

Curso

Ciência de Dados

aplicada à Saúde

Eduardo Ogasawara (CEFET/RJ)

<http://eic.cefet-rj.br/~eogasawara>

Vanderlei Pascoal de Matos (Fiocruz)

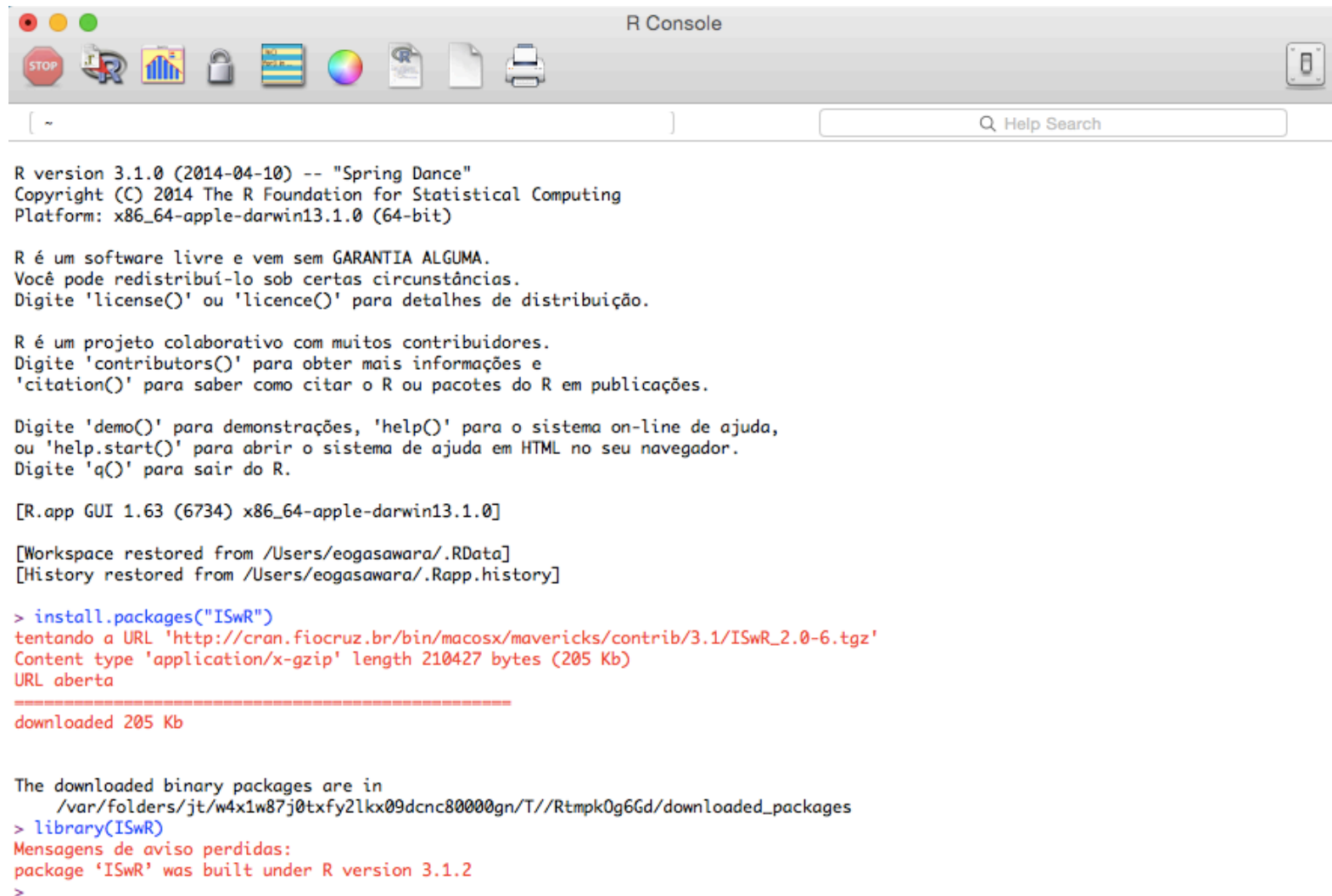
vanderlei.pascoal@icict.fiocruz.br

Noções Gerais de R

- Instalação do R e R Studio
- Pacotes R
- Básico do R
- Leitura de CSV
- Gráficos
- Funções
- Estrutura de Controle
- Estrutura de Repetição
- Estatística Básica
- Clustering

- R é uma linguagem e um ambiente de desenvolvimento integrado
- Foi criada por Ross Ihaka e Robert Gentleman na Universidade de Auckland, Nova Zelândia
- R veio da linguagem S (Bell Laboratories - AT&T)
- Instalação e ampla biblioteca de pacotes (CRAN)
<https://cran.r-project.org>
- A linguagem R é largamente usada entre estatísticos, data miners e cientistas de dados

R Console



```
R version 3.1.0 (2014-04-10) -- "Spring Dance"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.1.0 (64-bit)

R é um software livre e vem sem GARANTIA ALGUMA.
Você pode redistribuí-lo sob certas circunstâncias.
Digite 'license()' ou 'licence()' para detalhes de distribuição.

R é um projeto colaborativo com muitos contribuidores.
Digite 'contributors()' para obter mais informações e
'citation()' para saber como citar o R ou pacotes do R em publicações.

Digite 'demo()' para demonstrações, 'help()' para o sistema on-line de ajuda,
ou 'help.start()' para abrir o sistema de ajuda em HTML no seu navegador.
Digite 'q()' para sair do R.

[R.app GUI 1.63 (6734) x86_64-apple-darwin13.1.0]

[Workspace restored from /Users/eogasawara/.RData]
[History restored from /Users/eogasawara/.Rapp.history]

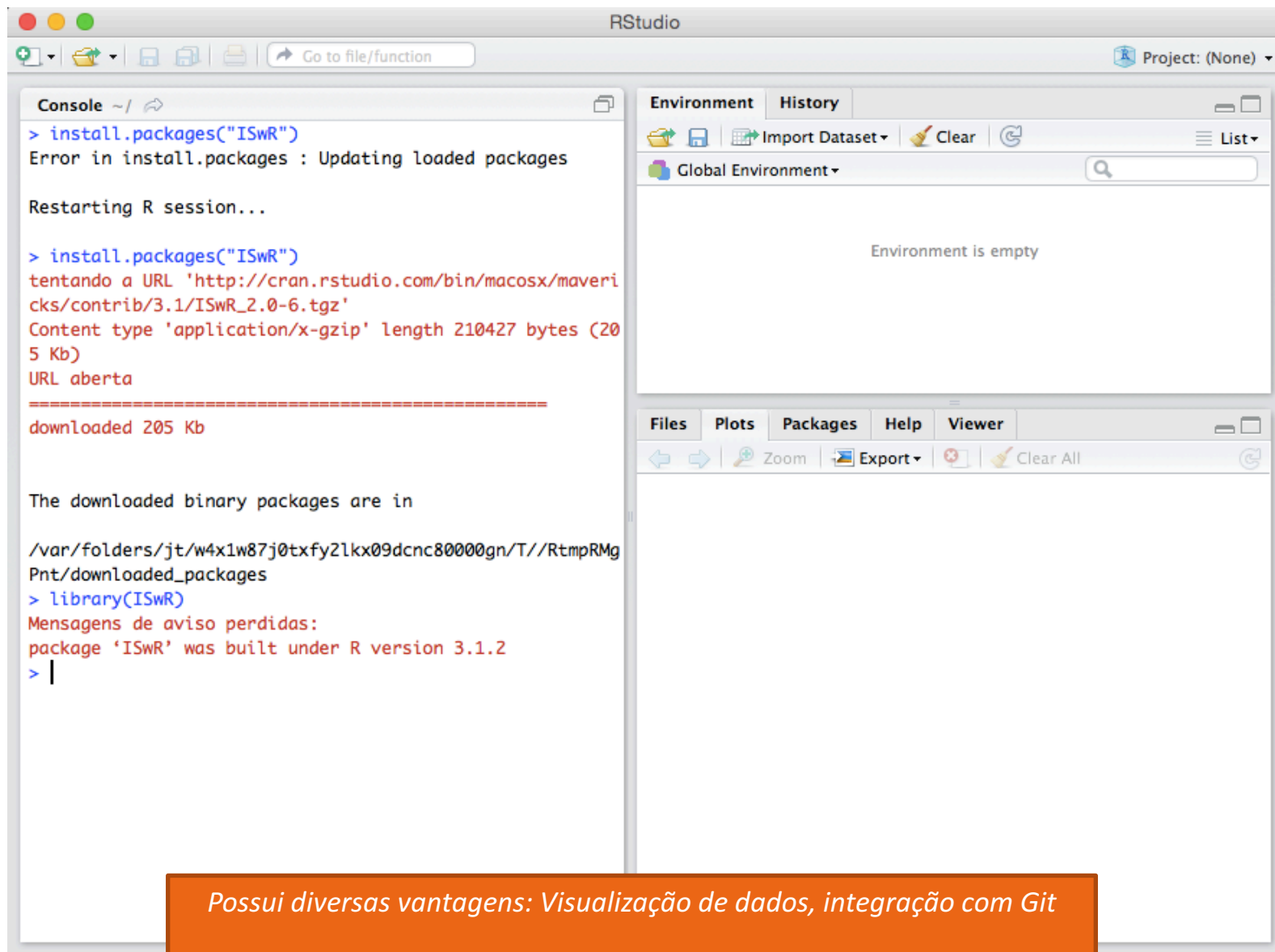
> install.packages("ISwR")
tentando a URL 'http://cran.fiocruz.br/bin/macosx/mavericks/contrib/3.1/ISwR_2.0-6.tgz'
Content type 'application/x-gzip' length 210427 bytes (205 Kb)
URL aberta
=====
downloaded 205 Kb

The downloaded binary packages are in
/var/folders/jt/w4x1w87j0txfy2lkx09dcnc80000gn/T//Rtmpk0g6Gd/downloaded_packages
> library(ISwR)
Mensagens de aviso perdidas:
package 'ISwR' was built under R version 3.1.2
>
```

Disponível para Mac, Windows e Linux

R Studio

<http://www.rstudio.com>



Possui diversas vantagens: Visualização de dados, integração com Git

Pacotes

- É o ponto forte do R
 - Existem mais de 4500 pacotes disponíveis feito por mais de 2000 colaboradores
 - <http://cran.r-project.org/web/packages/>

- Instalação de pacote:

```
install.packages("coefplot")  
install.packages("ISwR")  
install.packages("ggplot2")  
install.packages("TSPred")
```

- Carregamento:

```
require(coefplot)  
require(ISwR)  
require(ggplot2)  
require(TSPred)
```

Noções gerais


- Operações matemáticas:
- Atribuição:
- Exibição de valor:
- Teste lógico:

```
4 * (6 + 5)
```

```
x = 2
```

```
x
```

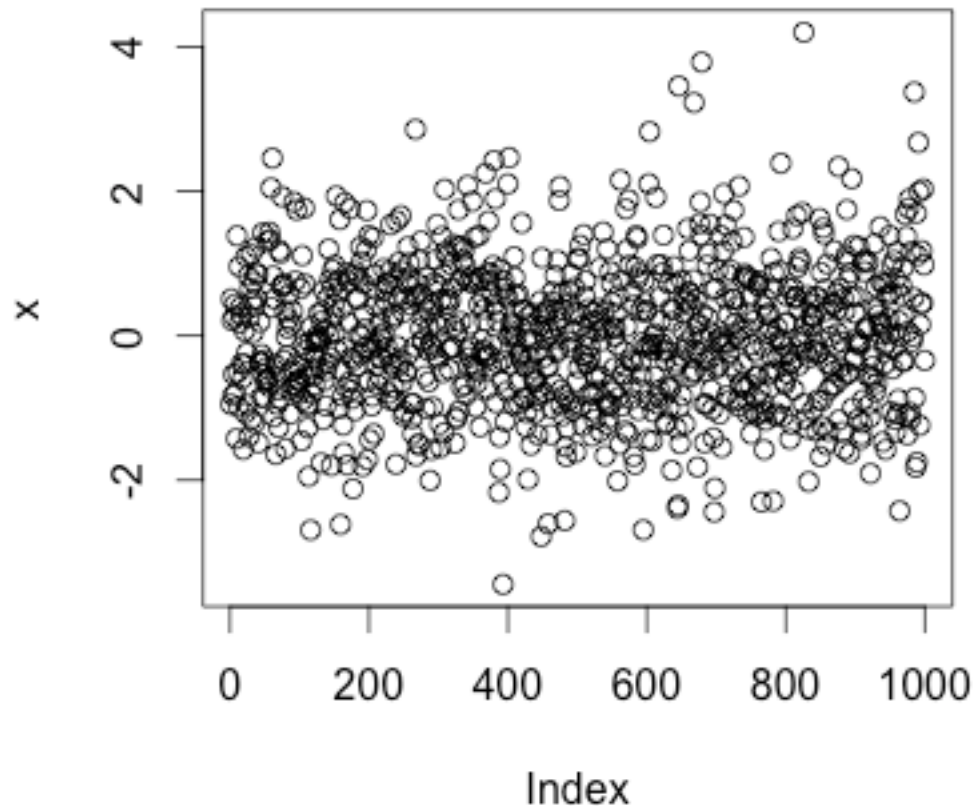
```
is.numeric(x)
```

```
Console ~/   
> 4 * (6 + 5)  
[1] 44  
>  
> x = 2  
>  
> x  
[1] 2  
>  
> is.numeric(x)  
[1] TRUE  
> |
```

