amostra aleatória de distribuição desconhecida, as hipóteses de interesse são:

$$\begin{cases} H_0 : X \sim N(\mu, \sigma^2) \\ H_A : X \neq N(\mu, \sigma^2) \end{cases} \tag{1}$$

Inicialmente, calcule:

$$D = \sum_{i=1}^{n}(X_i - \bar{X})^2$$

Posteriormente, calcule a estatística de teste T_3:

$$T_3 = \frac{1}{D}\left[\sum_{i=1}^{k} a_i(X^{(n-i+1)} - X^{(i)})\right]$$

onde a_i são valores tabelados gerados pelas médias, variâncias e covariâncias das estatísticas de ordem de uma amostra de tamanho n de uma distribuição Normal.

A estatística T_3 é basicamente o quadrado do coeficiente de correla cão de Pearson, calculado entre as estatísticas de ordem $X^{(i)}$ e os escores a_i, que representam o quanto as estatísticas de ordem deveriam ser se a população é normal. Se esta estatística é próxima de 1, podemos considerar que a amostra provém de uma distribuição normal. Ver Tabela A do apêndice para a obtenção dos valores padronizados de a_i.

Os quantis (W) de T_3 são fornecidos em uma tabela específica e devemos rejeitar H_0 se a estatística de teste for menor que o valor fornecido pela tabela (ver Tabela B do apêndice):

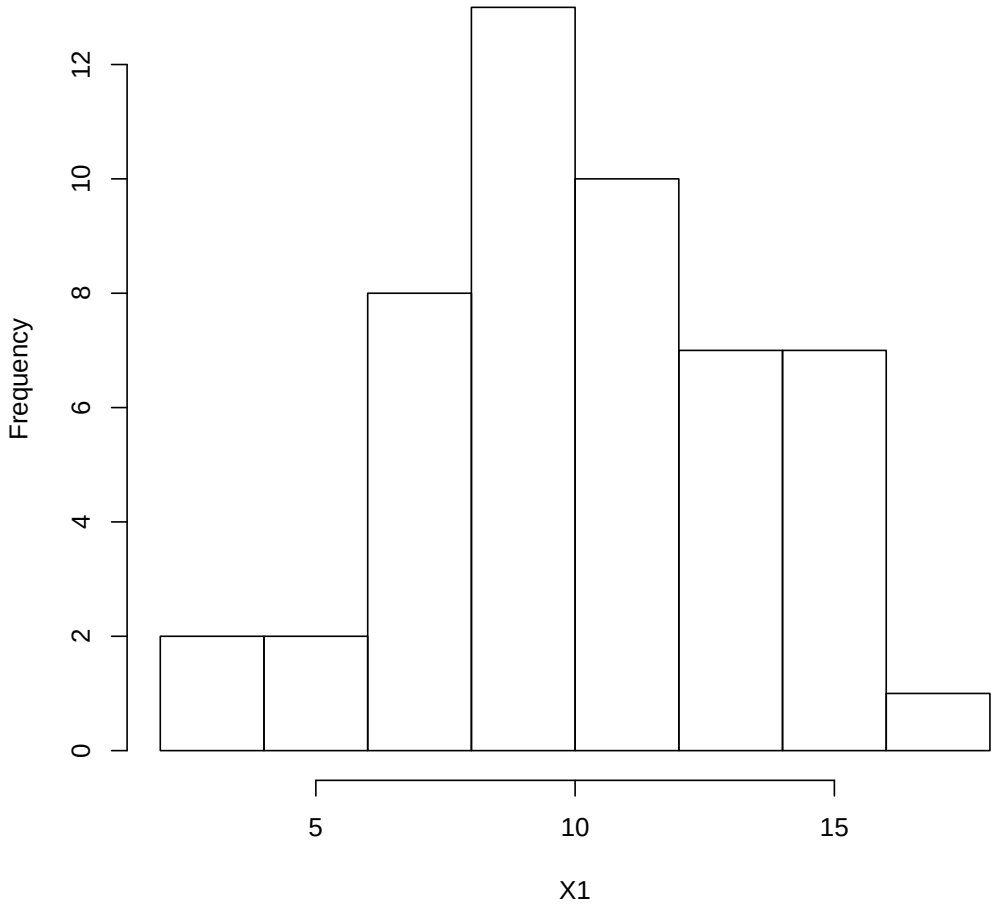
$$\begin{cases} H_0 : T_3 \geq W_p \\ H_A : T_3 < W_p \end{cases} \tag{2}$$

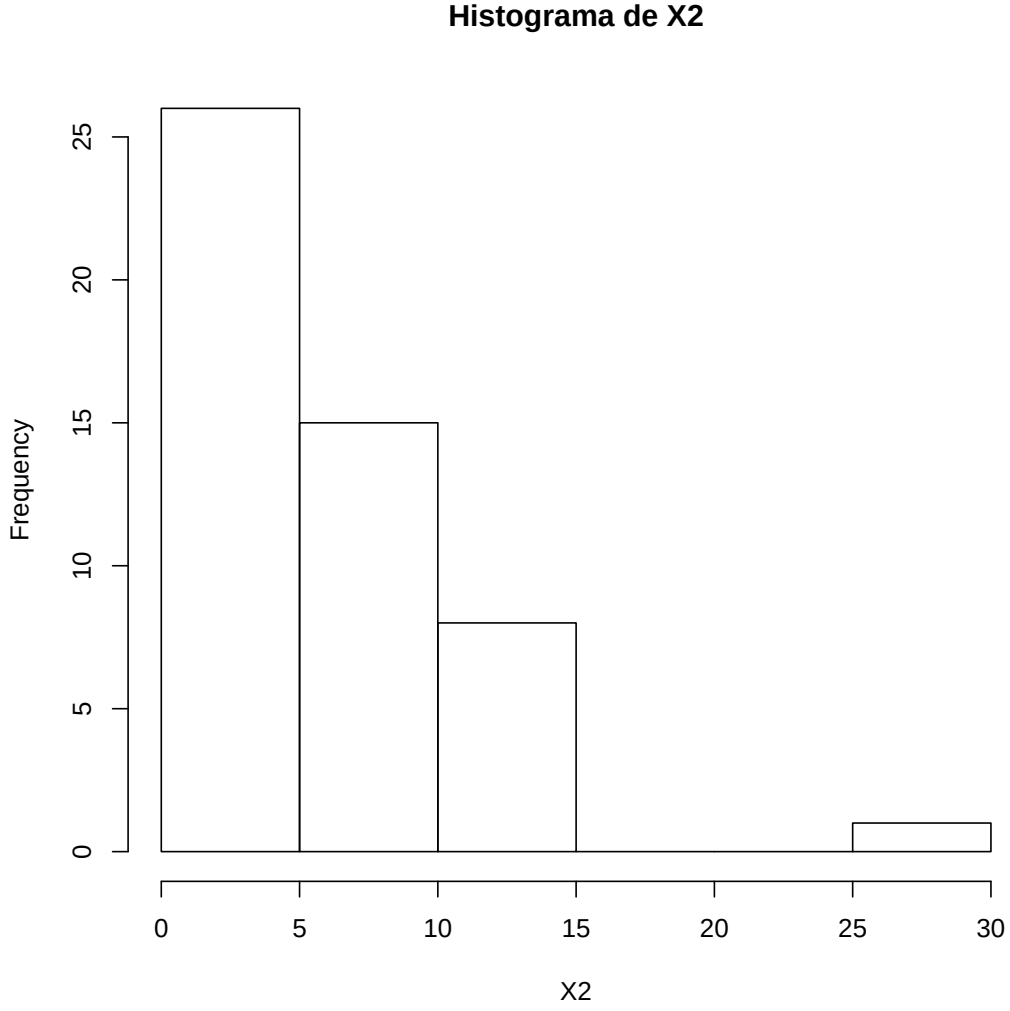
```r
rm(list=ls(all=TRUE))

set.seed(2)

# Gera uma amostra de 50 observacoes da distribuicao normal
X1 <- rnorm(50, mean=10, sd=3)

hist(X1,main="Histograma de X1")  # Plota o histograma de X1

## Exemplo teste de Shapiro-Wilk
shapiro.test(X1) # Nao rejeita a hipotese de normalidade
```

```
##
##  Shapiro-Wilk normality test
##
## data:  X1
## W = 0.9763, p-value = 0.4073
```

```r
X2 <- rchisq(50, df=5)

hist(X2,main="Histograma de X2") # Plota o histograma de X2

shapiro.test(X2) # Rejeita a hipotese de normalidade
```

```
##
##  Shapiro-Wilk normality test
##
## data:  X2
## W = 0.8172, p-value = 2.234e-06
```


Histograma de X1
Frequency
0
2
4
6
8
10
12
5
10
15
X1



7.1.2 Teste de Anderson-Darling

Assim como o teste de Shapiro-Wilk, o teste de Anderson-Darling é utilizado para verificar se os dados possuem distribuição Normal.

