Introdução ao R

André Luiz Lima Costa

“Hubris is the greatest danger that accompanies formal data analysis, including formalized statistical analysis. The feeling of “Give me (or more likely even, give my assistant) the data, and I will tell you what the real answer is!” is one we must all fight against again and again, and yet again.”

John Wilder Tukey

Introdução

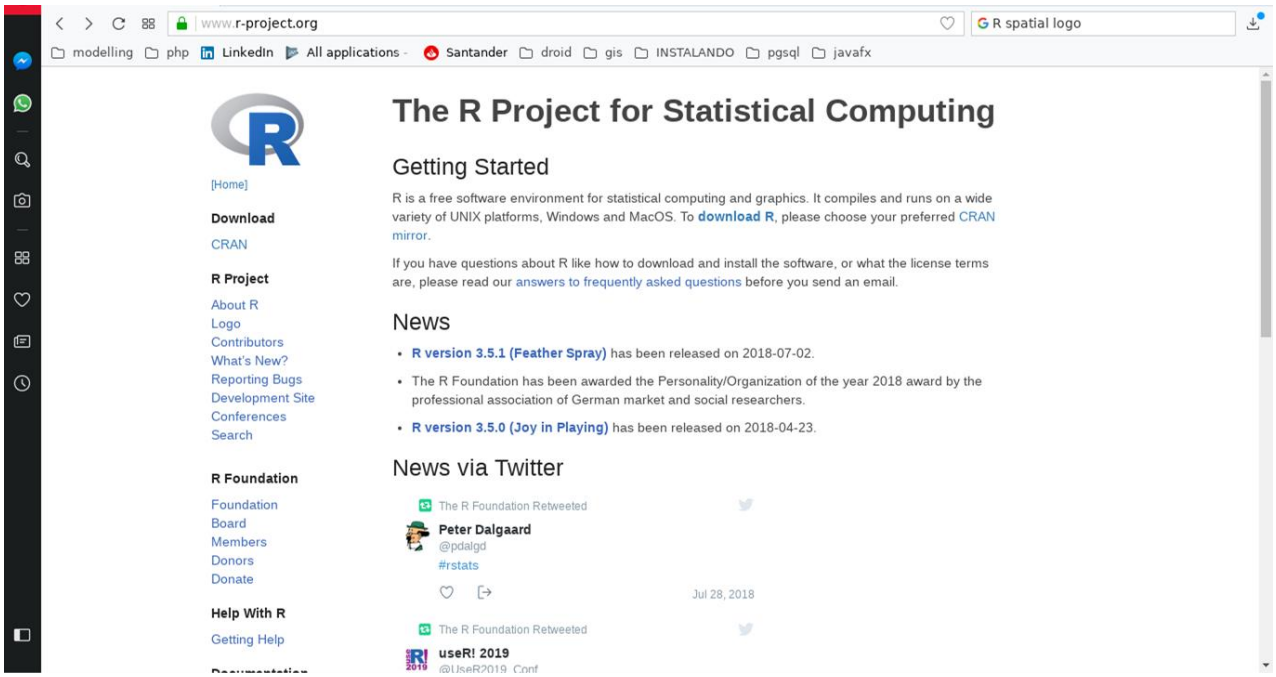
Esta apostila aspectos básicos da linguagem R e estatística usando R, como usar e como funciona. Ela dará uma base para seu aprendizado e foi baseada nos sites rspatial.org e r-tutor.com que oferecem de forma concisa uma introdução à linguagem R e estatística.

VISITE:

https://gdatasystems.com

Instalando R

R pode ser instalado facilmente em qualquer sistema operacional através do site da CRAN no www.r-project.org, basta escolher o 'mirror-site' e fazer o download para o seu sistema operacional.



Para iniciar digite R no 'prompt' (aqui representado por: ***[voce@aqui ~]$)*** da sua linha de comando e o seguinte aparecerá. Estamos prontos para começar!

```
[voce@aqui ~]$ R
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (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.
```

Prontos para começar!

Fortaleza, 26 de Junho 2020

PARTE 1 – A Linguagem R

1.1 – Tipos básicos

Vamos iniciar aprendendo os mais primitivos tipos de dados e funções auxiliares para cada tipo.