Exibição de gráfico

- 1000 valores aleatórios com distribuição normal
- Plot do gráfico

```
x = rnorm(1000)  
plot(x)
```



Vetores

- Definição
- Visualização
- Operação com escalar
- Operação com vetores


```
y = c(3, 1, 3, 1, 3)
x = c(1, 2, 3, 4, 5)

x

x * 3

x

x + y
```

```
Console ~/ 
> y = c(3, 1, 3, 1, 3)
> x = c(1, 2, 3, 4, 5)
>
> x
[1] 1 2 3 4 5
>
> x * 3
[1] 3 6 9 12 15
>
> x
[1] 1 2 3 4 5
>
> x + y
[1] 4 3 6 5 8
> |
```

Aritmética vetorial

```
weight = c(60, 72, 57, 90, 95, 72)
weight
height = c(1.75, 1.80, 1.65, 1.90, 1.74, 1.91)
bmi = weight/height^2
bmi
```

Console ~/ ↻

```
> weight = c(60, 72, 57, 90, 95, 72)
> weight
[1] 60 72 57 90 95 72
>
> height = c(1.75, 1.80, 1.65, 1.90, 1.74, 1.91)
> bmi = weight/height^2
> bmi
[1] 19.59184 22.22222 20.93664 24.93075 31.37799 19.73630
> |
```


Média

```
sum(weight)  
sum(weight)/length(weight)  
mean(weight)
```

```
Console ~/ ↻  
> sum(weight)  
[1] 446  
> sum(weight)/length(weight)  
[1] 74.33333  
> mean(weight)  
[1] 74.33333  
> |
```


Desvio padrão

```
xbar = sum(weight)/length(weight)
weight - xbar
(weight - xbar)^2
sum((weight - xbar)^2)
sqrt(sum((weight - xbar)^2)/(length(weight) - 1))
sd(weight)
```

```
Console ~/ 
> xbar = sum(weight)/length(weight)
> weight - xbar
[1] -14.333333 -2.333333 -17.333333 15.666667 20.666667
[6] -2.333333
> (weight - xbar)^2
[1] 205.444444 5.444444 300.444444 245.444444 427.111111
[6] 5.444444
> sum((weight - xbar)^2)
[1] 1189.333
> sqrt(sum((weight - xbar)^2)/(length(weight) - 1))
[1] 15.42293
> sd(weight)
[1] 15.42293
>
```

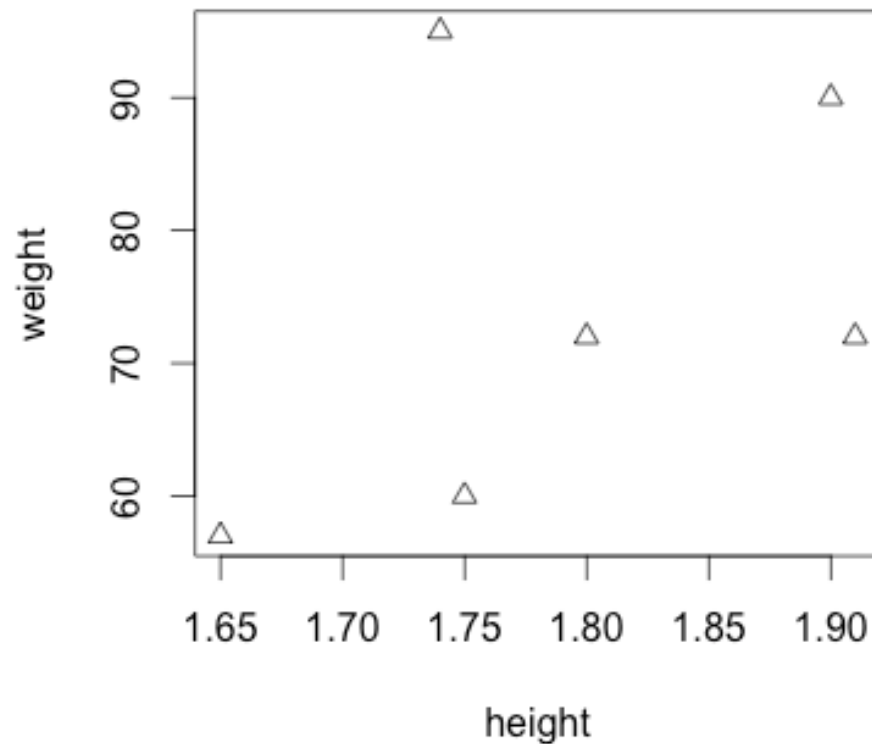
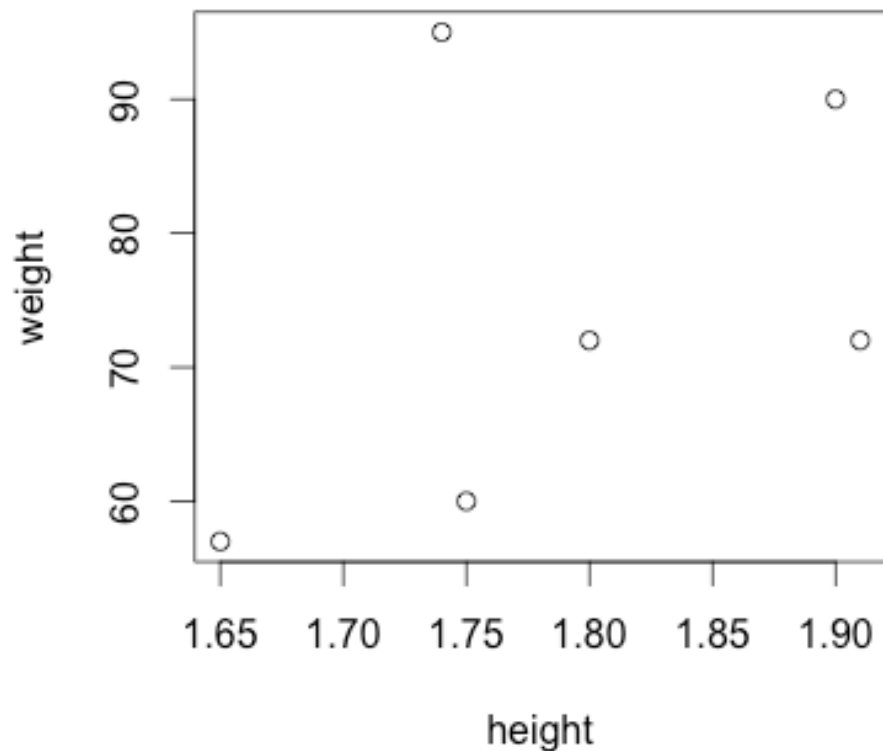
Uso de funções

```
mean(x)  
rnorm(15)
```

```
Console ~/   
> mean(x)  
[1] 3  
> rnorm(15)  
[1] -0.410529580 -0.138120217 0.006379301 0.907757756  
[5] 0.585317563 -0.283118260 0.754927111 1.534138123  
[9] -2.214438632 0.564672746 -2.397770549 0.485821572  
[13] -1.597648598 -0.896037029 -0.850080407  
> |
```

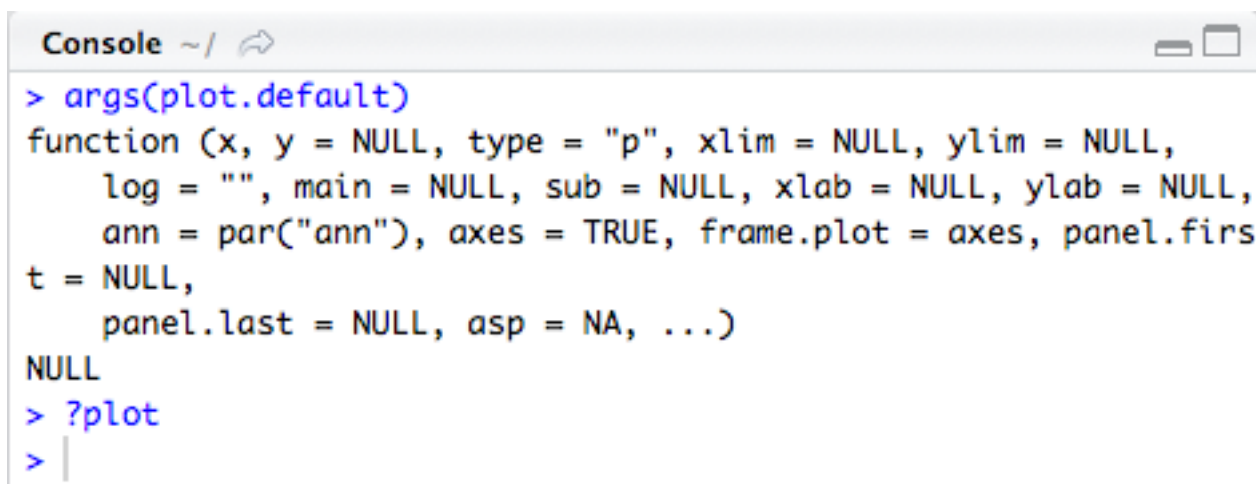
Parâmetro obrigatórios e com valor default

```
plot(height, weight)  
plot(height, weight, pch=2)
```



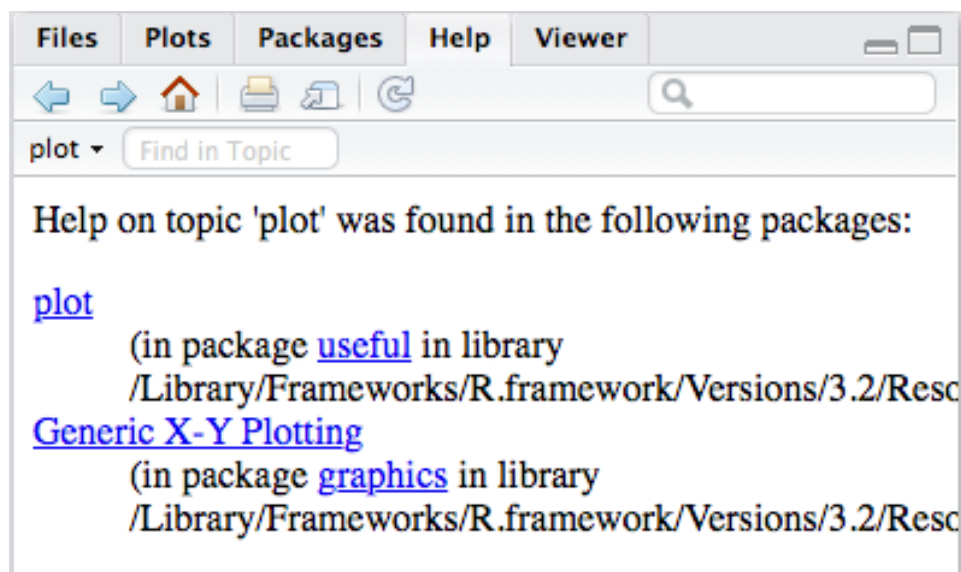
Argumentos default e ajuda da função

```
args(plot.default)  
?plot
```



Console ~/

```
> args(plot.default)  
function (x, y = NULL, type = "p", xlim = NULL, ylim = NULL,  
  log = "", main = NULL, sub = NULL, xlab = NULL, ylab = NULL,  
  ann = par("ann"), axes = TRUE, frame.plot = axes, panel.firs  
t = NULL,  
  panel.last = NULL, asp = NA, ...)  
NULL  
> ?plot  
> |
```