A estatística de teste é dada por:

$$Z = \left(1 + \frac{0.75}{n} + \frac{2.25}{n^2}\right)\left(- n - \frac{1}{n}\sum_{i=1}^{n}[2i-1][ln(p_{(i)}) + ln(1 - p_{n-i+1})]\right)$$

Em que $p_{(i)} = \Phi([y_i - \bar{y}]/s)$ são os percentis da distribuição normal padrão. Deve-se rejeitar a hipótese de normalidade se a estatística Z for maior que o valor crítico fornecido pela tabela:

$$\begin{cases} H_0 : \text{Os dados possuem distribuição normal} \\ H_A : \text{Os dados não possuem distribuição normal} \end{cases} \tag{3}$$

O Teste de Anderson-Darling é mais poderoso para amostras maiores que 50, enquanto o de Shapiro-Wilk é mais poderoso para amostras menores ou iguais a 50.

```r
install.packages("nortest")

##
## The downloaded binary packages are in
##  /var/folders/xl/tmfgg9y1719_9z70r1dydlm00000gn/T//RtmptGBvbq/downloaded_packages

library(nortest)      # Carrega o pacote necessario

# Fixando gerador de numeros aleatorios
set.seed(2)

# Gera uma amostra de 50 observacoes da distribuicao normal
X3 <- rnorm(500, mean=10, sd=3)
# Gera uma amostra de 50 observacoes da distribuicao qui-quadrado
X4 <- rchisq(500, df=5)

shapiro.test(X3)      # Nao rejeita a hipotese de normalidade

##
##  Shapiro-Wilk normality test
##
## data:  X3
## W = 0.9973, p-value = 0.5826

ad.test(X3)           # Nao rejeita a hipotese de normalidade

##
##  Anderson-Darling normality test
##
## data:  X3
## A = 0.2263, p-value = 0.8168

shapiro.test(X4)      # Rejeita a hipotese de normalidade

##
##  Shapiro-Wilk normality test
##
## data:  X4
## W = 0.9336, p-value = 4.081e-14

ad.test(X4)           # Rejeita a hipotese de normalidade

##
##  Anderson-Darling normality test
##
## data:  X4
## A = 8.0145, p-value < 2.2e-16
```

7.1.3 Teste de Kolmogorov-Smirnov

Avalia se os dados amostrais se aproximam de uma determinada distribuição.

Considere uma amostra aleatória simples de uma população com função de distribuição acumulada contínua F_X desconhecida. Queremos testar as seguintes hipóteses:

$$\begin{cases} H_0 : F_x(x) = F(x) \\ H_A : F_x(x) \neq F(x) \end{cases} \tag{4}$$

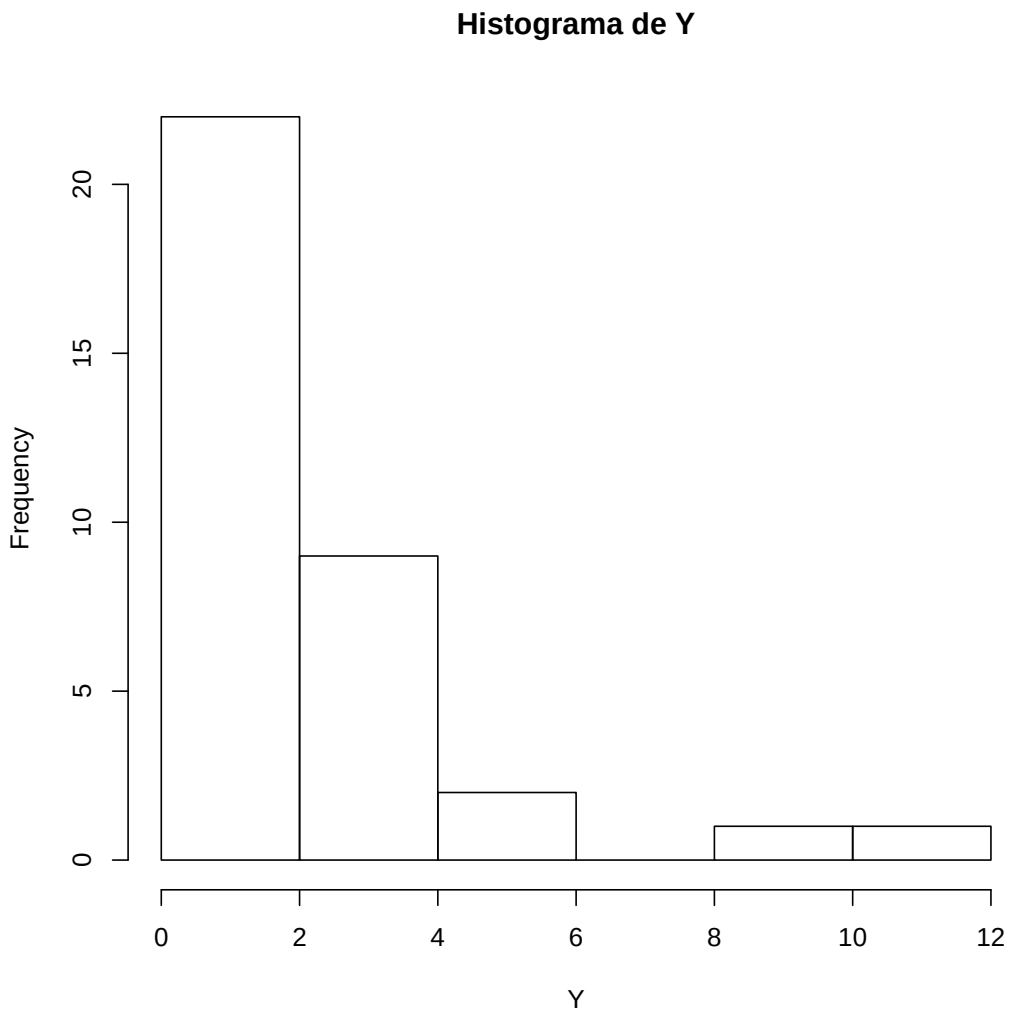
A estatística de teste é dada por:

$$D = \max\left(|F_x(X) - F(X)|\right)$$

Onde $F_x(X)$ é a função de distribuição acumulada empírica dos dados e $F(X)$ é a função distribuição acumulada assumida para os dados. Esta função corresponde à distância máxima vertical entre os gráficos de $F(X)$ e $F_x(X)$. Se a estatística de teste D for maior que o valor tabelado, devemos rejeitar a hipótese nula (ver Tabela C do Apêndice).

```
# Fixando gerador de numeros aleatorios
set.seed(4)

# Gera 35 obs de uma dist exponencial com taxa 0.5
Y <- rexp(35,0.5)

hist(Y,main="Histograma de Y")

# Exemplo teste de Kolmogorov-Smirnov #

 # Testa se dados vem de dist exponencial com taxa 0.5
ks.test(Y,"pexp",0.5)

##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Y
## D = 0.065, p-value = 0.9962
## alternative hypothesis: two-sided

# Testa se dados vem de dist exponencial com taxa empirica
# (quando nao se sabe o valor verdadeiro)
ks.test(Y,"pexp",1/mean(Y))

##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Y
## D = 0.0743, p-value = 0.9826
## alternative hypothesis: two-sided
```

```r
# Testa se dados vem de uma dist normal com media 10 e var 3
ks.test(X3,"pnorm",10,3)

##
##  One-sample Kolmogorov-Smirnov test
##
## data:  X3
## D = 0.0489, p-value = 0.1837
## alternative hypothesis: two-sided

# Testa se dados vem de uma dist exponencial com media 15 e var 3
ks.test(X3,"pnorm",15,3)

##
##  One-sample Kolmogorov-Smirnov test
##
## data:  X3
## D = 0.5611, p-value < 2.2e-16
## alternative hypothesis: two-sided
```



7.1.4 Teste de Wilcoxon Pareado

O teste de Wilcoxon Pareado é utilizado para comparar a distribuição de duas populações dependentes.

Considere duas amostras dependentes de duas populações X e Y, ou seja, temos uma amostra de pares:

$$[((X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n))]$$

Defina $D_i = X_i - Y_i$, para $i = 1, 2, \ldots, n$. Posteriormente, ordene os valores de $|D_i|$ e atribua postos a esses valores, em ordem crescente. A cada posto, atribua o sinal da diferença D_i. Seja T^+ a soma dos postos dos D_i com sinais positivos e T^- a soma dos postos dos D_i com sinais negativos. A estatística de teste é dada por:

$$T_0 = \max(T^+, T^-)$$

Para amostras pequenas ($n \leq 15$), o valor-p do teste pode ser obtido através da **tabela Siegel** (ver Tabela D do Apêndice). Para amostras grandes ($n \leq 15$), o valor-p é obtido através da aproximação pela distribuição Normal, com média $\mu = 0$ e desvio-padrão:

$$\sigma_T = \sqrt{\frac{n(n+1)(2n+1)}{6}}$$

```r
rm(list=ls(all=TRUE))

set.seed(2)

X1 <- rexp(50,0.3)  # Gera amostra de dist. exponencial com taxa 0.3
X2 <- rexp(50,0.5)  # Gera amostra de dist. exponencial com taxa 0.5

# Aplicando o teste:
wilcox.test(X1,X2,alternative="two.sided",mu=0,paired=TRUE)

##
##  Wilcoxon signed rank test with continuity correction
##
## data:  X1 and X2
## V = 951, p-value = 0.002515
## alternative hypothesis: true location shift is not equal to 0

# Testa se a mediana das duas distribuicoes e igual
```

7.1.5 Teste de Wilcoxon / Mann-Whitney

O teste de Wilcoxon / Mann-Whitney é utilizado para comparar duas amostras independentes. Para a realização deste teste, é preciso que as amostras estejam em escala pelo menos ordinal. Queremos testar se as duas populações possuem a mesma mediana, ou se uma população tende a ter valores maiores que a outra.

Considere duas amostras aleatórias das populações X_1 e X_2, de tamanhos n_1 e n_2, respectivamente. Combine as observações dos dois grupos e atribua postos a esses valores.