1.1.1 - Tipo numeric

Para criar nossa primeira variável digite:

```
> a <- 1
```

A seta <- foi usada para atribuir o valor 1 à variável 'a', poderíamos usar '=' mas, por convenção, o operador = é usado para assinalar um valor dentro de uma função (veremos mais a frente). O nome 'a' para a variável é arbitrário e poderíamos ter usado b, x, w, raiz, ou qualquer outro nome que nos ajudaria a identificar ele. Existem algumas restrições para o nome de variáveis, elas não podem iniciar com números, não podem ter espaço e nem caracteres especiais tipo '*' mas podem conter" ." .

Para visualizar o valor atribuído a 'a' podemos usar as funções show ou print:

```
>  show(a)
[1] 1
>  print(a)
[1] 1
```

Ou podemos simplesmente digitar o nome da variável:

```
> a
[1] 1
```

IMPORTANTE: Em R todos os tipos básicos são armazenados como um vetor, que é uma array unidimensional com n valores de um certo tipo, mesmo um simples número ou caractere é um vetor.

Podemos usar a função class para checar que tipo é determinado vetor:

```
> class(a)
[1] "numeric"
```

'numeric' significa que 'a' é um número real (decimal). Seu valor é equivalente a 1,000 mas os zeros não são mostrados por padrão. Em alguns casos, criar uma variável do tipo número inteiro (integer) pode ser ser necessário e o fazemos usando uma das duas formas abaixo:

```
> a<- as.integer(9)
> class(a)
[1] "integer"
> a<- 9L
> class(a)
[1] "integer"
```

Para criarmos um vetor com vários valores usamos a função c que combina os valores:

```
> b <- c(3.12, 2.09, 5)
> b
[1] 3.12 2.09 5.00
```

Para criarmos uma sequência regular é mais fácil usar o operador ‘:’ :

```
> s<- 1001:1007
> s
[1] 1001 1002 1003 1004 1005 1006 1007
```

Na ordem reversa:

```
> rs<- 1007:1001
> rs
[1] 1007 1006 1005 1004 1003 1002 1001
```

Podemos criar sequências mais elaboradas usando também a função seq e reverter usando a função rev:

```
> s<- seq(from=6, to=0, by=-0.5)
> s
[1] 6.0 5.5 5.0 4.5 4.0 3.5 3.0 2.5 2.0 1.5 1.0 0.5 0.0
> r<- rev(s)
> r
[1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0
```

Ou repetir o mesmo valor usando rep:

```
> bis<- rep(8,times=4)
> bis
[1] 8 8 8 8
> bis2<- rep(1:2,times=5)
> bis2
[1] 1 2 1 2 1 2 1 2 1 2
> bis3<- rep(1:2, each=5)
> bis3
[1] 1 1 1 1 1 2 2 2 2 2
> bis4<- rep(1:2, each=5,times=3)
> bis4
[1] 1 1 1 1 1 2 2 2 2 2 1 1 1 1 1 2 2 2 2 2 1 1 1 1 1 2 2 2 2 2
```

1.1.2 - Tipo character

O tipo ‘character’ é usado para representar palavras. Também são chamados de ‘string’:

```
> str<-'Altitude Z'
> str
[1] "Altitude Z"
> str<-"Valor 1"
> str
[1] "Valor 1"
```

Usando com função class:

```
> class(str)
[1] "character"
```

Podemos usar “ ou ‘ mas nunca misturar os dois assim: **str<-“Valor’** .
Criando um vetor de strings:

```
> furos<- c('AT-001','AT-002','AT-003','AT-004')
> furos
[1] "AT-001" "AT-002" "AT-003" "AT-004"
```

A função length informa quantos elementos estão presentes no vetor:

```
> length(furos)
[1] 4
```

A função nchar nos informa o número de caracteres de cada elemento do vetor:

```
> nchar(furos)
[1] 6 6 6 6
```

A função paste une ou concatena strings usando o separador padrão que é o espaço:

```
> lat<- "latitude"
> lon<- "longitude"
> paste(lat,"x",lon)
[1] "latitude x longitude"
```

Podemos alterar o separador usando o argumento sep:

```
> paste(lat,"x",lon,sep='-')
[1] "latitude-x-longitude"
```