Files Plots Packages Help Viewer

plot ▾ Find in Topic

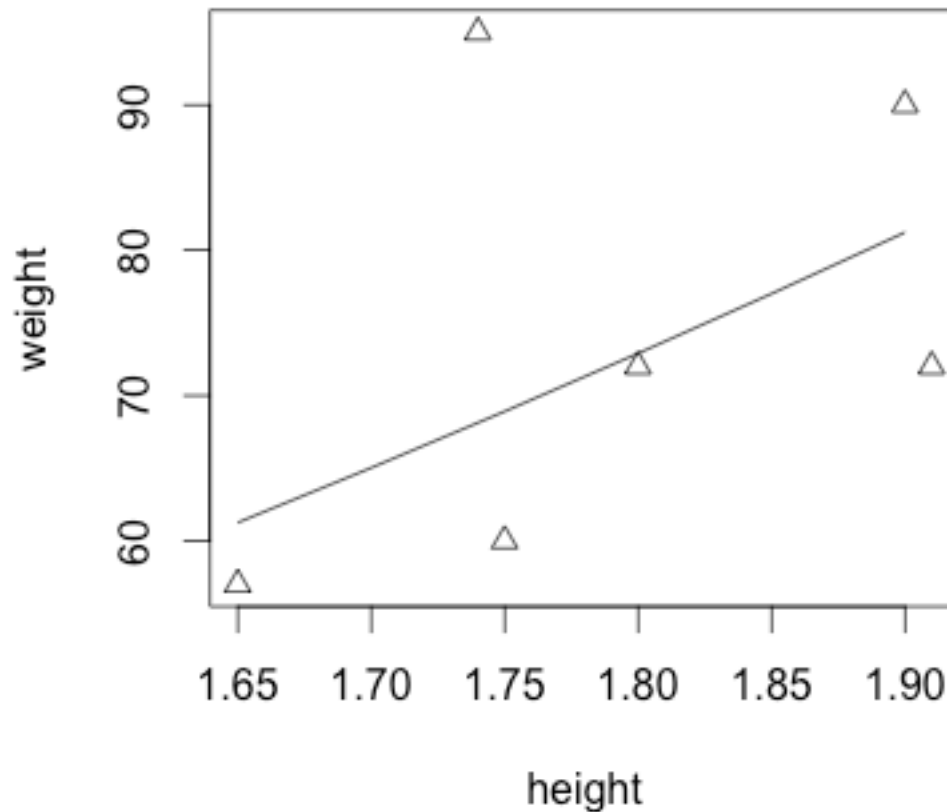
Help on topic 'plot' was found in the following packages:

[plot](#)
(in package [useful](#) in library
/Library/Frameworks/R.framework/Versions/3.2/Resc

[Generic X-Y Plotting](#)
(in package [graphics](#) in library
/Library/Frameworks/R.framework/Versions/3.2/Resc

Ultimo gráfico plotado continua com canvas ativo

```
hh = c(1.65, 1.70, 1.75, 1.80, 1.85, 1.90)  
lines(hh, 22.5 * hh^2)
```



Inferência Estatística

■ A média do BMI é igual ao valor teórico de 22.5?

- Assumimos que a distribuição de bmi é normal
- A hipótese nula é de que não diferença entre a média do bmi e o valor teórico esperado
- p-value default de 5%

```
t.test(bmi, mu=22.5)
```

Console ~/ ↩

```
> t.test(bmi, mu=22.5)
```

One Sample t-test

```
data:  bmi
```

```
t = 0.34488, df = 5, p-value = 0.7442
```

```
alternative hypothesis: true mean is not equal to 22.5
```

```
95 percent confidence interval:
```

```
18.41734 27.84791
```

```
sample estimates:
```

```
mean of x
```

```
23.13262
```

```
> |
```

O valor alto para p-value indica que não evidencias que refutem a hipótese nula.

Inferência Estatística

■ A média do BMI é igual ao valor teórico de 15?

- Assumimos que a distribuição de bmi é normal
- A hipótese nula é de que não diferença entre a média do bmi e o valor teórico esperado
- p-value default de 5%

```
t.test(bmi, mu=15)
```

```
Console ~/ ↵  
> t.test(bmi, mu=15)  
  
One Sample t-test  
  
data:  bmi  
t = 4.4336, df = 5, p-value = 0.006805  
alternative hypothesis: true mean is not equal to 15  
95 percent confidence interval:  
 18.41734 27.84791  
sample estimates:  
mean of x  
 23.13262  
  
>
```

O valor baixo para p-value refuta a hipótese nula. São diferentes.

Missing values

```
x = c(10, NA, 13)
```

```
mean(x)
```

```
mean(x, na.rm=TRUE)
```

Console ~/ ↻

```
> x = c(10, NA, 13)
```

```
>
```

```
> mean(x)
```

```
[1] NA
```

```
>
```

```
> mean(x, na.rm=TRUE)
```

```
[1] 11.5
```


```
> |
```

Nomes para elementos

```
x = c(red=1, blue=2, green=3)
x
```


```
names(x)
x["blue"]*x
```

```
names(x) = c("red", "green", "blue")
x["blue"]*x
```

```
Console ~/ 
> x = c(red=1, blue=2, green=3)
> x
  red  blue green
    1    2    3
>
> names(x)
[1] "red"  "blue" "green"
> x["blue"]*x
  red  blue green
    2    4    6
>
> names(x) = c("red", "green", "blue")
> x["blue"]*x
  red green  blue
    3    6    9
>
```

Matrices

```
x = 1:12  
dim(x) = c(3,4)  
x
```

Console ~/ 

```
> x = 1:12  
> dim(x) = c(3,4)  
> x  
      [,1] [,2] [,3] [,4]  
[1,]    1    4    7   10  
[2,]    2    5    8   11  
[3,]    3    6    9   12  
> |
```

Matrizes formadas por linhas e transposição

```
x = matrix(1:12,nrow=3,byrow=T)
rownames(x) = LETTERS[1:3]
x
t(x)
```

Console ~/ ↗

```
> x = matrix(1:12,nrow=3,byrow=T)
> rownames(x) = LETTERS[1:3]
> x
  [,1] [,2] [,3] [,4]
A    1    2    3    4
B    5    6    7    8
C    9   10   11   12
> t(x)
      A B  C
[1,] 1 5  9
[2,] 2 6 10
[3,] 3 7 11
[4,] 4 8 12
> |
```

Fatores

```
pain = c(0,3,2,2,1)
fpain = factor(pain,levels=0:3)
levels(fpain) = c("none","mild","medium","severe")

fpain

as.numeric(fpain)

levels(fpain)
```

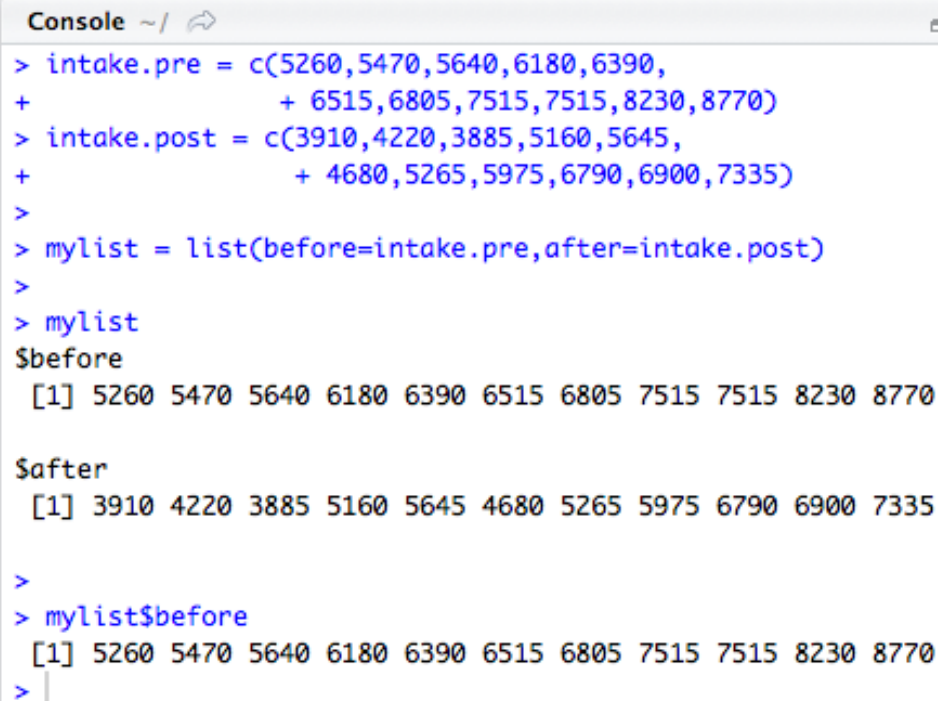
Console ~/ ↻



```
> pain = c(0,3,2,2,1)
> fpain = factor(pain,levels=0:3)
> levels(fpain) = c("none","mild","medium","severe")
>
> fpain
[1] none    severe medium medium mild
Levels: none mild medium severe
>
> as.numeric(fpain)
[1] 1 4 3 3 2
>
> levels(fpain)
[1] "none"  "mild"  "medium" "severe"
> |
```

Listas

```
intake.pre = c(5260,5470,5640,6180,6390,  
              + 6515,6805,7515,7515,8230,8770)  
intake.post = c(3910,4220,3885,5160,5645,  
               + 4680,5265,5975,6790,6900,7335)
```

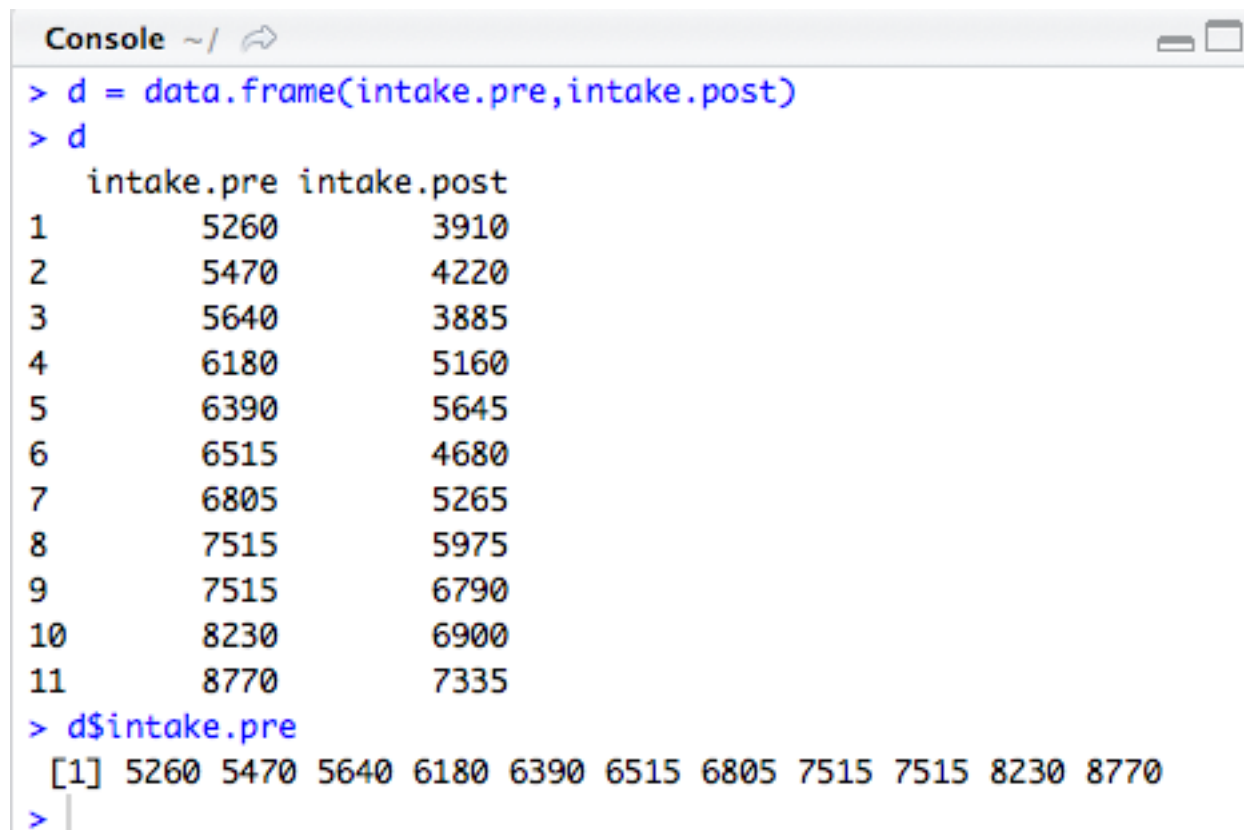
```
mylist = list(before=intake.pre,after=intake.post)  
mylist  
mylist$before
```






```
Console ~/    
> intake.pre = c(5260,5470,5640,6180,6390,  
+               + 6515,6805,7515,7515,8230,8770)  
> intake.post = c(3910,4220,3885,5160,5645,  
+               + 4680,5265,5975,6790,6900,7335)  
>  
> mylist = list(before=intake.pre,after=intake.post)  
>  
> mylist  
$before  
[1] 5260 5470 5640 6180 6390 6515 6805 7515 7515 8230 8770  
  
$after  
[1] 3910 4220 3885 5160 5645 4680 5265 5975 6790 6900 7335  
  
>  
> mylist$before  
[1] 5260 5470 5640 6180 6390 6515 6805 7515 7515 8230 8770  
> |
```