Sejam S_1 e S_2 as somas dos postos relacionados aos elementos das populações 1 e 2. Calcule U_1 e U_2:

$$U_1 = S_1 - \frac{1}{2}n_1(n_1+1)$$
$$U_2 = S_2 - \frac{1}{2}n_2(n_2+1)$$

A estatística de teste U é dada pelo mínimo entre U_1 e U_2. Para amostras pequenas (entre 5 e 40 observações) usar os valores críticos tabelados de U (ver Tabela E do Apêndice). Para amostras grandes, a distribuição de U se aproxima pela distribuição normal com média e desvio-padrão dados por:

$$\mu_U = \frac{n_1 n_2}{2}$$
$$\sigma_U = \sqrt{\frac{n_1 n_2(n_1+n_2+1)}{12}}$$

Para os valores críticos de z, ver Tabela F do Apêndice.

```
rm(list=ls(all=TRUE))

set.seed(3)

X1 <- rexp(20,1)  # Gera amostra de dist. exponencial com taxa 1
X2 <- rexp(30,1.5)  # Gera amostra de dist. exponencial com taxa 1.5

# Aplicando o teste
wilcox.test(X1,X2,alternative="two.sided",mu=0,paired=FALSE)

##
##  Wilcoxon rank sum test
##
## data:  X1 and X2
## W = 290, p-value = 0.8522
## alternative hypothesis: true location shift is not equal to 0

# Testa se a mediana das duas distribuicoes e igual
```

7.1.6 Teste de Kruskall-Wallis

Este teste é utilizado para comparar as medianas de mais de dois grupos e é um teste alternativo à Anova (*Analysis of Variance*). È uma extensão do teste de Wilcoxon / Mann-Whitney. O objetivo deste teste é verificar se existe pelo menos uma distribuição diferente entre as k populações.

Combine todas as amostras, ordene e atribua postos. Calcule a soma dos postos S_i e os postos médios $\bar{R_{(i)}}$ de cada população $i = 1, 2, \ldots, k$.

A estatística de teste é dada por:

$$H = \frac{12}{N(N-1)} \sum_{j=1}^{k} \frac{\bar{R_i}^2}{n_j} - 3(N+1)$$

Para amostras pequenas ($k = 3, n_i \leq 5$), utilizar tabela Siegel. Para amostras grandes, H segue uma distribuição aproximadamente χ^2 com $k-1$ graus de liberdade (ver Tabela G do Apêndice).

```
rm(list=ls(all=TRUE))

set.seed(5)

# Simulando dados vetoriais
X1 <- rexp(10,1)  # Gera amostra de dist. exponencial com taxa 1
X2 <- rexp(8,2)  # Gera amostra de dist. exponencial com taxa 1.5
X3 <- rexp(12,0.8) # Gera amostra de dist. exponencial com taxa 0.8
X4 <- rexp(8,3) # Gera amostra de dist. exponencial com taxa 3

X <- cbind(c(X1,X2,X3,X4),
           c(rep(1,10),rep(2,8),rep(3,12),rep(4,8)))
head(X)

##             [,1] [,2]
## [1,] 1.98800998    1
## [2,] 0.37043719    1
## [3,] 0.07253794    1
## [4,] 0.40211492    1
## [5,] 0.05591997    1
## [6,] 0.61587040    1

tail(X)

##              [,1] [,2]
## [33,] 0.49250053    4
## [34,] 0.08264433    4
## [35,] 0.59208554    4
## [36,] 0.22796127    4
## [37,] 0.03635407    4
## [38,] 0.19106146    4

kruskal.test(X[,1],X[,2])

##
##  Kruskal-Wallis rank sum test
##
## data:  X[, 1] and X[, 2]
## Kruskal-Wallis chi-squared = 8.1746, df = 3, p-value = 0.04254

# Usando o teste a partir de data frame
X <- data.frame(x=c(X1,X2,X3,X4),k=c(c(rep(1,10),
                                       rep(2,8),rep(3,12),rep(4,8))))
kruskal.test(X$x,X$k)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  X$x and X$k
## Kruskal-Wallis chi-squared = 8.1746, df = 3, p-value = 0.04254
```

7.1.7 Teste de Friedman

O teste de Friedman é utilizado quando se deseja comparar diversos tratamentos em amostras dependentes. Para a aplicação deste teste, é preciso que os dados possuam nível de mensuração pelo menos ordinal. As hipóteses de interesse são:

$$\begin{cases} H_0 : \text{As amostras vem da mesma população} \\ H_A : \text{Pelo menos uma das } k \text{ distribuições é diferente} \end{cases} \tag{5}$$

Suponha que tenhamos medidas de b unidades amostrais em k tratamentos diferentes. Primeiramente, deve-se ordenar as k observações da menor para a maior de forma separada em cada um dos b blocos e atribuir postos $(1, 2, \ldots, k)$ para cada bloco da tabela. Assim, a posição esperada de cada observação sob H_0 é $\frac{(k+1)}{2}$.

Seja r_{ij} o posto do indivíduo i na j-ésima medida, definimos a soma de todos os ranks da coluna j (ou seja, de cada tratamento), por:

$$R_j = \sum_{i=1}^{b} r_{ij}, j = 1, \ldots, k$$

A estatística de teste de Friedman é dada por:

$$S = \left[\frac{12}{bk(k+1)} \sum_{j=1}^{k} {R_j}^2 \right] - 3b(k+1)$$

Sob H_0, a estatística S tem distribuição aproximadamente χ^2 com $k-1$ graus de liberdade (ver Tabela G do Apêndice). Portanto, rejeitamos H_0 ao nível α de significância se:

$$S > \chi^2_{k-1,\alpha}$$

```r
rm(list=ls(all=TRUE))

set.seed(6)

# Simulando dados vetoriais
X1 <- rexp(12,0.4)  # Gera amostra de dist. exponencial com taxa 0.4
X2 <- rexp(12,0.7)  # Gera amostra de dist. exponencial com taxa 0.7
X3 <- rexp(12,0.8) # Gera amostra de dist. exponencial com taxa 0.8
X4 <- rexp(12,0.2) # Gera amostra de dist. exponencial com taxa 0.2

dados <- data.frame(X=c(X1,X2,X3,X4),
```

```
                    grupo=c(rep(1,12),rep(2,12),rep(3,12),rep(4,12)),
                    bloco=rep(1:12,4))

head(dados)

##           X grupo bloco
## 1 0.5313415     1     1
## 2 0.4580872     1     2
## 3 1.5374169     1     3
## 4 0.8831536     1     4
## 5 5.2776757     1     5
## 6 0.7178638     1     6

tail(dados)

##           X grupo bloco
## 43 2.844721     4     7
## 44 5.198732     4     8
## 45 4.964011     4     9
## 46 2.390678     4    10
## 47 7.348806     4    11
## 48 4.966519     4    12

# Cria um banco de dados no formato longo com os dados gerados

# Aplica o teste de Friedman:
friedman.test(dados$X,dados$grupo,dados$bloco)

##
##  Friedman rank sum test
##
## data:  dados$X, dados$grupo and dados$bloco
## Friedman chi-squared = 14, df = 3, p-value = 0.002905
```

7.2 Testes Paramétricos

7.2.1 Teste para média de uma população com variância conhecida

Suponha que tenhamos uma amostra $X_1, X_2, \ldots, X_n$ de distribuição Normal, com média μ desconhecida e variância σ^2 conhecida e desejamos testar as seguintes hipóteses:

$$\begin{cases} H_0 : \mu = \mu_0 \\ H_A : \mu \neq \mu_0 \end{cases} \tag{6}$$

A estatística de teste é dada por:

$$z_0 = \frac{(\bar{x} - \mu_0)}{\sqrt{\frac{\sigma^2}{n}}}$$

A região crítica para um teste bilateral com um nível α de significância, a região crítica assume a seguinte forma:

$$RC = \{z \in \mathbb{R} : z > z_{(\alpha/2)} | z < -z_{(\alpha/2)}\}$$

onde $z_{(\frac{\alpha}{2})}$ é encontrado na tabela da distribuição Normal padrão e é tal que:

$$P(Z > z_{(\frac{\alpha}{2})}) = \frac{\alpha}{2}$$

O p-valor do teste é dado por:

$$\text{p-valor} = 2P(Z > |z_0| \, | H_0)$$

Caso a hipótese de interesse seja unilateral, ou seja, quando temos as seguintes hipóteses:

$$\begin{cases} H_0 : \mu \leq \mu_0 \\ H_A : \mu > \mu_0 \end{cases} \tag{7}$$

ou

$$\begin{cases} H_0 : \mu \geq \mu_0 \\ H_A : \mu < \mu_0 \end{cases} \tag{8}$$

A região crítica será, respectivamente, dada por:

$$\text{p-valor} = P(Z > z_0 | H_0)$$

ou

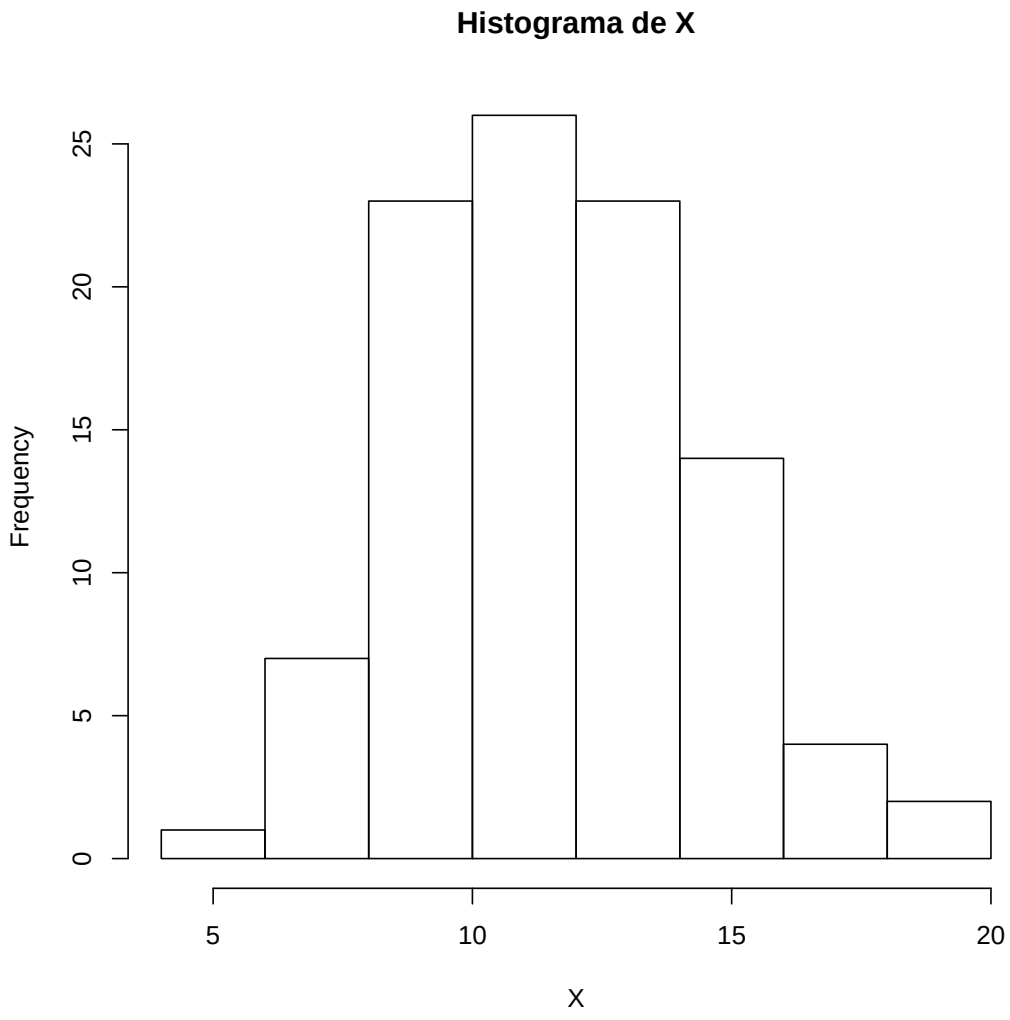
$$\text{p-valor} = P(Z < -z_0 | H_0)$$

e o p-valor será dado por:

$$\text{p-valor} = P(Z > |z_0| \, | H_0)$$

```
# Teste para a m?dia de uma popula??o com vari?ncia conhecida

set.seed(11)

# Gera uma amostra de 25 observacoes da distribuicao normal
#com media 12 e desvio padrao 3
X <- rnorm(100,mean=12,sd=3)

# Histograma da amostra gerada acima:

hist(X,main="Histograma de X")

# Nao existe uma funcao que faca o teste
# para a media com variancia conhecida,
# porque na pratica esse teste nao e comum.
# Portanto, iremos fazer o teste na mao:

# Funcao z.test: implementacao do teste Z
```

```r
z.test <- function(x,mu=0,var=1){
  z0 <- (mean(x)-mu)/sqrt(var/length(x)) # Calcula estatistica de teste
  valorP <- 2*(1-pnorm(abs(z0))) # Calcula o valor-p
  IClow <- (mean(x)-1.95*sqrt(var/length(x))) # Calcula IC Inf de 95%
  IChigh <- (mean(x)+1.95*sqrt(var/length(x))) # Calcula IC Sup de 95%
  return(cbind(z0=z0,valor_p=valorP,IC_Low=IClow,IC_High=IChigh))
}

# Aplicando o teste Z criado acima
z.test(X,12,9)

##             z0   valor_p   IC_Low  IC_High
## [1,] -1.235137 0.2167794 11.04446 12.21446
```



7.2.2 Teste para a média de uma população com variância desconhecida

Suponhamos agora que tenhamos uma amostra $X_1, X_2, \ldots, X_n$ de distribuição Normal, com média μ desconhecida e variância σ^2 desconhecida e desejamos testar as seguintes

hipóteses:

$$\begin{cases} H_0 : \mu = \mu_0 \\ H_A : \mu \neq \mu_0 \end{cases} \tag{9}$$

Neste caso, a estatística de teste é dada por:

$$z_0 = \frac{(\bar{x} - \mu_0)}{\sqrt{\frac{s^2}{n}}}$$

onde

$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}$$

é a variância amostral. A região crítica para um teste bilateral com um nível α de significância, assume a seguinte forma:

$$RC = \{t \in \mathbb{R} : t > t_{(\alpha/2)} | t < -t_{(\alpha/2)}\}$$

onde $t_{(\frac{\alpha}{2})}$ é encontrado na tabela da distribui c ão T de Student com $n-1$ graus de liberdade e é tal que:

$$P(T > t_{(\frac{\alpha}{2})}) = \frac{\alpha}{2}$$

O p-valor do teste é dado por:

$$\text{p-valor} = 2P(T > |t_0| \, | H_0)$$

Caso a hipótese de interesse seja unilateral, a região crítica e o valor-p são obtidos de maneira análoga ao caso de variância conhecida, no entanto, utilizando como referência a distribuição t com $n-1$ graus de liberdade.

```
# Aplicando o teste para o vetor X:

t.test(X,mu=12)  # Testa se a media e igual a 12

##
##  One Sample t-test
##
## data:  X
## t = -1.3507, df = 99, p-value = 0.1799
## alternative hypothesis: true mean is not equal to 12
## 95 percent confidence interval:
##  11.08511 12.17381
## sample estimates:
## mean of x
##  11.62946

t.test(X,mu=11)  # Testa se a media e igual a 11

##
##  One Sample t-test
##
```

```
## data:  X
## t = 2.2944, df = 99, p-value = 0.02388
## alternative hypothesis: true mean is not equal to 11
## 95 percent confidence interval:
##  11.08511 12.17381
## sample estimates:
## mean of x
##  11.62946
```

7.2.3 Teste para comparação de duas amostras

Nosso objetivo é comparar duas populações cujas médias populacionais são desconhecidas.

Amostras Dependentes

Considere duas amostras dependentes de duas populações X e Y, ou seja, temos uma amostra de pares:

$$[((X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n))]$$

com médias μ_X, μ_Y desconhecidas. Suponha ainda que queremos verificar se existe diferença entre as médias das populações X e Y, ou seja:

$$\begin{cases} H_0 : \mu_X = \mu_Y \\ H_A : \mu_X \neq \mu_Y \end{cases} \tag{10}$$

Considere a diferença:

$$D = X - Y$$

Consideremos a amostra formada pelas diferenças, ou seja, consideremos:

$$D_1 = X_1 - Y_1, \ldots, D_n = X_n - Y_n$$

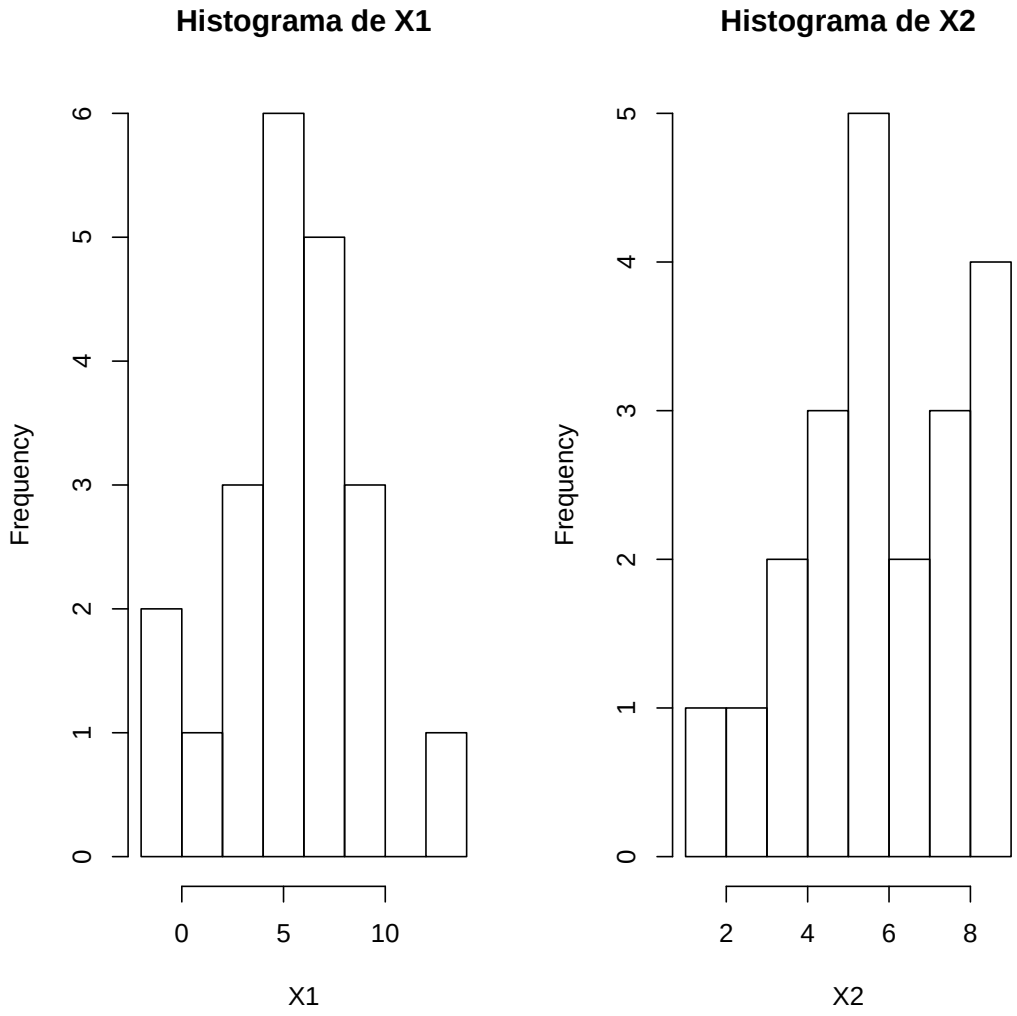
Se $D_1, \ldots, D_n$ é uma amostra aleatória simples e normalmente distribuída com média μ_D. Neste caso, as hipóteses ($H_A : \mu_X = \mu_Y$ e $H_A : \mu_X \neq \mu_Y$) são equivalentes a:

$$\begin{cases} H_0 : \mu_D = 0 \\ H_A : \mu_D \neq 0 \end{cases} \tag{11}$$

Para testar as hipóteses acima, basta utilizar o teste para a média de uma população com variância desconhecida citado anteriormente.

```
## Teste para compara??o de duas amostras

# Teste t-pareado:

 # Gerando amostra de 21 obs da dist normal(5,3)
X1 <- rnorm(21,mean=5,sd=3)
 # Gerando amostra de 21 obs da dist normal(6,2)
```

```r
X2 <- rnorm(21,mean=6,sd=2)

# Histogramas:
par(mfrow=c(1,2))
hist(X1,main="Histograma de X1")
hist(X2,main="Histograma de X2")

# Aplicando o teste:
t.test(X1,X2,mu=0,paired=TRUE)

##
##  Paired t-test
##
## data:  X1 and X2
## t = -0.4795, df = 20, p-value = 0.6368
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.194551  1.374251
## sample estimates:
## mean of the differences
##              -0.4101499
```