Usando o argumento collapse podemos concatenar todos os elementos de um vetor string:

```
> paste(furos,collapse=' / ')
[1] "AT-001 / AT-002 / AT-003 / AT-004"
```

A função substr retorna o intervalo dentro de uma string:

```
> substr("Alvo: Zinco2",1,5)
[1] "Alvo:"
> substr("Alvo: Zinco2",7,12)
[1] "Zinco2"
```

A função sub substitui parte de uma string com outra string:

```
> sub('Zinco','Chumbo',"Alvo: Zinco2")
[1] "Alvo: Chumbo2"
```

A função grep retorna o índice da string num vetor que contém o argumento fornecido:

```
> sondas<- c('LF-230', 'Cs-4002','Cs-4000')
> grep('Cs',sondas)
[1] 2 3
> grep('2',sondas)
[1] 1 2
```

Podemos usar caracteres ‘coringas’ com grep , $ significa termina e ^ significa começa;

```
> grep('2$',sondas)
[1] 2
> grep('^L',sondas)
[1] 1
```

Para ter acesso a cada elemento usamos [n] n representa o número índice e começa com 1:

```
> sondas[1]
[1] "LF-230"
> sondas[1:3]
[1] "LF-230"  "Cs-4002" "Cs-4000"
> sondas[grep('2',sondas)]
[1] "LF-230"  "Cs-4002"
```

1.1.3 - Tipo logical

O tipo lógico representa TRUE ou FALSE (verdadeiro ou falso). São muito utilizados em R e em programação em geral:

```
> verdade <- TRUE
> verdade
[1] TRUE
> class(verdade)
[1] “logical”
```

Valores lógicos são frequentemente o resultado de computação. Por exemplo:

```
> x<- 30
> x<20
[1] FALSE
> x<40
[1] TRUE
```

Valores lógicos podem ser tratados como números TRUE é 1 e FALSE é 0:

```
> v<- 2
> v+TRUE
[1] 3
> v+FALSE
[1] 2
> v*FALSE
[1] 0
> as.logical(1)
[1] TRUE
> as.logical(0)
[1] FALSE
```

1.1.4 - factor

Um fator é uma variável categorizadora com um conjunto de valores chamados levels (níveis). Eles podem ser criados usando a função as.factor. Em R você tipicamente precisa converter uma variável string para um fator para identificar os grupos em modelos e testes estatísticos:

```
> comp<- c('ore','ore','waste','ore','waste','waste','ore','not_defined')
> cat<- as.factor(comp)
> cat
[1] ore         ore         waste       ore         waste       waste
[7] ore         not_defined
Levels: not_defined ore waste
```

Mas números também podem ser usados.

```
> se<- c(1:3,1:5,2:4)
> sec<- as.factor(se)
> sec
 [1] 1 2 3 1 2 3 4 5 2 3 4
Levels: 1 2 3 4 5
```

Vamos agora mover para tipos mais complexos no capítulo seguinte. Pratique e experimente o que vimos neste capítulo.

1.2 – Estrutura de dados básicos

Vimos até agora os tipos básicos de dados em R, números, strings, lógico e fator. Todos estes armazenados em vetor. Agora veremos estruturas adicionais de dados que armazenam dados básicos: Matrix, array, data.frame e list.

1.2.1 - Matrix

O vetor é uma array unidimensional. Uma array bidimensional é representada como uma Matrix. Veja como podemos criar uma matrix com duas linhas e três colunas:

```
> m <- matrix(ncol=3,nrow=2)
> m
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
> m<- matrix(1:6,ncol=3,nrow=2)
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

IMPORTANTE Uma matrix pode ser criada usando as funções cbind (unir colunas) e rbind (unir linhas). Elas são extensivamente usadas em R e peço sua dedicação em entender essas funções.