Frames

```
d = data.frame(intake.pre, intake.post)
d
d$intake.pre
```



The screenshot shows a R console window with the following content:

```
Console ~/   
```

```
> d = data.frame(intake.pre, intake.post)
> d
  intake.pre intake.post
1      5260      3910
2      5470      4220
3      5640      3885
4      6180      5160
5      6390      5645
6      6515      4680
7      6805      5265
8      7515      5975
9      7515      6790
10     8230      6900
11     8770      7335
> d$intake.pre
[1] 5260 5470 5640 6180 6390 6515 6805 7515 7515 8230 8770
> |
```

Seleção condicional

```
intake.post[intake.pre > 7000]
```

```
intake.post[intake.pre > 7000 | intake.pre < 6000]
```

```
d[d$intake.pre > 7000 | d$intake.pre < 6000,]
```

Console ~/ ↻

```
> intake.post[intake.pre > 7000]
```

```
[1] 5975 6790 6900 7335
```

```
>
```

```
> intake.post[intake.pre > 7000 | intake.pre < 6000]
```

```
[1] 3910 4220 3885 5975 6790 6900 7335
```

```
>
```

```
> d[d$intake.pre > 7000 | d$intake.pre < 6000,]
```

	intake.pre	intake.post	c	e
1	5260	3910	1350	1350
2	5470	4220	1250	1250
3	5640	3885	1755	1755
8	7515	5975	1540	1540
9	7515	6790	725	725
10	8230	6900	1330	1330
11	8770	7335	1435	1435

```
> |
```

Grupos

```
energy
exp.lean = energy$expend[energy$stature=="lean"]
exp.obese = energy$expend[energy$stature=="obese"]
l = split(energy$expend, energy$stature)
l
```

Console ~/ ↻		
> energy		
	expend	stature
1	9.21	obese
2	7.53	lean
3	7.48	lean
4	8.08	lean
5	8.09	lean
6	10.15	lean
7	8.40	lean
8	10.88	lean
9	6.13	lean
10	7.90	lean
11	11.51	obese
12	12.79	obese
13	7.05	lean
14	11.85	obese
15	9.97	obese
16	7.48	lean
17	8.79	obese

```
> exp.lean = energy$expend[energy$stature=="lean"]
> exp.obese = energy$expend[energy$stature=="obese"]
> l = split(energy$expend, energy$stature)
> l
$lean
 [1]  7.53  7.48  8.08  8.09 10.15  8.40 10.88  6.13  7.90
[10]  7.05  7.48  7.58  8.11

$obese
 [1]  9.21 11.51 12.79 11.85  9.97  8.79  9.69  9.68  9.19

> |
```

Laços implícitos – *sapply*, *lapply*

```
lapply(thuesen, mean, na.rm=T)
```

```
sapply(thuesen, mean, na.rm=T)
```

- l – list, s – simple (vetor ou matriz)
- o segundo argumento é a função
- os argumentos seguintes são aplicados a função

Console ~/ ↗

```
> lapply(thuesen, mean, na.rm=T)
```

```
$blood.glucose
```

```
[1] 10.3
```

```
$short.velocity
```

```
[1] 1.325652
```

```
>
```

```
> sapply(thuesen, mean, na.rm=T)
```

```
blood.glucose short.velocity
```

```
10.300000
```

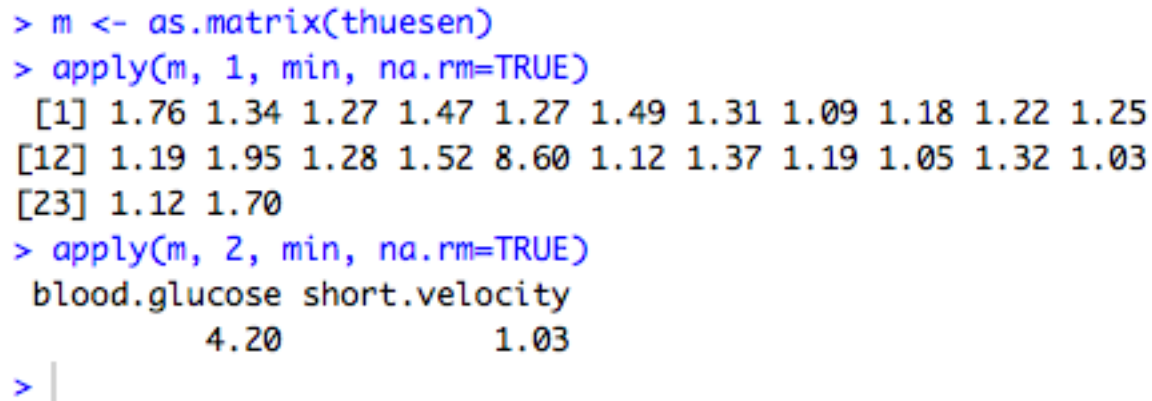
```
1.325652
```




```
> |
```

Laços implícitos - apply

```
m <- as.matrix(thuesen)
apply(m, 1, min, na.rm=TRUE)
apply(m, 2, min, na.rm=TRUE)
```

- na função apply, o segundo argumento refere-se: 1 – linha, 2 - coluna




```
Console ~/     
> m <- as.matrix(thuesen)  
> apply(m, 1, min, na.rm=TRUE)  
[1] 1.76 1.34 1.27 1.47 1.27 1.49 1.31 1.09 1.18 1.22 1.25  
[12] 1.19 1.95 1.28 1.52 8.60 1.12 1.37 1.19 1.05 1.32 1.03  
[23] 1.12 1.70  
> apply(m, 2, min, na.rm=TRUE)  
blood.glucose short.velocity  
4.20 1.03  
> |
```

Laços implícitos - tapply


```
tapply(energy$expend, energy$stature, median)
```

na função tapply, o segundo argumento é o vetor de fatores

```
Console ~/   
> tapply(energy$expend, energy$stature, median)  
lean obese  
7.90 9.69  
> |
```

Ordenação e ordem

```
intake$post
sort(intake$post)
order(intake$post)
o = order(intake$post)
intake$post[o]
intake$pre[o]
```

Console ~/ 

```
> intake$post
[1] 3910 4220 3885 5160 5645 4680 5265 5975 6790 6900 7335
> sort(intake$post)
[1] 3885 3910 4220 4680 5160 5265 5645 5975 6790 6900 7335
> order(intake$post)
[1] 3 1 2 6 4 7 5 8 9 10 11
> o = order(intake$post)
> intake$post[o]
[1] 3885 3910 4220 4680 5160 5265 5645 5975 6790 6900 7335
> intake$pre[o]
[1] 5640 5260 5470 6515 6180 6805 6390 7515 7515 8230 8770
> |
```

Ordenando um data frame

```
intake  
intake.sorted = intake[o,]  
intake.sorted
```

Console ~/ ↗

```
> intake  
   pre post  
1  5260 3910  
2  5470 4220  
3  5640 3885  
4  6180 5160  
5  6390 5645  
6  6515 4680  
7  6805 5265  
8  7515 5975  
9  7515 6790  
10 8230 6900  
11 8770 7335  
> intake.sorted = intake[o,]  
> |
```

Console ~/ ↗

```
> intake.sorted  
   pre post  
3  5640 3885  
1  5260 3910  
2  5470 4220  
6  6515 4680  
4  6180 5160  
7  6805 5265  
5  6390 5645  
8  7515 5975  
9  7515 6790  
10 8230 6900  
11 8770 7335  
> |
```


Funções


```
double.num = function(x)
{
    return(x * 2)
}
double.num(5)
```

Console ~/ ↗

```
> double.num = function(x)
+ {
+   return(x * 2)
+ }
> double.num(5)
[1] 10
> |
```

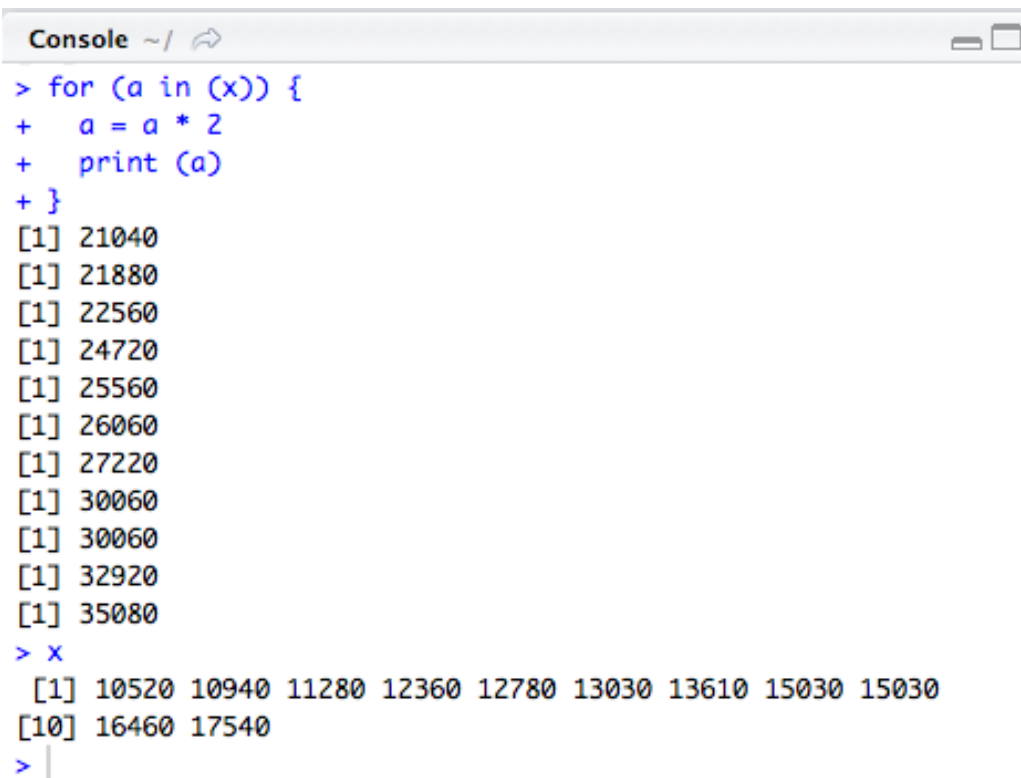
Condicionais

```
double.numpar = function(x)
{
  if (x %% 2 == 0) {
    return(x * 2)
  }
  else {
    return(x)
  }
}
double.numpar(5)
double.numpar(6)
```




```
Console ~/ 
> double.numpar = function(x)
+ {
+   if (x %% 2 == 0) {
+     return(x * 2)
+   }
+   else {
+     return(x)
+   }
+ }
> double.numpar(5)
[1] 5
> double.numpar(6)
[1] 12
> |
```

Estrutura de repetição

```
for (a in (x)) {  
  a = a * 2  
  print (a)  
}  
x
```