Amostras Independentes

Sejam $X_1, X_2, \ldots, X_n$ uma amostra aleatória simples selecionada na população X e seja $Y_1, Y_2, \ldots, Y_n$ uma amostra aleatória simples selecionada na população Y. Admita que a população X seja independente da população Y. Assuma também que as duas populações sejam normalmente distribuídas com variâncias populacionais σ_X^2 e σ_Y^2, ambas desconhecidas.

O teste terá a seguinte estrutura:

$$\begin{cases} H_0 : \mu_X = \mu_Y \\ H_A : \mu_X \neq \mu_Y \end{cases} \tag{12}$$

Caso 1: Variâncias Iguais $(\sigma X^2 = \sigma Y^2)$

Nesse caso, a estatística de teste para as hipóteses ($H_A : \mu_X = \mu_Y$ e $H_A : \mu_X \neq \mu_Y$) é dada por:

$$t_0 = \frac{(\bar{X} - \bar{Y})}{\sqrt{s_P^2(\frac{1}{n_X} + \frac{1}{n_Y})}}$$

onde

$$s_p^2 = \frac{(n_X - 1)s_X^2 + (n_Y - 1)s_Y^2}{(n_X + n_Y - 2)}$$

é a variância amostral combinada das populações X e Y.

Sob H_0, a estatística t_0 tem distribuição t de Student com $(n_1 + n_2 - 2)$ graus de liberdade.

A região crítica para um teste bilateral com um nível α de significância, assume a seguinte forma:

$$RC = \{t \in \mathbb{R} : t > t_{(\alpha/2)} | t < -t_{(\alpha/2)}\}$$

nde $t_{(\frac{\alpha}{2})}$ é encontrado na tabela da distribuição t de Student com $(n_1 + n_2 - 2)$ graus de liberdade e é tal que:

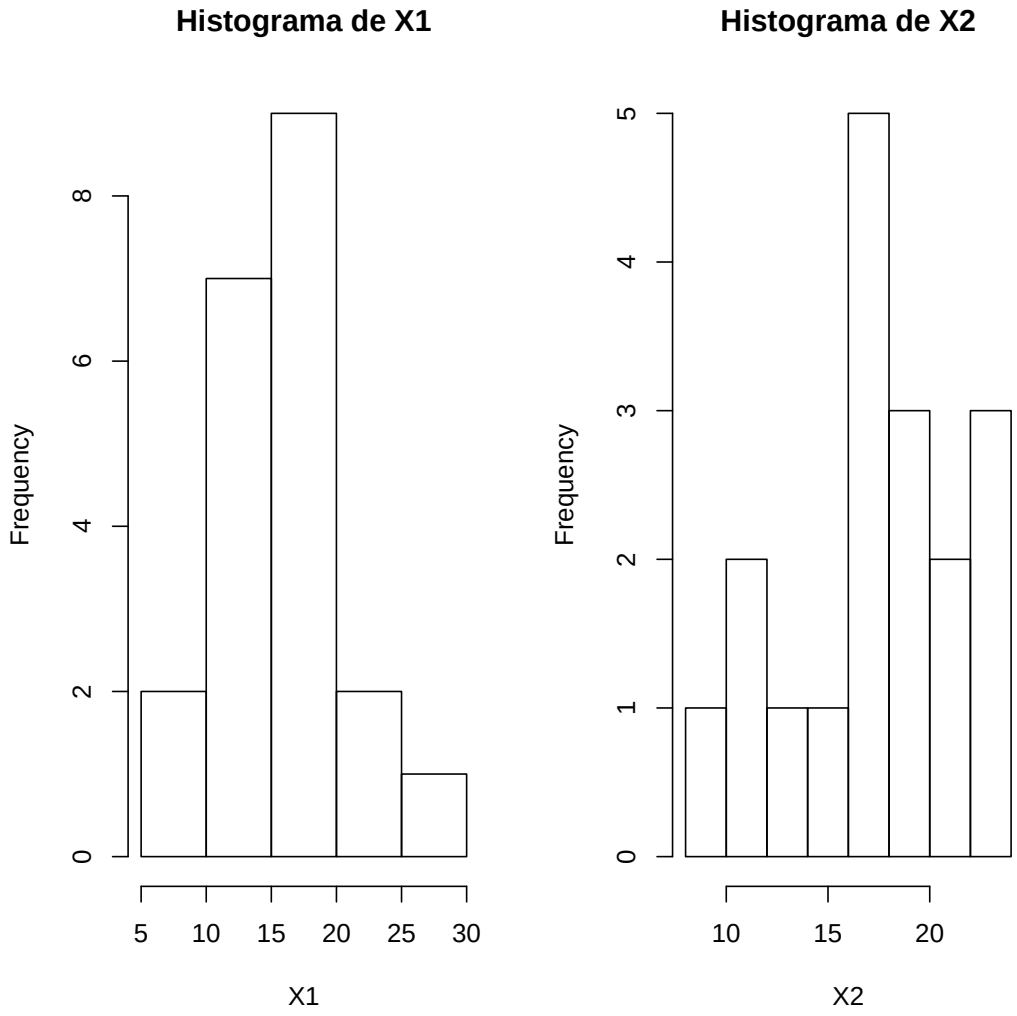
$$P(T > t_{(\frac{\alpha}{2})}) = \frac{\alpha}{2}$$

O p-valor do teste é dado por:

$$\text{p-valor} = 2P(T > |t_0| \, | H_0)$$

Para os testes unilaterais, a região crítica e o valor-p são obtidos de maneira análoga ao caso do teste de variância conhecida; no entanto, utiliza-se como referência a distribuição t com $(n_1 + n_2 - 2)$ graus de liberdade.

```
# Amostras Independentes com variancias iguais

# Gerando amostra de 21 obs da dist normal(15,5)
X1 <- rnorm(21,mean=15,sd=5)
# Gerando amostra de 18 obs da dist normal(17,5)
X2 <- rnorm(18,mean=17,sd=5)

# Histogramas:
par(mfrow=c(1,2))
hist(X1,main="Histograma de X1")
hist(X2,main="Histograma de X2")

# Aplicando o teste:
t.test(X1,X2,mu=0,paired=FALSE,var.equal=TRUE)

##
##  Two Sample t-test
##
## data:  X1 and X2
## t = -1.0994, df = 37, p-value = 0.2787
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.554953  1.350671
## sample estimates:
## mean of x mean of y
##  15.73298  17.33512
```



Caso 2: Variâncias Diferentes ($\sigma X^2 \neq \sigma Y^2$)

Nesse caso, a estatística de teste para as hipóteses:

$$\begin{cases} H_0 : \mu_X = \mu_Y \\ H_0 : \mu_X \neq \mu_Y \end{cases} \tag{13}$$

é dada por:

$$t_0 = \frac{(\bar{X} - \bar{Y})}{\sqrt{\left(\frac{s_X^2}{n_X} + \frac{S_Y^2}{n_Y}\right)}}$$

Sob H_0, a estatística t_0 tem distribuição t de Student com ν graus de liberdade:

$$\nu = \frac{\left(\frac{s_X^2}{n_X} + \frac{s_Y^2}{n_Y}\right)^2}{\frac{\left(\frac{s_X^2}{n_X}\right)^2}{n_X - 1} + \frac{\left(\frac{s_Y^2}{n_Y}\right)^2}{n_Y - 1}}$$

A região crítica para um teste bilateral com um nível α de significância, assume a seguinte forma:

$$RC = \{t \in \mathbb{R} : t > t_{(\alpha/2)} | t < -t_{(\alpha/2)}\}$$

onde $t_{\frac{\alpha}{2}}$ é encontrado na tabela da distribuição t de Student com ν graus de liberdade e é tal que:

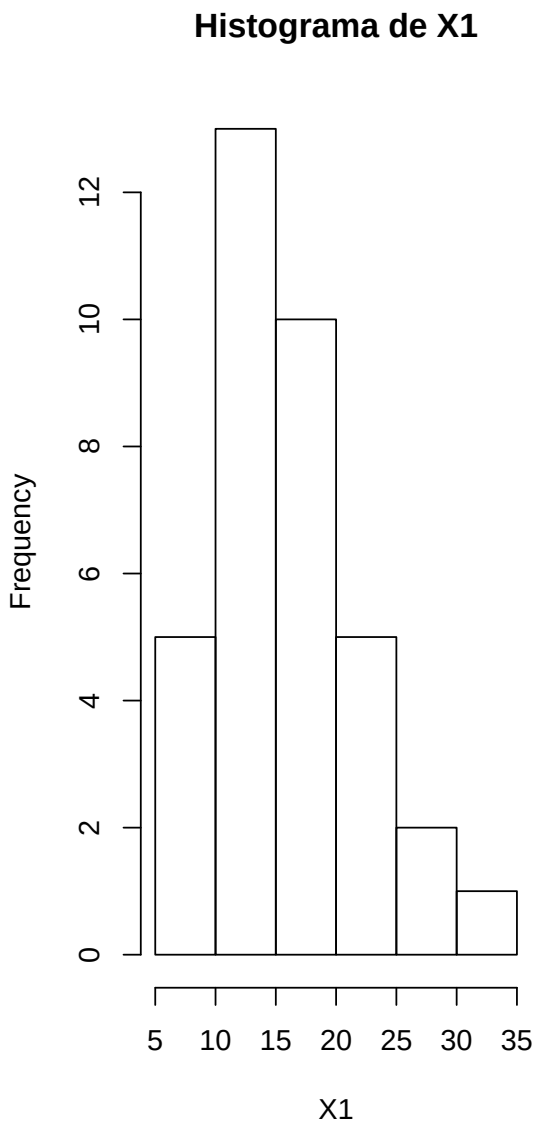
$$P\left(T > t_{\frac{\alpha}{2}}\right) = \frac{\alpha}{2}$$

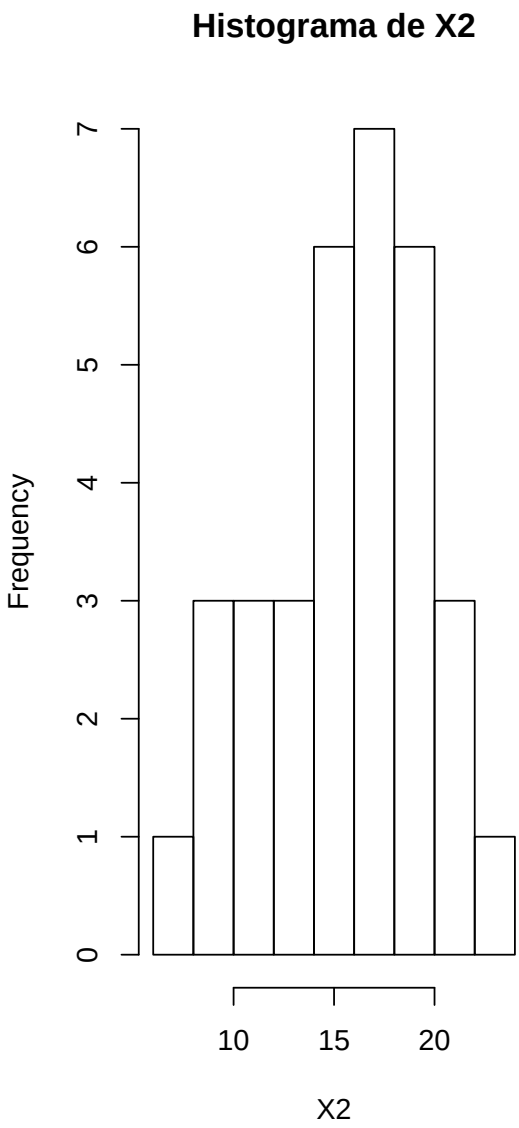
O valor-p é dado por:

$$\text{p-valor} = 2P(T > |t_0|\,|H_0)$$

Para os testes unilaterais, a região crítica e o valor-p são obtidos de maneira análoga ao caso do teste com variância conhecida, utilizando, no entanto, a distribuição t de Student, com ν graus de liberdade como referência.

```
# Amostras Independentes com variancias diferentes

# Gerando amostra de 21 obs da dist normal(15,6)
X1 <- rnorm(36,mean=15,sd=6)
# Gerando amostra de 21 obs da dist normal(16,4)
X2 <- rnorm(33,mean=16,sd=4)

# Histogramas:
par(mfrow=c(1,2))
hist(X1,main="Histograma de X1")
hist(X2,main="Histograma de X2")

# Aplicando o teste:
t.test(X1,X2,mu=0,paired=FALSE,var.equal=FALSE)

##
##  Welch Two Sample t-test
##
## data:  X1 and X2
## t = 0.1645, df = 61.943, p-value = 0.8698
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.219328  2.617449
## sample estimates:
## mean of x mean of y
##  16.01870  15.81963
```


Histograma de X1
Frequency
0
2
4
6
8
10
12
5
10
15
20
25
30
35
X1


Histograma de X2
Frequency
0
1
2
3
4
5
6
7
10
15
20
X2

A Apêndice

Coefficientes a_i para o teste Shapiro-Wilk para normalidade

i \ n	2	3	4	5	6	7	8	9	10
1	0.7071	0.7071	0.6872	0.6646	0.6431	0.6233	0.6052	0.5888	0.5739
2	-	0.0000	0.1677	0.2413	0.2806	0.3031	0.3164	0.3244	0.3291
3	-	-	-	0.0000	0.0875	0.1401	0.1743	0.1976	0.2141
4	-	-	-	-	-	0.0000	0.0561	0.0947	0.1224
5	-	-	-	-	-	-	-	0.0000	0.0399

i \ n	11	12	13	14	15	16	17	18	19	20
1	0.5601	0.5475	0.5359	0.5251	0.5150	0.5056	0.4968	0.4886	0.4808	0.4734
2	0.3315	0.3325	0.3325	0.3318	0.3306	0.3290	0.3273	0.3253	0.3232	0.3211
3	0.2260	0.2347	0.2412	0.2460	0.2495	0.2521	0.2540	0.2553	0.2561	0.2565
4	0.1429	0.1586	0.1707	0.1802	0.1878	0.1939	0.1988	0.2027	0.2059	0.2085
5	0.0695	0.0922	0.1099	0.1240	0.1353	0.1447	0.1524	0.1587	0.1641	0.1686
6	0.0000	0.0303	0.0539	0.0727	0.0880	0.1005	0.1109	0.1197	0.1271	0.1334
7	-	-	0.0000	0.0240	0.0433	0.0593	0.0725	0.0837	0.0932	0.1013
8	-	-	-	-	0.0000	0.0196	0.0359	0.0496	0.0612	0.0711
9	-	-	-	-	-	-	0.0000	0.0163	0.0303	0.0422
10	-	-	-	-	-	-	-	-	0.0000	0.0140

i \ n	21	22	23	24	25	26	27	28	29	30
1	0.4643	0.4590	0.4542	0.4493	0.4450	0.4407	0.4366	0.4328	0.4291	0.4254
2	0.3185	0.3156	0.3126	0.3098	0.3069	0.3043	0.3018	0.2992	0.2968	0.2944
3	0.2578	0.2571	0.2563	0.2554	0.2543	0.2533	0.2522	0.2510	0.2499	0.2487
4	0.2119	0.2131	0.2139	0.2145	0.2148	0.2151	0.2152	0.2151	0.2150	0.2148
5	0.1736	0.1764	0.1787	0.1807	0.1822	0.1836	0.1848	0.1857	0.1864	0.1870
6	0.1399	0.1443	0.1480	0.1512	0.1539	0.1563	0.1584	0.1601	0.1616	0.1630
7	0.1092	0.1150	0.1201	0.1245	0.1283	0.1316	0.1346	0.1372	0.1395	0.1415
8	0.0804	0.0878	0.0941	0.0997	0.1046	0.1089	0.1128	0.1162	0.1192	0.1219
9	0.0530	0.0618	0.0696	0.0764	0.0823	0.0876	0.0923	0.0965	0.1002	0.1036
10	0.0263	0.0368	0.0459	0.0539	0.0610	0.0672	0.0728	0.0778	0.0822	0.0862
11	0.0000	0.0122	0.0228	0.0321	0.0403	0.0476	0.0540	0.0598	0.0650	0.0697
12	-	-	0.0000	0.0107	0.0200	0.0284	0.0358	0.0424	0.0483	0.0537
13	-	-	-	-	0.0000	0.0094	0.0178	0.0253	0.0320	0.0381
14	-	-	-	-	-	-	0.0000	0.0084	0.0159	0.0227
15	-	-	-	-	-	-	-	-	0.0000	0.0076

i \ n	31	32	33	34	35	36	37	38	39	40
1	0.4220	0.4188	0.4156	0.4127	0.4096	0.4068	0.4040	0.4015	0.3989	0.3964
2	0.2921	0.2898	0.2876	0.2854	0.2834	0.2813	0.2794	0.2774	0.2755	0.2737
3	0.2475	0.2462	0.2451	0.2439	0.2427	0.2415	0.2403	0.2391	0.2380	0.2368
4	0.2145	0.2141	0.2137	0.2132	0.2127	0.2121	0.2116	0.2110	0.2104	0.2098
5	0.1874	0.1878	0.1880	0.1882	0.1883	0.1883	0.1883	0.1881	0.1880	0.1878
6	0.1641	0.1651	0.1660	0.1667	0.1673	0.1678	0.1683	0.1686	0.1689	0.1691
7	0.1433	0.1449	0.1463	0.1475	0.1487	0.1496	0.1505	0.1513	0.1520	0.1526
8	0.1243	0.1265	0.1284	0.1301	0.1317	0.1331	0.1344	0.1356	0.1366	0.1376
9	0.1066	0.1093	0.1116	0.1140	0.1160	0.1179	0.1196	0.1211	0.1225	0.1237
10	0.0899	0.0931	0.0961	0.0988	0.1013	0.1036	0.1056	0.1075	0.1092	0.1108
11	0.0739	0.0777	0.0812	0.0844	0.0873	0.0900	0.0924	0.0947	0.0967	0.0986
12	0.0585	0.0629	0.0669	0.0706	0.0739	0.0770	0.0798	0.0824	0.0848	0.0870
13	0.0435	0.0485	0.0530	0.0572	0.0610	0.0645	0.0677	0.0706	0.0733	0.0759
14	0.0289	0.0344	0.0395	0.0441	0.0484	0.0523	0.0559	0.0592	0.0622	0.0651
15	0.0144	0.0206	0.0262	0.0314	0.0361	0.0404	0.0444	0.0481	0.0515	0.0546
16	0.0000	0.0068	0.0131	0.0187	0.0239	0.0287	0.0331	0.0372	0.0409	0.0444
17	-	-	0.0000	0.0062	0.0119	0.0172	0.0220	0.0264	0.0305	0.0343
18	-	-	-	-	0.0000	0.0057	0.0110	0.0158	0.0203	0.0244
19	-	-	-	-	-	-	0.0000	0.0053	0.0101	0.0146
20	-	-	-	-	-	-	-	-	0.0000	0.0049