```
> a<- 1:3
> b<- 4:6
> a
[1] 1 2 3
> b
[1] 4 5 6
```

Usando cbind teremos:

```
> m1<- cbind(a,b)
> m1
     a b
[1,] 1 4
[2,] 2 5
[3,] 3 6
```

Usando rbind teremos:

```
> m2<- rbind(a,b)
> m2
  [,1] [,2] [,3]
a    1    2    3
b    4    5    6
```

Podemos usar cbind e rbind para combinar matrizes, desde que o número de colunas ou linhas dos dois objetos sejam o mesmo:

```
> m3<- cbind(b,b,a)
> m<- cbind(m1,m3)
> m
     a b b b a
[1,] 1 4 4 4 1
```

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

Podemos acessar informações da estrutura da matrix com as funções nrow, ncol, dim e length:

```
> nrow(m)
[1] 3
> ncol(m)
[1] 5
> dim(m)
[1] 3 5
> length(m)
[1] 15
```

Colunas possuem nomes que podem ser mudados:

```
> colnames(m)
[1] "a" "b" "b" "b" "a"
> colnames(m)<- c('ID','X','Y','Z','V1')
> m
     ID X Y Z V1
[1,]  1 4 4 4  1
[2,]  2 5 5 5  2
[3,]  3 6 6 6  3
```

Da mesma forma as linhas:

```
> rownames(m)<- paste0('linha_',1:nrow(m))
> m
        ID X Y Z V1
linha_1  1 4 4 4  1
linha_2  2 5 5 5  2
linha_3  3 6 6 6  3
```

Uma matrix, assim como um vetor, só pode armazenar um único tipo de dado. Se tentarmos misturar diferentes tipos os mesmos serão convertidos para a categoria que valida todos os dados (muito provavelmente do tipo character):

```
> x<- c(1,2,3)
> x
[1] 1 2 3
> x<- c(x,'a','b')
> x
[1] "1" "2" "3" "a" "b"
> cbind(c('a','b'),1:2)
     [,1] [,2]
[1,] "a"  "1"
[2,] "b"  "2"
```

1.2.2 - Array

Um vetor é uma array unidimensional, uma matriz é uma array bidimensional, acima disso (3 dimensões em diante), denominamos simplesmente array.

Vamos criar agora uma array tridimensional onde colunas são grid na direção x, linhas são grid na direção y e cada array é um nível vertical. Os valores serão digamos teor (como um bloco diagrama 20X20X20, cada elemento seria o valor do teor no bloco):

```
> a1<- c(3,2,0,2,2,1,3,2,1)
```

```
> a2<- c(3,3,1,2,2,2,3,3,2)
> a3<- c(4,3,2,2,2,2,4,3,2)
> coluna.nomes<- c(300,340,360)
> linha.nomes<- c(20,40,60)
> matriz.nomes<- c(0,-20,-40)
> r<- array(c(a1,a2,a3),dim=c(3,3,3),dimnames=list(linha.nomes,coluna.nomes,
matriz.nomes))
> r
, , 0
   300 340 360
20   3   2   3
40   2   2   2
60   0   1   1

, , -20
   300 340 360
20   3   2   3
40   3   2   3
60   1   2   2

, , -40
   300 340 360
20   4   2   4
40   3   2   3
60   2   2   2
```

1.2.3 - list

Uma lista é uma estrutura muito flexível para armazenar dados. Cada elemento de uma lista pode conter qualquer tipo de objeto R (vetor, matrix, array, data.frame, outra lista, etc).

Uma lista simples que mostra que o primeiro elemento da lista [[1]] é o vetor 1 2 3:

```
> list(1:3)
[[1]]
[1] 1 2 3
```

Agora uma lista com dois tipos de dados, string e vetor de números:

```
> l<- list('xyz',c(100,200))
> l
[[1]]
[1] "xyz"

[[2]]
[1] 100 200
```

E uma lista mais complexa contendo uma lista, uma matrix e dois vetores:

```
> m<- matrix(1:6,ncol=3,nrow=2)
> ll<- list(l,m,”abc”,c(0.6,0.5,0.9))
> ll
[[1]]
[[1]][[1]]
[1] “xyz”
[[1]][[2]]
```

```
[1] 100 200
[[2]]
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
[[3]]
[1] “abc”
[[4]]
[1] 0.6 0.5 0.9
```

1.2.4 - data.frame

O formato de dado estruturado data.frame é a força da análise estatística de dados em R. Ele é bidimensional mas pode ter colunas de diferente tipos. Um data.frame é o que você obtêm quando importa uma planilha com funções do tipo read.table ou read.csv (que veremos mais adiante).

Podemos também montar um data.frame usando código:

```
> furo<- c('AT-001','AT-002','AT-002','AT-003')
> intervalo<- c(0.1,2.9,3.7,3.6)
> tipo<- c('waste','ore','ore','ore')
> teor<- c(0.04,0.9,1.4,1.4)
> df<- data.frame(furo,intervalo,tipo,teor)
> df
    furo intervalo  tipo teor
1 AT-001       0.1 waste 0.04
2 AT-002       2.9   ore 0.90
3 AT-002       3.7   ore 1.40
4 AT-003       3.6   ore 1.40
```

Funções usadas com data.frame

```
> is.list(df)
[1] TRUE
> names(df)
[1] "furo"      "intervalo" "tipo"      "teor"
> nrow(df)
[1] 4
> ncol(df)
[1] 4
> dim(df)
[1] 4 4
> colnames(df)
[1] "furo"      "intervalo" "tipo"      "teor"
```

Agora que cobrimos os tipos básicos de formatos e estrutura de dados, vamos aprofundar um pouco mais no aprendizado de R. Continue praticando.

1.3 – Índices

Existem várias maneiras de acessar ou substituir valores em vetores ou outras estruturas de dados. A mais comum é usando ‘indexing’ (índice). Também chamado de ‘slicing’ (fatiamento).

1.3.1 - Índice em vetores

Como visto no capítulo 1-1 brevemente, usamos [] para envolver o índice.

Abaixo mostramos alguns exemplos de ‘indexing’

```
> sb<- 1:10
> sb
 [1]  1  2  3  4  5  6  7  8  9 10
> sb[1]
[1] 1
> sb[10]
[1] 10
> sb[4:8]
[1] 4 5 6 7 8
```

Neste outro exemplo excluímos do resultado o elemento de índice 5.

```
> sb[-5]
[1]  1  2  3  4  6  7  8  9 10
```

Podemos também usar índice para modificar valores:

```
> sb[5]<- 64
> sb
 [1]  1  2  3  4 64  6  7  8  9 10
> sb[3:4]<- 999
> sb
 [1]   1   2 999 999  64   6   7   8   9  10
> sb[-1]<- 0
> sb
 [1] 1 0 0 0 0 0 0 0 0 0
```

Uma característica de vetores em R é que vetores menores são reusados quantas vezes forem necessárias para satisfazer o preenchimento do vetor destino:

```
> sb[2:10]<- c(1,2,3)
> sb
 [1] 1 1 2 3 1 2 3 1 2 3
> sb[2:10]<- c(1,2)
Warning message:
In sb[2:10] <-  c(1, 2) :
  número de itens para para substituir não é um múltiplo do comprimento do
substituto
> sb
 [1] 1 1 2 1 2 1 2 1 2 1
```

Note o aviso que o número usado para preencher não é múltiplo dos valores a serem substituídos no segundo caso, mas a substituição é efetuada assim mesmo. Esse comportamento, caso não desejado pode levar a erros na frente.

1.3.2 - Índice em matrix

Como em vetores, elementos de uma matriz podem ser acessados usando índices retornando matrizes ou vetores.

Vamos criar a seguinte matriz 3x3 como exemplo:

```
> m<- matrix(1:9,nrow=3,ncol=3, byrow=TRUE)
> m
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

Existem duas maneiras para acessar um elemento da matriz: usando índice duplo ou simples. O índice duplo nos dá a posição (linha, coluna) e o simples nos dá o enésimo elemento de índice n:

```
> m[2,2]
[1] 5
> m[5]
[1] 5
```

Podemos também obter os seguintes resultados usando índices:

```
> m[1:2,1:2]
     [,1] [,2]
[1,]    1    2
[2,]    4    5
> m[2,]
[1] 4 5 6
> m[,2]
[1] 2 5 8
```

Podemos também acessar colunas pelo seu nome definido conforme abaixo:

```
> colnames(m)<- c('a','b','c')
> m
     a b c
[1,] 1 2 3
[2,] 4 5 6
[3,] 7 8 9
> m[,'b']
[1] 2 5 8
> m[,c('a','c')]
     a c
[1,] 1 3
[2,] 4 6
[3,] 7 9
```