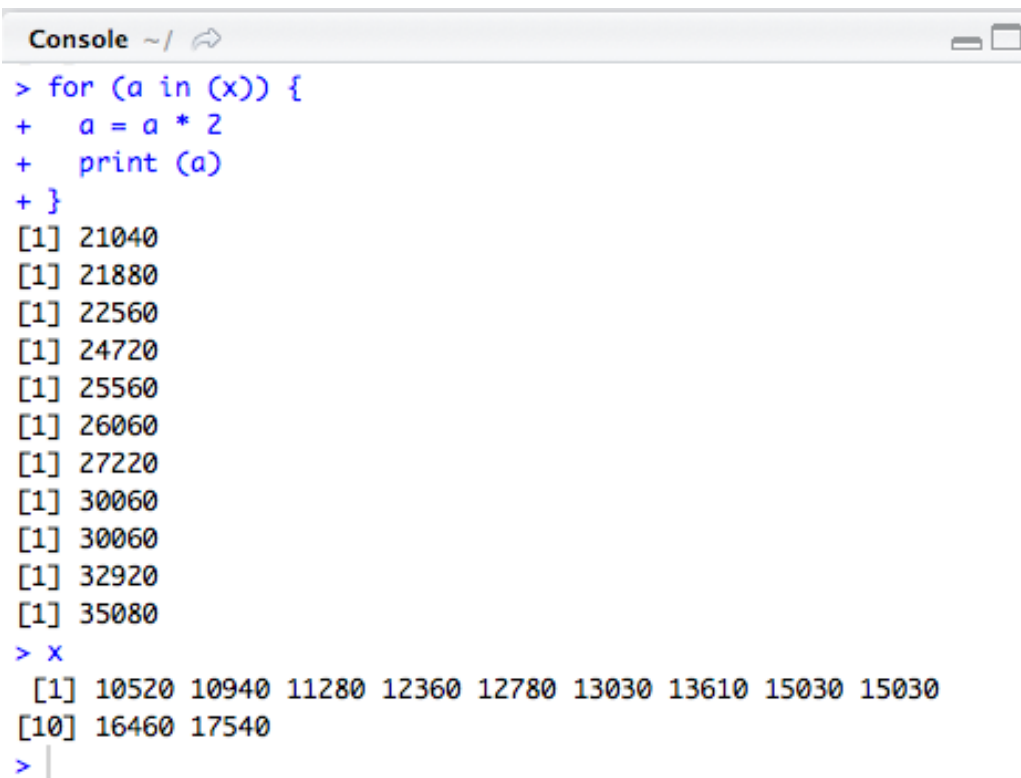


The screenshot shows a R console window with the following text:

```
Console ~/     
> for (a in (x)) {  
+   a = a * 2  
+   print (a)  
+ }  
[1] 21040  
[1] 21880  
[1] 22560  
[1] 24720  
[1] 25560  
[1] 26060  
[1] 27220  
[1] 30060  
[1] 30060  
[1] 32920  
[1] 35080  
> x  
[1] 10520 10940 11280 12360 12780 13030 13610 15030 15030  
[10] 16460 17540  
> |
```

Impressão

```
for (a in (x)) {  
  a = a * 2  
  print (a)  
}  
x
```



Console ~/ ↻

```
> for (a in (x)) {  
+   a = a * 2  
+   print (a)  
+ }  
[1] 21040  
[1] 21880  
[1] 22560  
[1] 24720  
[1] 25560  
[1] 26060  
[1] 27220  
[1] 30060  
[1] 30060  
[1] 32920  
[1] 35080  
> x  
[1] 10520 10940 11280 12360 12780 13030 13610 15030 15030  
[10] 16460 17540  
> |
```

Agregações

■ Plyr

- Split-apply-combine (agregações com cláusulas de group by)
- [i][o]ply – ddply – data frame de input e data frame de output

```
require(plyr)  
head(baseball)
```

```
> head(baseball)  
      id year stint team lg  g  ab  r  h X2b X3b hr rbi sb cs bb so  ibb hbp sh sf gidp  
4   ansonca01 1871     1  RC1   25 120 29 39  11   3  0  16  6  2  2  1  NA  NA NA NA  NA  
44  forceda01 1871     1  WS3   32 162 45 45   9   4  0  29  8  0  4  0  NA  NA NA NA  NA  
68  mathebo01 1871     1  FW1   19  89 15 24   3   1  0  10  2  1  2  0  NA  NA NA NA  NA  
99  startjo01 1871     1  NY2   33 161 35 58   5   1  1  34  4  2  3  0  NA  NA NA NA  NA  
102 suttoez01 1871     1  CL1   29 128 35 45   3   7  3  23  3  1  1  0  NA  NA NA NA  NA  
106 whitede01 1871     1  CL1   29 146 40 47   6   5  1  21  2  2  4  1  NA  NA NA NA  NA
```

H = hits

BB = Bases on Balls (walks)

HBP = Times Hit by Pitch

AB = At bats

SF = Sacrifice Flies

Aggregações

$$OBP = \frac{H + BB + HBP}{AB + BB + HBP + SF}$$

- Preparação de dados

```
baseball$sf[baseball$year<1954] <- 0  
any(is.na(baseball$sf))  
  
baseball$hbp[is.na(baseball$hbp)] <- 0  
any(is.na(baseball$hbp))  
  
baseball <- baseball[baseball$ab >= 50,]
```

- Calculo de OBP para cada jogador / ano

```
baseball$OBP <- with(baseball, (h+bb+hbp) / (ab+bb+hbp+sf))  
tail(baseball)
```

Agregações com group by

■ Agregação com group by

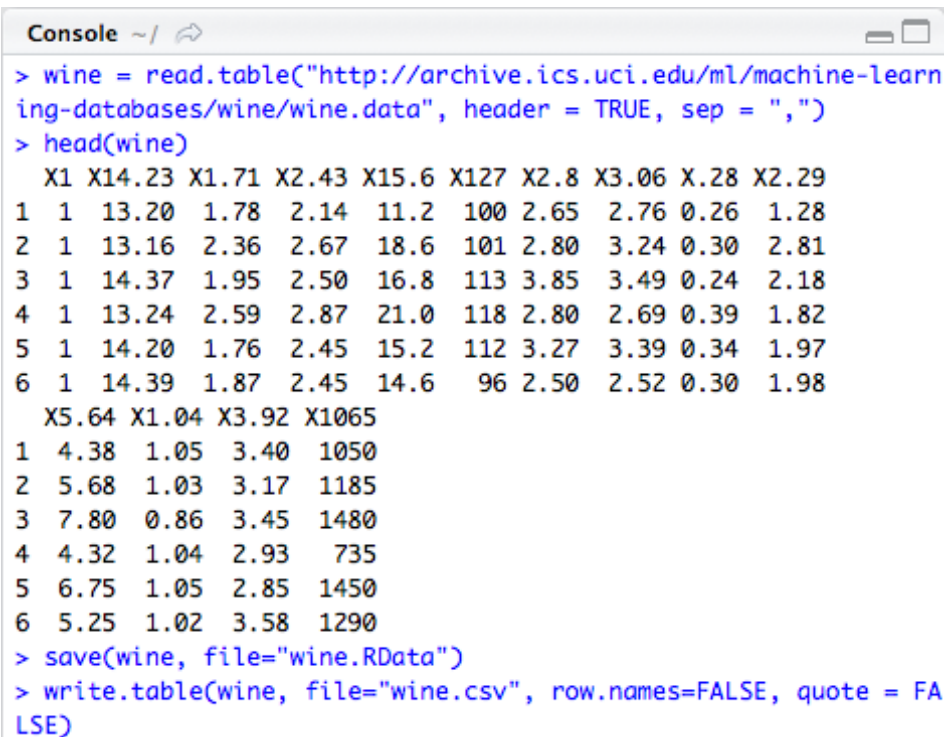
```
obp <- function(data) {  
  c(OBP = with(data, sum(h+bb+hbp) / sum(ab+bb+hbp+sf)))  
}
```

```
careerOBP <- ddply(baseball, .variables="id", .fun=obp)  
careerOBP <- careerOBP[order(careerOBP$OBP, decreasing = TRUE),]  
head(careerOBP, 10)
```

	id	OBP
1089	willite01	0.4816861
875	ruthba01	0.4742209
658	mcgrajo01	0.4657478
356	gehrilo01	0.4477848
85	bondsba01	0.4444622
476	hornsro01	0.4339068
184	cobbty01	0.4329655
327	foxxji01	0.4290509
953	speaktr01	0.4283386
191	collied01	0.4251246

Carregamento e gravação de arquivos

```
wine = read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data",
  header = TRUE, sep = ",")
head(wine)
save(wine, file="wine.RData")
rm(wine)
load("wine.RData")
write.table(wine, file="wine.csv", row.names=FALSE, quote = FALSE)
```

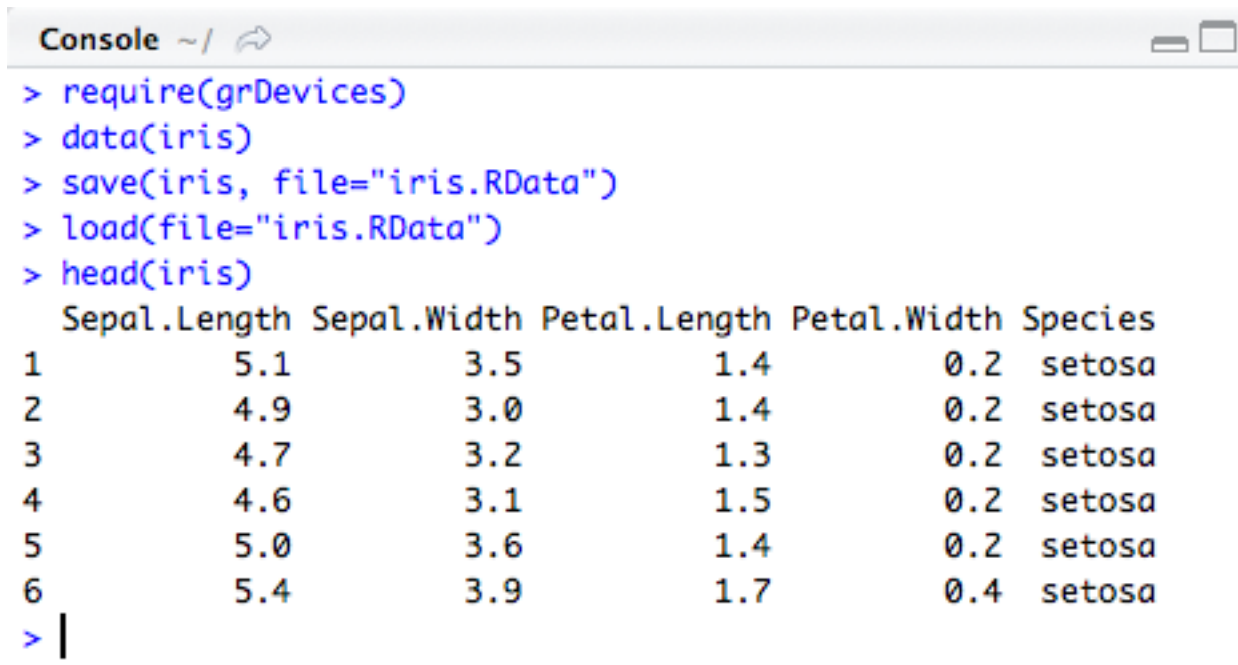


The screenshot shows a R console window with the following output:




```
> wine = read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data", header = TRUE, sep = ",")
> head(wine)
  X1 X14.23 X1.71 X2.43 X15.6 X127 X2.8 X3.06 X.28 X2.29
1  1  13.20  1.78  2.14  11.2  100  2.65  2.76  0.26  1.28
2  1  13.16  2.36  2.67  18.6  101  2.80  3.24  0.30  2.81
3  1  14.37  1.95  2.50  16.8  113  3.85  3.49  0.24  2.18
4  1  13.24  2.59  2.87  21.0  118  2.80  2.69  0.39  1.82
5  1  14.20  1.76  2.45  15.2  112  3.27  3.39  0.34  1.97
6  1  14.39  1.87  2.45  14.6   96  2.50  2.52  0.30  1.98
  X5.64 X1.04 X3.92 X1065
1  4.38  1.05  3.40  1050
2  5.68  1.03  3.17  1185
3  7.80  0.86  3.45  1480
4  4.32  1.04  2.93   735
5  6.75  1.05  2.85  1450
6  5.25  1.02  3.58  1290
> save(wine, file="wine.RData")
> write.table(wine, file="wine.csv", row.names=FALSE, quote = FALSE)
```


Alguns datasets estão disponíveis em pacotes do R

```
require(grDevices)
data(iris)
save(iris, file="iris.RData")
load(file="iris.RData")
head(iris)
```



The screenshot shows an R console window with the following content:

```
Console ~/   
```