i \ n	41	42	43	44	45	46	47	48	49	50
1	0.3940	0.3917	0.3894	0.3872	0.3850	0.3830	0.3808	0.3789	0.3770	0.3751
2	0.2719	0.2701	0.2684	0.2667	0.2651	0.2635	0.2620	0.2604	0.2589	0.2574
3	0.2357	0.2345	0.2334	0.2323	0.2313	0.2302	0.2291	0.2281	0.2271	0.2260
4	0.2091	0.2085	0.2078	0.2072	0.2065	0.2058	0.2052	0.2045	0.2038	0.2032
5	0.1876	0.1874	0.1871	0.1868	0.1865	0.1862	0.1859	0.1855	0.1851	0.1847
6	0.1693	0.1694	0.1695	0.1695	0.1695	0.1695	0.1695	0.1693	0.1692	0.1691
7	0.1531	0.1535	0.1539	0.1542	0.1545	0.1548	0.1550	0.1551	0.1553	0.1554
8	0.1384	0.1392	0.1398	0.1405	0.1410	0.1415	0.1420	0.1423	0.1427	0.1430
9	0.1249	0.1259	0.1269	0.1278	0.1286	0.1293	0.1300	0.1306	0.1312	0.1317
10	0.1123	0.1136	0.1149	0.1160	0.1170	0.1180	0.1189	0.1197	0.1205	0.1212
11	0.1004	0.1020	0.1035	0.1049	0.1062	0.1073	0.1085	0.1095	0.1105	0.1113
12	0.0891	0.0909	0.0927	0.0943	0.0959	0.0972	0.0986	0.0998	0.1010	0.1020
13	0.0782	0.0804	0.0824	0.0842	0.0860	0.0876	0.0892	0.0906	0.0919	0.0932
14	0.0677	0.0701	0.0724	0.0745	0.0765	0.0783	0.0801	0.0817	0.0832	0.0846
15	0.0575	0.0602	0.0628	0.0651	0.0673	0.0694	0.0713	0.0731	0.0748	0.0764
16	0.0476	0.0506	0.0534	0.0560	0.0584	0.0607	0.0628	0.0648	0.0667	0.0685
17	0.0379	0.0411	0.0442	0.0471	0.0497	0.0522	0.0546	0.0568	0.0588	0.0608
18	0.0283	0.0318	0.0352	0.0383	0.0412	0.0439	0.0465	0.0489	0.0511	0.0532
19	0.0188	0.0227	0.0263	0.0296	0.0328	0.0357	0.0385	0.0411	0.0436	0.0459
20	0.0094	0.0136	0.0175	0.0211	0.0245	0.0277	0.0307	0.0335	0.0361	0.0386
21	0.0000	0.0045	0.0087	0.0126	0.0163	0.0197	0.0229	0.0259	0.0288	0.0314
22	-	-	0.0000	0.0042	0.0081	0.0118	0.0153	0.0185	0.0215	0.0244
23	-	-	-	-	0.0000	0.0039	0.0076	0.0111	0.0143	0.0174
24	-	-	-	-	-	-	0.0000	0.0037	0.0071	0.0104
25	-	-	-	-	-	-	-	-	0.0000	0.0035

B Apêndice

Quantis do teste Shapiro-Wilk para normalidade (valores de W tal que $100p\%$ da distribuição de $W < W_p$

n	$W_{0.01}$	$W_{0.02}$	$W_{0.05}$	$W_{0.10}$	$W_{0.50}$
3	0.753	0.756	0.767	0.789	0.959
4	0.687	0.707	0.748	0.792	0.935
5	0.686	0.715	0.762	0.806	0.927
6	0.713	0.743	0.788	0.826	0.927
7	0.730	0.760	0.803	0.838	0.928
8	0.749	0.778	0.818	0.851	0.932
9	0.764	0.791	0.829	0.859	0.935
10	0.781	0.806	0.842	0.869	0.938
11	0.792	0.817	0.850	0.876	0.940
12	0.805	0.828	0.859	0.883	0.943
13	0.814	0.837	0.866	0.889	0.945
14	0.825	0.846	0.874	0.895	0.947
15	0.835	0.855	0.881	0.901	0.950
16	0.844	0.863	0.887	0.906	0.952
17	0.851	0.869	0.892	0.910	0.954
18	0.858	0.874	0.897	0.914	0.956
19	0.863	0.879	0.901	0.917	0.957
20	0.868	0.884	0.905	0.920	0.959
21	0.873	0.888	0.908	0.923	0.960
22	0.878	0.892	0.911	0.926	0.961
23	0.881	0.895	0.914	0.928	0.962
24	0.884	0.898	0.916	0.930	0.963
25	0.886	0.901	0.918	0.931	0.964
26	0.891	0.904	0.920	0.933	0.965
27	0.894	0.906	0.923	0.935	0.965
28	0.896	0.908	0.924	0.936	0.966
29	0.898	0.910	0.926	0.937	0.966
30	0.900	0.912	0.927	0.939	0.967
31	0.902	0.914	0.929	0.940	0.967
32	0.904	0.915	0.930	0.941	0.968
33	0.906	0.917	0.931	0.942	0.968
34	0.908	0.919	0.933	0.943	0.969
35	0.910	0.920	0.934	0.944	0.969
36	0.912	0.922	0.935	0.945	0.970
37	0.914	0.924	0.936	0.946	0.970
38	0.916	0.925	0.938	0.947	0.971
39	0.917	0.927	0.939	0.948	0.971
40	0.919	0.928	0.940	0.949	0.972
41	0.920	0.929	0.941	0.950	0.972
42	0.922	0.930	0.942	0.951	0.972
43	0.923	0.932	0.943	0.951	0.973
44	0.924	0.933	0.944	0.952	0.973
45	0.926	0.934	0.945	0.953	0.973
46	0.927	0.935	0.945	0.953	0.974
47	0.928	0.936	0.946	0.954	0.974
48	0.929	0.937	0.947	0.954	0.974
49	0.929	0.937	0.947	0.955	0.974
50	0.930	0.938	0.947	0.955	0.974

C Apêndice
Tabela do Teste de Kolmogov-Smirnov

SAMPLE SIZE (N)	LEVEL OF SIGNIFICANCE FOR D = MAXIMUM [$F_0(X)$ - $S_n(X)$]				
	.20	.15	.10	.05	.01
1	.900	.925	.950	.975	.995
2	.684	.726	.776	.842	.929
3	.565	.597	.642	.708	.828
4	.494	.525	.564	.624	.733
5	.446	.474	.510	.565	.669
6	.410	.436	.470	.521	.618
7	.381	.405	.438	.486	.577
8	.358	.381	.411	.457	.543
9	.339	.360	.388	.432	.514
10	.322	.342	.368	.410	.490
11	.307	.326	.352	.391	.468
12	.295	.313	.338	.375	.450
13	.284	.302	.325	.361	.433
14	.274	.292	.314	.349	.418
15	.266	.283	.304	.338	.404
16	.258	.274	.295	.328	.392
17	.250	.266	.286	.318	.381
18	.244	.259	.278	.309	.371
19	.237	.252	.272	.301	.363
20	.231	.246	.264	.294	.356
25	.210	.220	.240	.270	.320
30	.190	.200	.220	.240	.290
35	.180	.190	.210	.230	.270
OVER 35	$\frac{1.07}{\sqrt{N}}$	$\frac{1.14}{\sqrt{N}}$	$\frac{1.22}{\sqrt{N}}$	$\frac{1.36}{\sqrt{N}}$	$\frac{1.63}{\sqrt{N}}$

D Apêndice
Tabela Siegel

Tábua G. Valores Críticos de T na Prova de Wilcoxon*

N	Nível de significância para prova unilateral		
	0,025	0,01	0,005
	Nível de significância para prova bilateral		
	0,05	0,02	0,01
6	0	—	—
7	2	0	—
8	4	2	0
9	6	3	2
10	8	5	3
11	11	7	5
12	14	10	7
13	17	13	10
14	21	16	13
15	25	20	16
16	30	24	20
17	35	28	23
18	40	33	28
19	46	38	32
20	52	43	38
21	59	49	43
22	66	56	49
23	73	62	55
24	81	69	61
25	89	77	68

E Apêndice
Tabela de Valores Críticos de U - Mann-Whitney

		n																		
α	m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0.10	1																				
	2																				
	3		0	1																	
	4		0	1	3																
	5		1	2	4	5															
	6		1	3	5	7	9														
	7		1	4	6	8	11	13													
	8		2	5	7	10	13	16	19												
	9	0	2	5	9	12	15	18	22	25											
	10	0	3	6	10	13	17	21	24	28	32										
	11	0	3	7	11	15	19	23	27	31	36	40									
	12	0	4	8	12	17	21	26	30	35	39	44	49								
	13	0	4	9	13	18	23	28	33	38	43	48	53	58							
	14	0	5	10	15	20	25	31	36	41	47	52	58	63	69						
	15	0	5	10	16	22	27	33	39	45	51	57	63	68	74	80					
	16	0	5	11	17	23	29	36	42	48	54	61	67	74	80	86	93				
	17	0	6	12	18	25	31	38	45	52	58	65	72	79	85	92	99	106			
	18	0	6	13	20	27	34	41	48	55	62	69	77	84	91	98	106	113	120		
	19	1	7	14	21	28	36	43	51	58	66	73	81	89	97	104	112	120	128	135	
	20	1	7	15	22	30	38	46	54	62	70	78	86	94	102	110	119	127	135	143	151
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0.05	1																				
	2																				
	3			0																	
	4			0	1																
	5		0	1	2	4															
	6		0	2	3	5	7														
	7		0	2	4	6	8	11													
	8		1	3	5	8	10	13	15												
	9		1	4	6	9	12	15	18	21											
	10		1	4	7	11	14	17	20	24	27										
	11		1	5	8	12	16	19	23	27	31	34									
	12		2	5	9	13	17	21	26	30	34	38	42								
	13		2	6	10	15	19	24	28	33	37	42	47	51							
	14		3	7	11	16	21	26	31	36	41	46	51	56	61						
	15		3	7	12	18	23	28	33	39	44	50	55	61	66	72					
	16		3	8	14	19	25	30	36	42	48	54	60	65	71	77	83				
	17		3	9	15	20	26	33	39	45	51	57	64	70	77	83	89	96			
	18		4	9	16	22	28	35	41	48	55	61	68	75	82	88	95	102	109		
	19	0	4	10	17	23	30	37	44	51	58	65	72	80	87	94	101	109	116	123	
	20	0	4	11	18	25	32	39	47	54	62	69	77	84	92	100	107	115	123	130	138
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

F Apêndice
Tabela de Valores Críticos de Z

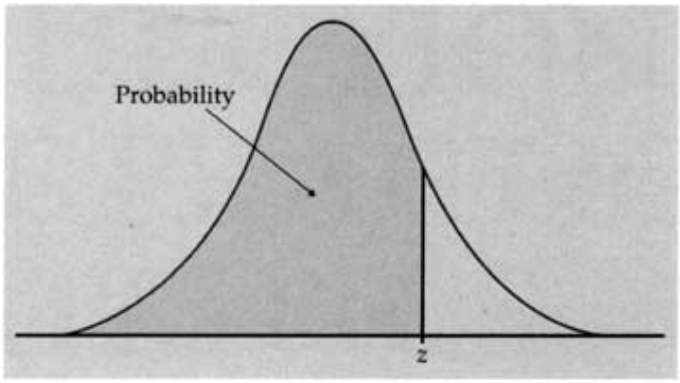


Table entry for z is the area under the standard normal curve to the left of z.

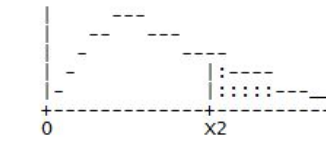
TABLE A Standard normal probabilities (*continued*)

z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
0.0	.5000	.5040	.5080	.5120	.5160	.5199	.5239	.5279	.5319	.5359
0.1	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5753
0.2	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
0.3	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
0.4	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	.6879
0.5	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
0.6	.7257	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7517	.7549
0.7	.7580	.7611	.7642	.7673	.7704	.7734	.7764	.7794	.7823	.7852
0.8	.7881	.7910	.7939	.7967	.7995	.8023	.8051	.8078	.8106	.8133
0.9	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8365	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8531	.8554	.8577	.8599	.8621
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.9015
1.3	.9032	.9049	.9066	.9082	.9099	.9115	.9131	.9147	.9162	.9177
1.4	.9192	.9207	.9222	.9236	.9251	.9265	.9279	.9292	.9306	.9319
1.5	.9332	.9345	.9357	.9370	.9382	.9394	.9406	.9418	.9429	.9441
1.6	.9452	.9463	.9474	.9484	.9495	.9505	.9515	.9525	.9535	.9545
1.7	.9554	.9564	.9573	.9582	.9591	.9599	.9608	.9616	.9625	.9633
1.8	.9641	.9649	.9656	.9664	.9671	.9678	.9686	.9693	.9699	.9706
1.9	.9713	.9719	.9726	.9732	.9738	.9744	.9750	.9756	.9761	.9767
2.0	.9772	.9778	.9783	.9788	.9793	.9798	.9803	.9808	.9812	.9817
2.1	.9821	.9826	.9830	.9834	.9838	.9842	.9846	.9850	.9854	.9857
2.2	.9861	.9864	.9868	.9871	.9875	.9878	.9881	.9884	.9887	.9890
2.3	.9893	.9896	.9898	.9901	.9904	.9906	.9909	.9911	.9913	.9916
2.4	.9918	.9920	.9922	.9925	.9927	.9929	.9931	.9932	.9934	.9936
2.5	.9938	.9940	.9941	.9943	.9945	.9946	.9948	.9949	.9951	.9952
2.6	.9953	.9955	.9956	.9957	.9959	.9960	.9961	.9962	.9963	.9964
2.7	.9965	.9966	.9967	.9968	.9969	.9970	.9971	.9972	.9973	.9974
2.8	.9974	.9975	.9976	.9977	.9977	.9978	.9979	.9979	.9980	.9981
2.9	.9981	.9982	.9982	.9983	.9984	.9984	.9985	.9985	.9986	.9986
3.0	.9987	.9987	.9987	.9988	.9988	.9989	.9989	.9989	.9990	.9990
3.1	.9990	.9991	.9991	.9991	.9992	.9992	.9992	.9992	.9993	.9993
3.2	.9993	.9993	.9994	.9994	.9994	.9994	.9994	.9995	.9995	.9995
3.3	.9995	.9995	.9995	.9996	.9996	.9996	.9996	.9996	.9996	.9997
3.4	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9998

G Apêndice

Tabela de Valores Críticos da Distribuição Qui-Quadrado, χ^2

VALUES OF CHI-SQUARE (ALPHA) OF THE CHI-SQUARE DISTRIBUTION
(CHI-SQUARE TABLE)



DF	X2(.995)	X2(.99)	X2(.975)	X2(.95)	X2(.05)	X2(.025)	X2(.01)	X2(.005)
1	0.000	0.000	0.001	0.004	3.841	5.024	6.635	7.879
2	0.010	0.020	0.051	0.103	5.991	7.378	9.210	10.597
3	0.072	0.115	0.216	0.352	7.815	9.348	11.345	12.838
4	0.207	0.297	0.484	0.711	9.488	11.143	13.277	14.860
5	0.412	0.554	0.831	1.145	11.071	12.833	15.086	16.750
6	0.676	0.872	1.237	1.635	12.592	14.449	16.812	18.548
7	0.989	1.239	1.690	2.167	14.067	16.013	18.475	20.278
8	1.344	1.646	2.180	2.733	15.507	17.535	20.090	21.955
9	1.735	2.088	2.700	3.325	16.919	19.023	21.666	23.589
10	2.156	2.558	3.247	3.940	18.307	20.483	23.209	25.188
11	2.603	3.053	3.816	4.575	19.675	21.920	24.725	26.757
12	3.074	3.571	4.404	5.226	21.026	23.337	26.217	28.300
13	3.565	4.107	5.009	5.892	22.362	24.736	27.688	29.819
14	4.075	4.660	5.629	6.571	23.685	26.119	29.141	31.319
15	4.601	5.229	6.262	7.261	24.996	27.488	30.578	32.801
16	5.142	5.812	6.908	7.962	26.296	28.845	32.000	34.267
17	5.697	6.408	7.564	8.672	27.587	30.191	33.409	35.718
18	6.265	7.015	8.231	9.390	28.869	31.526	34.805	37.156
19	6.844	7.633	8.907	10.117	30.144	32.852	36.191	38.582
20	7.434	8.260	9.591	10.851	31.410	34.170	37.566	39.997
21	8.034	8.897	10.283	11.591	32.671	35.479	38.932	41.401
22	8.643	9.542	10.982	12.338	33.924	36.781	40.289	42.796
23	9.260	10.196	11.689	13.091	35.172	38.076	41.638	44.181
24	9.886	10.856	12.401	13.848	36.415	39.364	42.980	45.559
25	10.520	11.524	13.120	14.611	37.652	40.646	44.314	46.928
26	11.160	12.198	13.844	15.379	38.885	41.923	45.642	48.290
27	11.808	12.879	14.573	16.151	40.113	43.195	46.963	49.645
28	12.461	13.565	15.308	16.928	41.337	44.461	48.278	50.993
29	13.121	14.256	16.047	17.708	42.557	45.722	49.588	52.336
30	13.787	14.953	16.791	18.493	43.773	46.979	50.892	53.672
31	14.458	15.655	17.539	19.281	44.985	48.232	52.191	55.003
32	15.134	16.362	18.291	20.072	46.194	49.480	53.486	56.328
33	15.815	17.074	19.047	20.867	47.400	50.725	54.776	57.648
34	16.501	17.789	19.806	21.664	48.602	51.966	56.061	58.964
35	17.192	18.509	20.569	22.465	49.802	53.203	57.342	60.275
40	20.707	22.164	24.433	26.509	55.758	59.342	63.691	66.766
50	27.991	29.707	32.357	34.764	67.505	71.420	76.154	79.490
60	35.534	37.485	40.482	43.188	79.082	83.298	88.379	91.952
70	43.275	45.442	48.758	51.739	90.531	95.023	100.425	104.215
80	51.172	53.540	57.153	60.391	101.879	106.629	112.329	116.321
90	59.196	61.754	65.647	69.126	113.145	118.136	124.116	128.299
100	67.328	70.065	74.222	77.929	124.342	129.561	135.807	140.169
110	75.550	78.458	82.867	86.792	135.480	140.917	147.414	151.948
120	83.852	86.923	91.573	95.705	146.567	152.211	158.950	163.648

www.statext.com