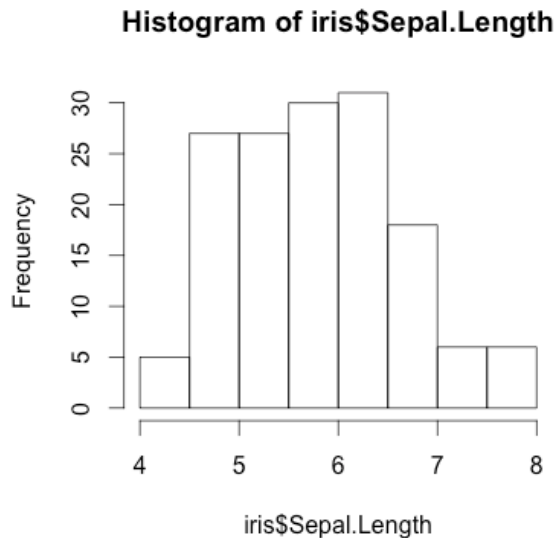
```
> require(grDevices)
> data(iris)
> save(iris, file="iris.RData")
> load(file="iris.RData")
> head(iris)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

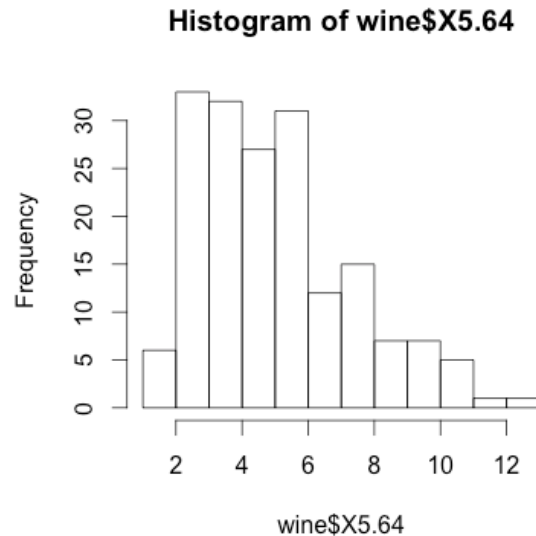
```
> |
```

Histograma

- Gráfico de frequências tabuladas por categorias
 - As categorias são comumente definidas como intervalos
 - As categorias (barras) devem ser adjacentes
- A área da barra denota o valor tabulado
 - Relevante quando as categorias não têm largura uniforme



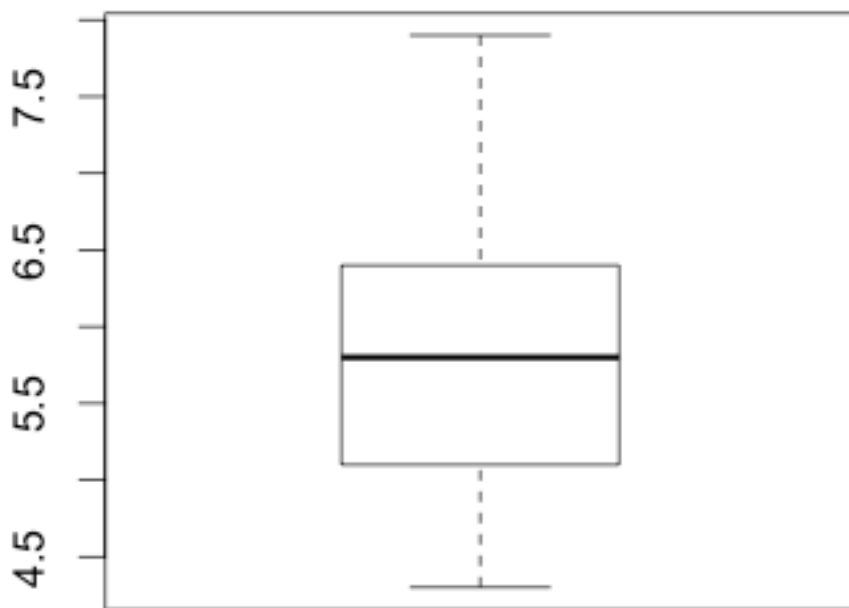
```
hist(iris$Sepal.Length)
```



```
hist(wine$X5.64)
```

Box-Plot

- Os dados são representados com uma caixa
 - As extremidades da caixa correspondem ao Q_1 e Q_3
 - A altura da caixa é $IQR = Q_3 - Q_1$
 - A mediana é marcada por uma linha dentro da caixa
 - Bigodes: duas linhas fora da caixa estendido para mínimo e máximo (dentro dos limites para outliers)



```
boxplot(iris$Sepal.Length)
```

Outlier

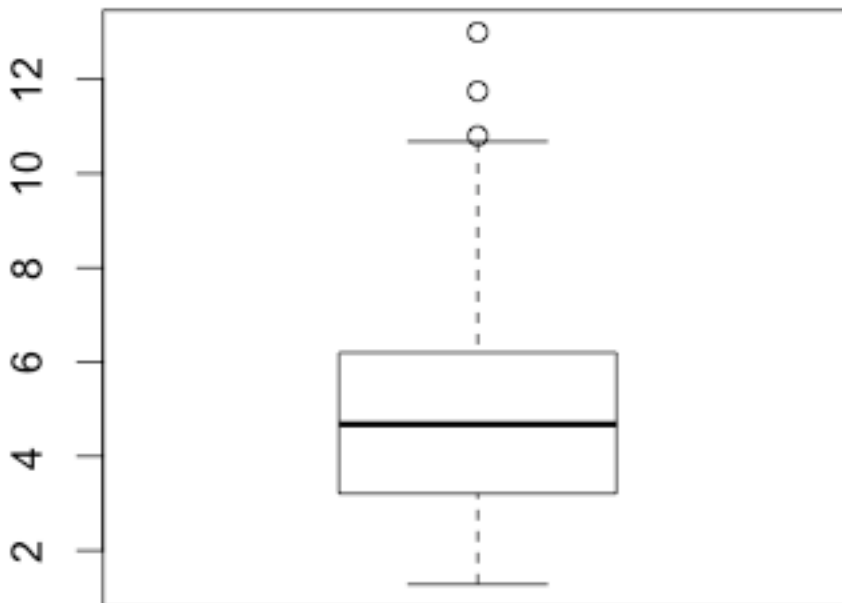
- Números fora do padrão

- Via Box-Plot:

$$outlier < Q_1 - 1.5 \cdot IQR \vee outlier > Q_3 + 1.5 \cdot IQR$$

- Via Teorema do Limite Central (distribuição normal):

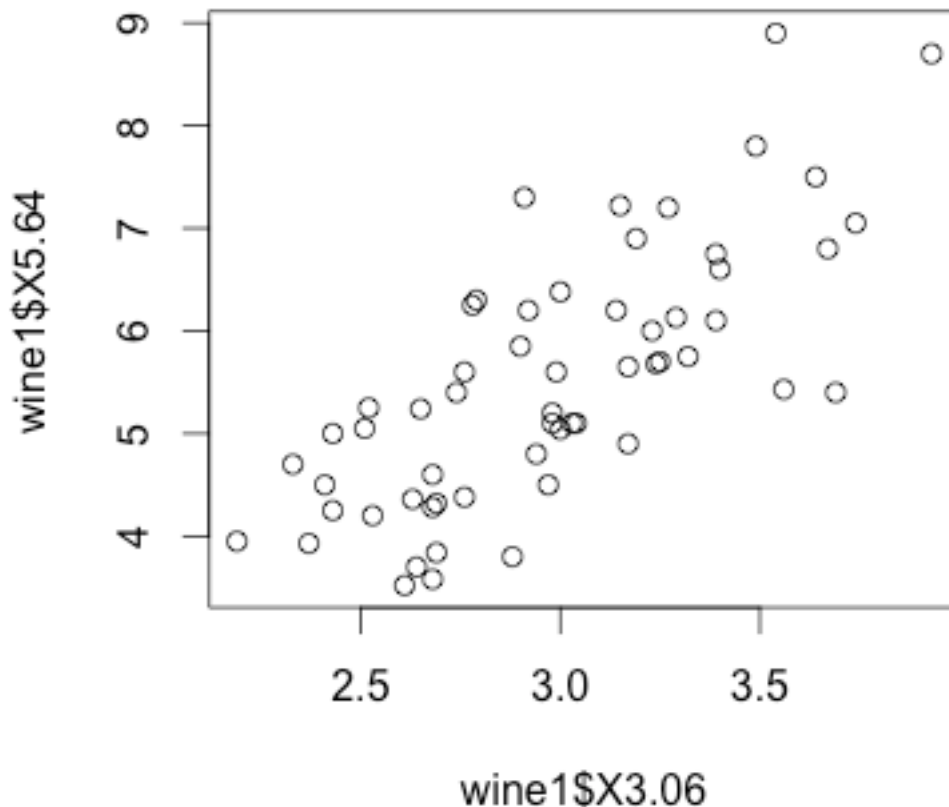
$$outlier < \bar{x} - 2.698 \cdot \sigma \vee outlier > \bar{x} + 2.698 \cdot \sigma$$



```
boxplot(wine$X5.64)
```

Gráfico de dispersão (scatter)

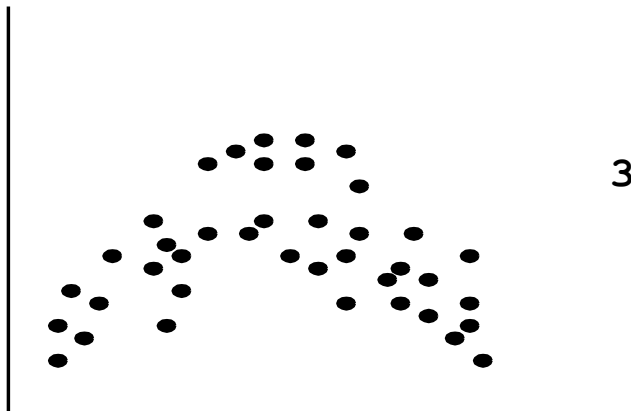
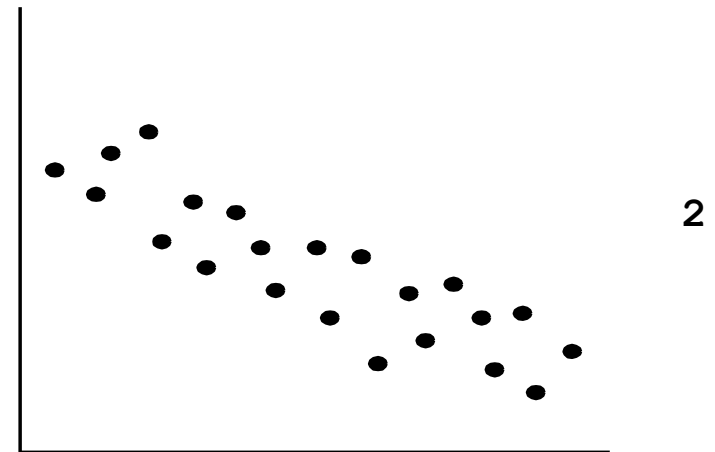
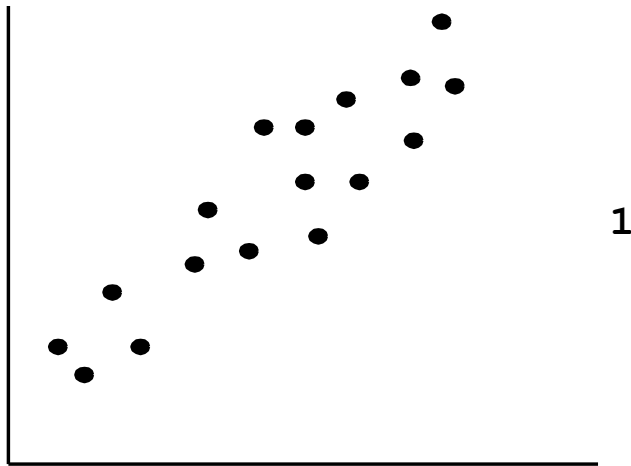
- Fornece um primeiro olhar para dados bivariados para ver grupos de pontos, valores atípicos, etc
- Cada par de valores é tratado como um par de coordenadas e representados como pontos no plano



```
wine1 = wine[wine$X1==1,]  
plot(wine1$X2.8 ~ wine1$X3.06)
```

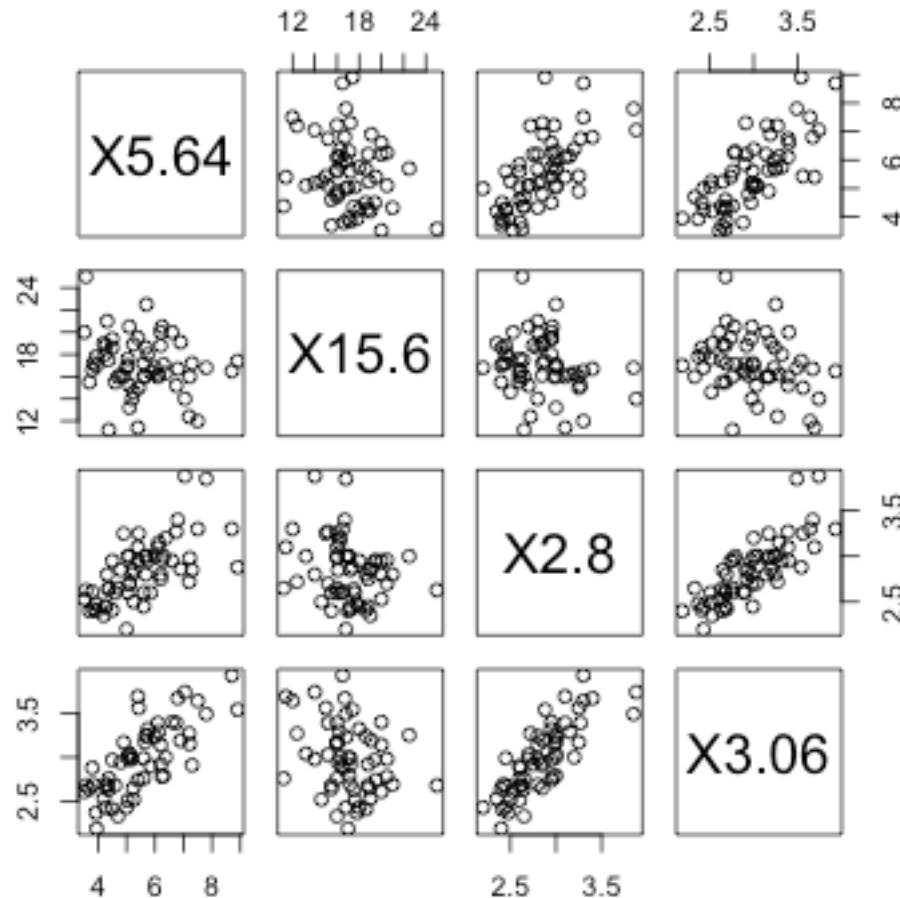
Dados correlacionados

- (1) correlação positiva e (2) correlação negativa
- (3) correlação não-linear e (4) sem correlação



Matrizes de gráficos de dispersão

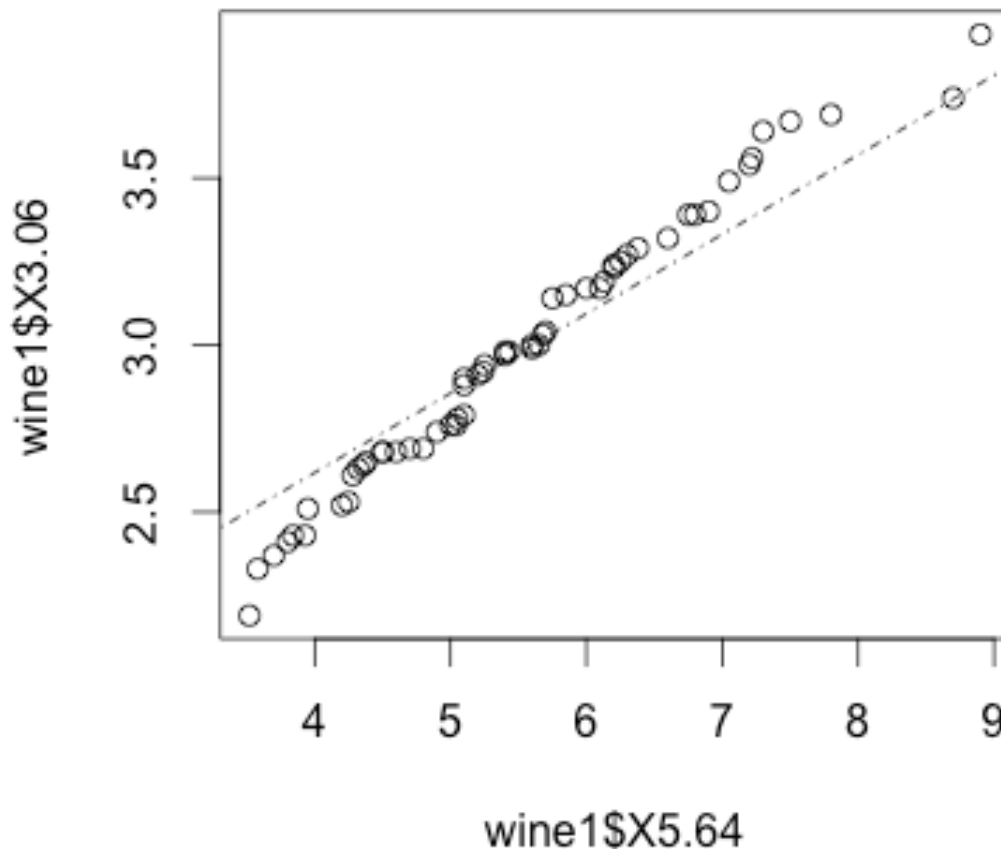
- Matriz de gráficos de dispersão (scatter diagrams) para combinações dois a dois entre k dimensões de dados



```
plot(~ X14.23 + X15.6 + X2.8 + X3.06, data=wine1)
```

Gráfico Quantile-Quantile (Q-Q)

- Avalia a distribuição de duas variáveis

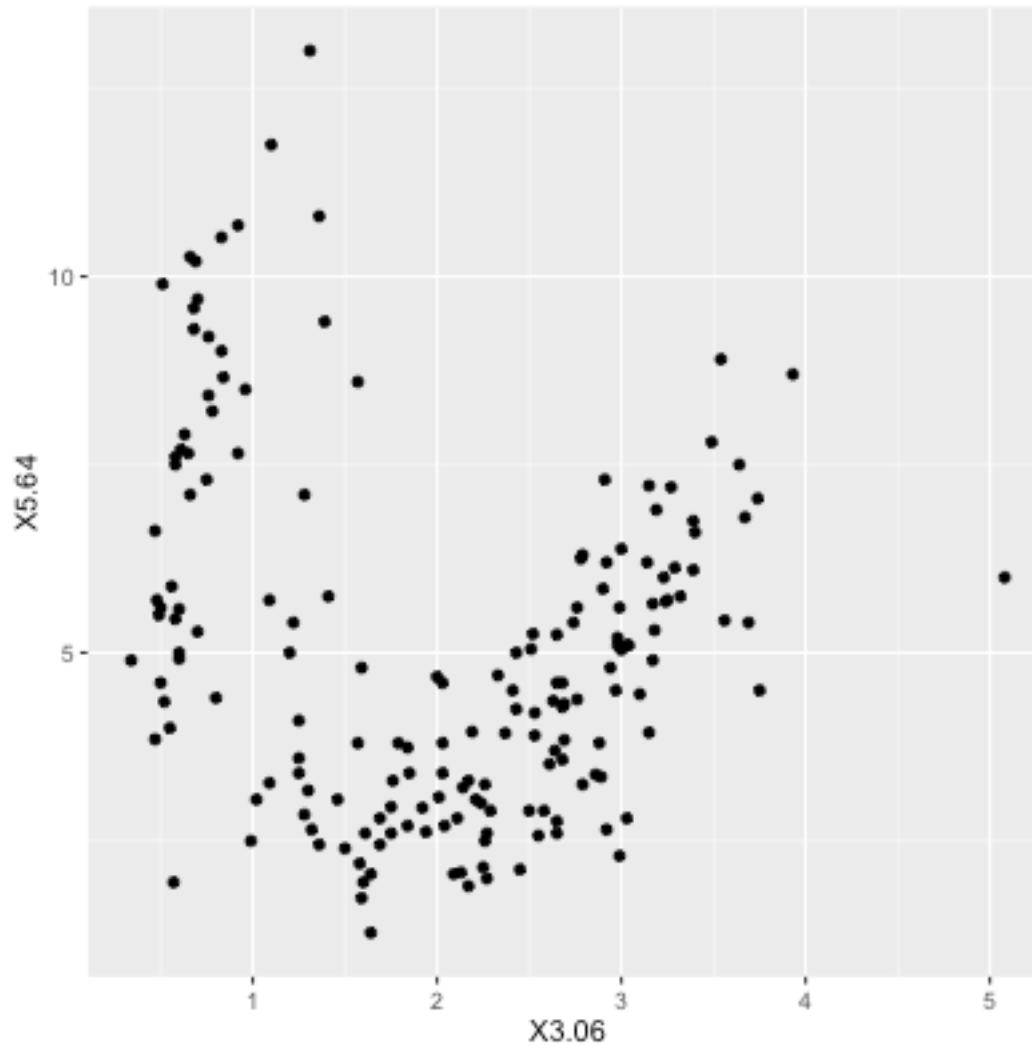


```
qqplot(wine1$X2.8,wine1$X3.06)  
fm <- lm(wine1$X3.06 ~ wine1$X2.8)  
abline(coef(fm), lty=4)
```


Exploração Visual de Dados

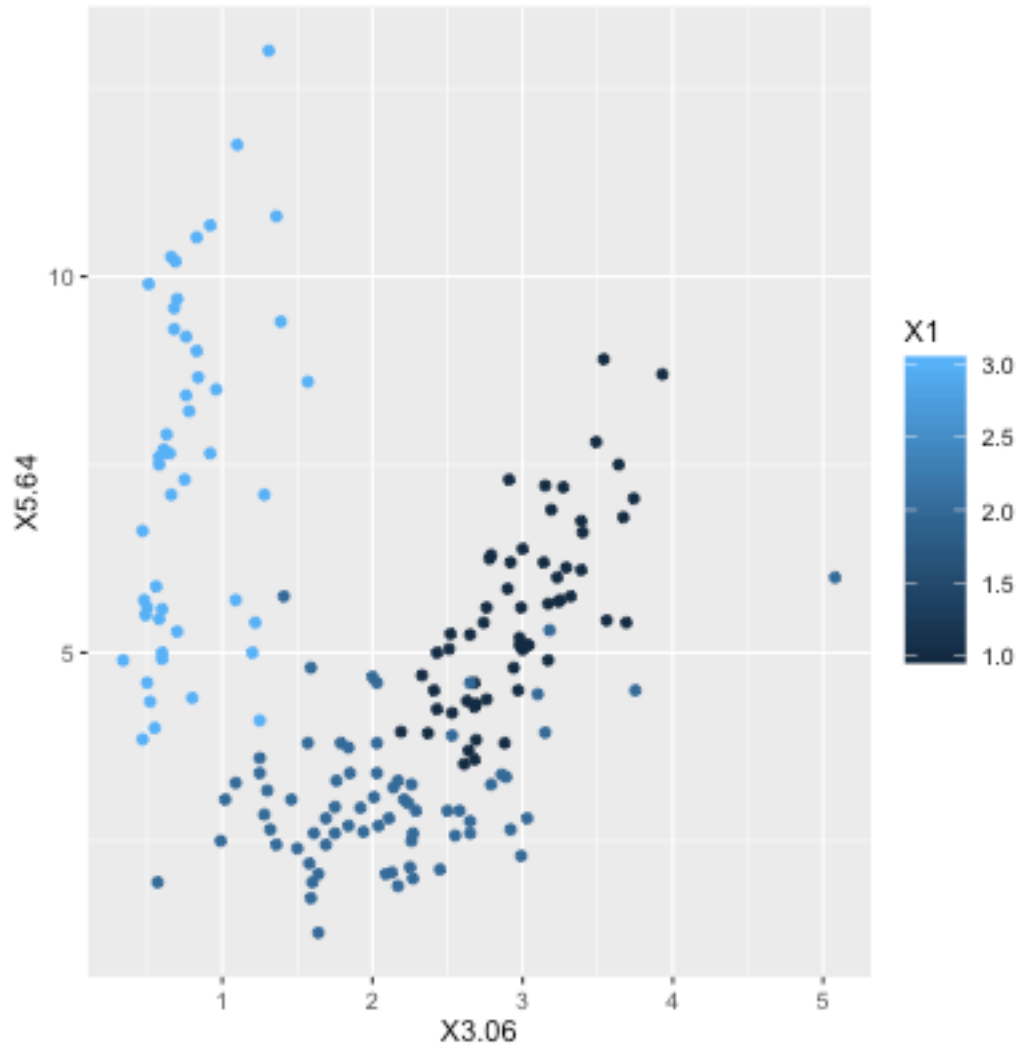
- Obtém uma visão de um espaço de informação por dados mapeamento sobre primitivas gráficas
- Fornecer uma visão geral qualitativa de grandes conjuntos de dados
- Busca de padrões, tendências, estrutura, irregularidades, as relações entre os dados
- Ajude a encontrar regiões interessantes e parâmetros adequados para posterior análise quantitativa
- Fornecer uma prova visual de representações de computador derivado

Gráficos avançados



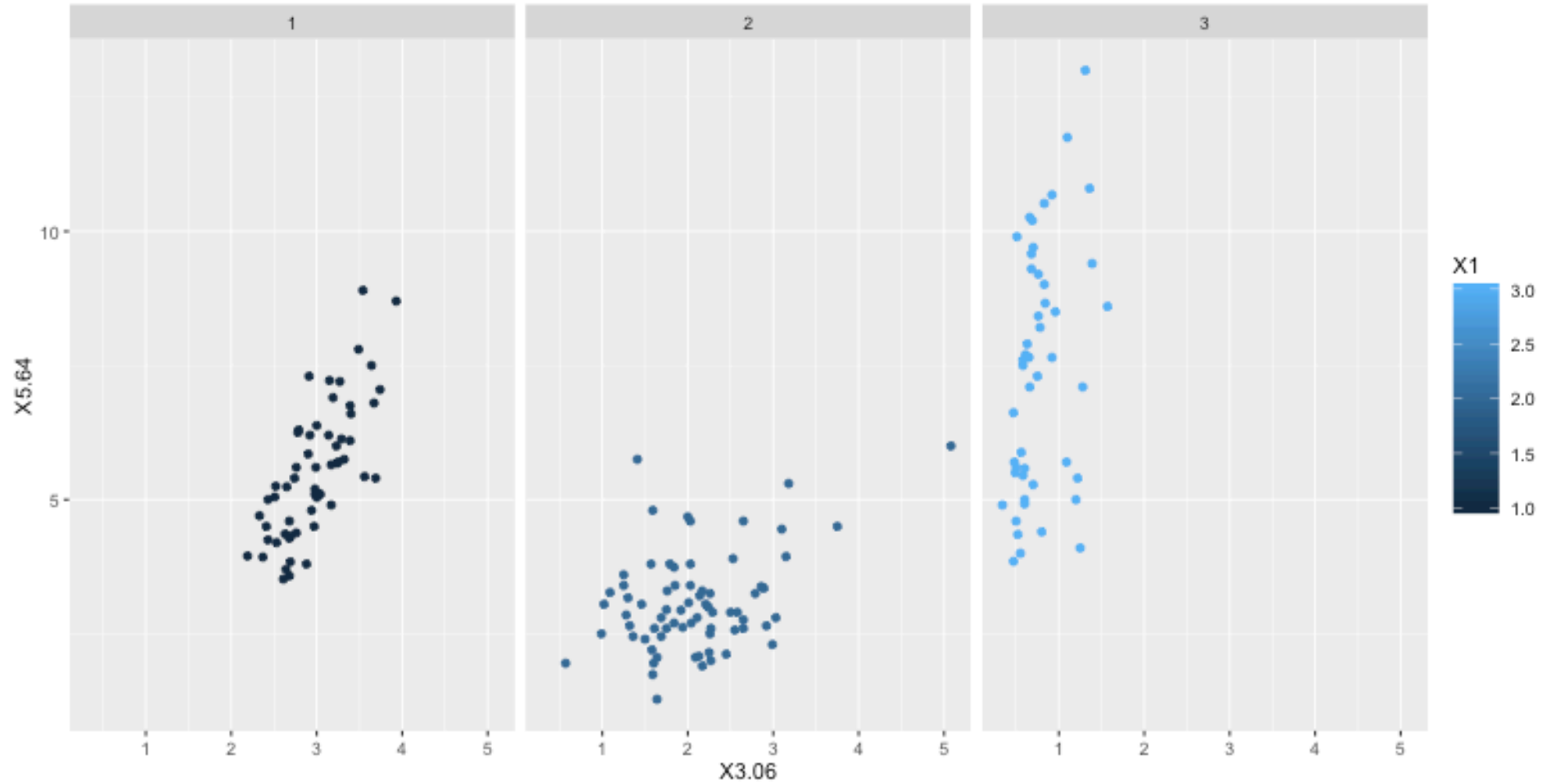
```
g = ggplot(wine, aes(x = X3.06, y = X2.8))  
g + geom_point()
```

Gráficos avançados



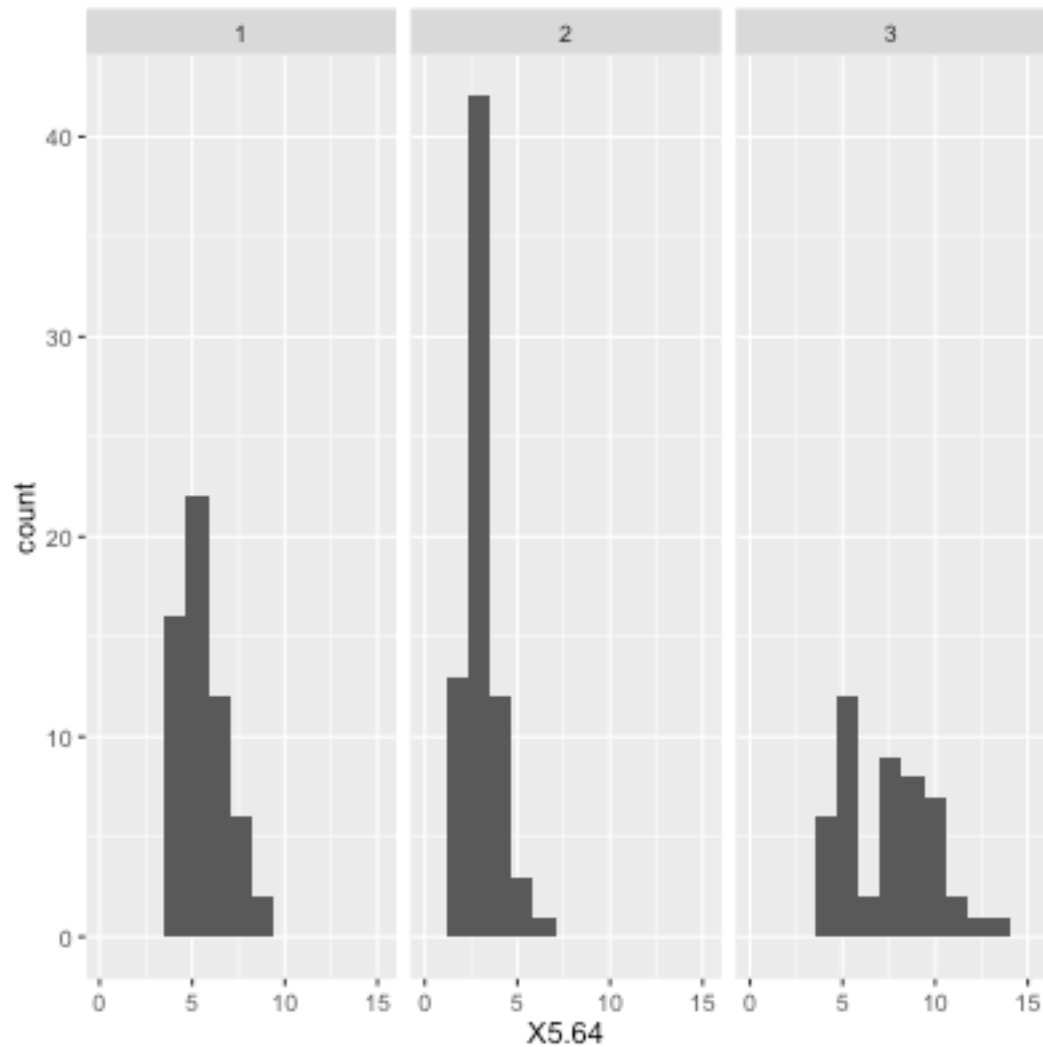
```
g + geom_point(aes(color = X1))
```

Gráficos avançados



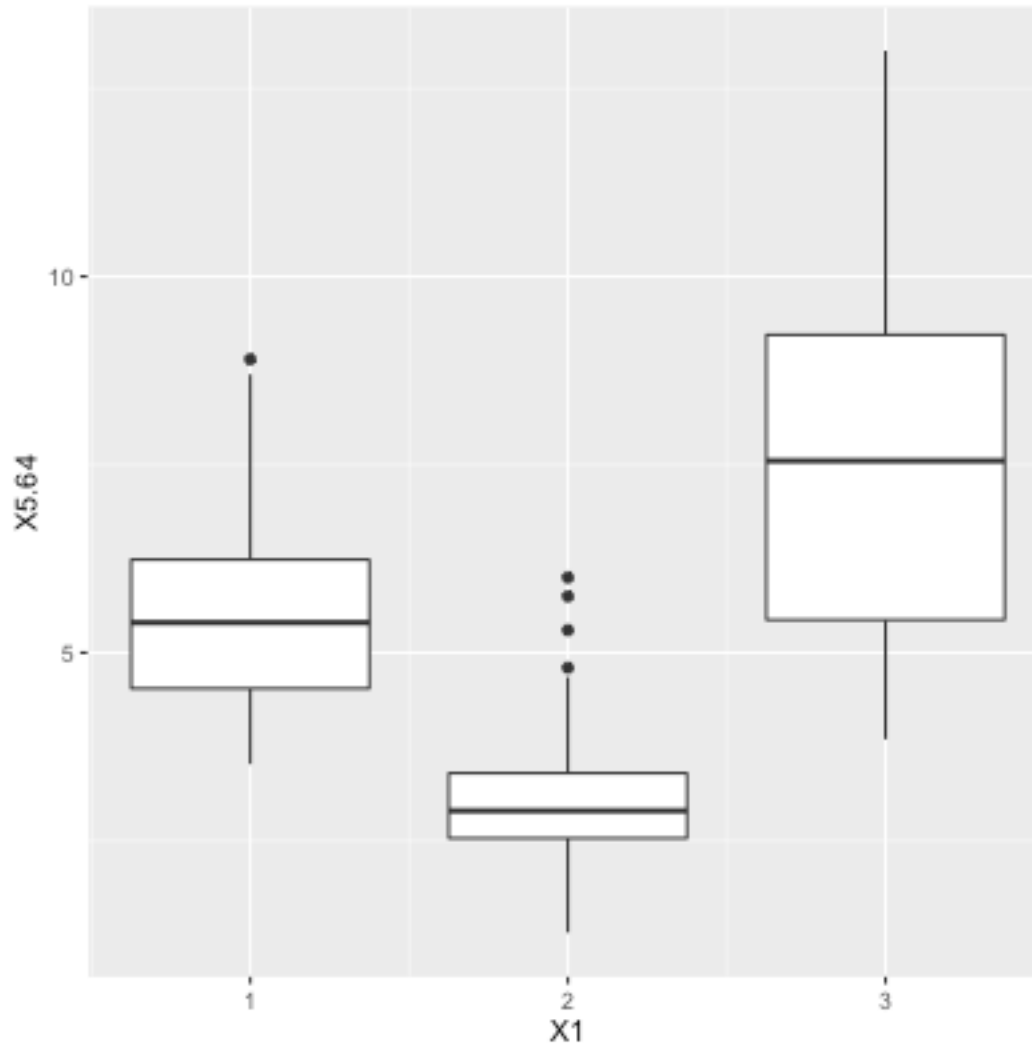
```
g + geom_point(aes(color = X1)) + facet_grid(~ X1)
```

Gráficos – histograma por fator



```
ggplot(wine, aes(x = X5.64)) +  
geom_histogram(bins=10) + facet_wrap(~X1)
```

Gráficos – Bloxplot por fator



```
ggplot(wine, aes(group = X1, x = X1, y = X5.64))  
+ geom_boxplot()
```

Referências