Note que quando acessamos uma coluna ela se transforma automaticamente em um vetor, para manter o formato de coluna podemos usar:

```
> m[,'b',drop=FALSE]
     b
[1,] 2
[2,] 5
[3,] 8
```

Assinalar valores a elementos de uma matrix pode ser feito usando o índice, veja abaixo:

```
> m[1,1]<-  5
> m
     a b c
[1,] 5 2 3
[2,] 4 5 6
[3,] 7 8 9

> m[3,]<-  10
> m
      a  b  c
[1,]  5  2  3
[2,]  4  5  6
[3,] 10 10 10

> m[,2:3] <- 3:1
> m
      a b c
[1,]  5 3 3
[2,]  4 2 2
[3,] 10 1 1
```

O valor da diagonal de uma matrix pode ser assinalado usando a função diag:

```
> diag(m)
[1] 5 2 1
> diag(m)<- 0
> m
      a b c
[1,]  0 3 3
[2,]  4 0 2
[3,] 10 1 0
```

1.3.3 - Índice em list

Índice em lista pode ser um pouco confuso uma vez que você pode se referir a elementos de uma lista ou a elementos de um dado de uma lista (talvez uma matriz). Note a diferença do colchete duplo e do colchete simples.

Ao criarmos a lista abaixo acessamos o terceiro elemento da lista com li[3] mas acessamos a matriz com li[[3]] ou seja, a estrutura da lista é desfeita retornando o elemento na sua forma original:

```
> m<- matrix(1:9,nrow=3,ncol=3,byrow=TRUE)
> colnames(m)<- c('a','b','c')
> li<- list(list(1:3),c('x','y','z','w'),m)
> li[3]
[[1]]
     a b c
[1,] 1 2 3
[2,] 4 5 6
[3,] 7 8 9

> li[[3]]
     a b c
[1,] 1 2 3
[2,] 4 5 6
[3,] 7 8 9
```

Os elementos de uma lista podem ter nomes:

```
> names(li)<- c('lista','vetor','matriz')
> li
$lista
$lista[[1]]
[1] 1 2 3

$vetor
[1] "x" "y" "z" "w"

$matriz
     a b c
[1,] 1 2 3
[2,] 4 5 6
[3,] 7 8 9
```

E seus elementos podem ser extraídos usando esses nomes:

```
> li$vetor
[1] "x" "y" "z" "w"
> li[['matriz']]
     a b c
[1,] 1 2 3
[2,] 4 5 6
[3,] 7 8 9
```

O operador $ também pode ser usado com data.frame (que um tipo especial de list) mas não com matriz.

1.3.4 - Índices em data.frame

Índices em um data.frame podem ser usados como feito em matriz ou lista. Vamos primeiro criar o data.frame a partir da matriz m:

```
> df<- data.frame(m)
> class(df)
[1] "data.frame"
> df
  a b c
1 1 2 3
2 4 5 6
3 7 8 9
```

Note abaixo a diferença quando usamos o índice 1 com colchete duplo ou simples:

```
> df[[1]]
[1] 1 4 7
> df[1]
  a
1 1
2 4
3 7
> df[,1]
[1] 1 4 7
```

```
> df[1,]
  a b c
1 1 2 3
```

1.3.5 - Which, %in% e match

Algumas vezes você não sabe o índice dos valores que você precisa e estas funções e operadores o ajudaram a encontrar os índices.

Vejamos como usar which e vetor lógico (TRUE é retornado):

```
> x <- 10:20
> i <- which(x > 15)
> x
 [1] 10 11 12 13 14 15 16 17 18 19 20
> i
[1]  7  8  9 10 11
> x[i]
[1] 16 17 18 19 20

> b <- x>15
> x
 [1] 10 11 12 13 14 15 16 17 18 19 20
> b
 [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
> x[b]
[1] 16 17 18 19 20
```

Um operador bem útil é o %in% que nos permite perguntar qual valor em um vetor está presente em outro.

Vejamos como:

```
> x<- 10:20
> j<- c(1,3,11,19)
> j %in% x
[1] FALSE FALSE  TRUE  TRUE
> which(j %in% x)
[1] 3 4
```

A função match também nos permite comparar dois vetores e saber os índices dos valores coincidentes.

Note que match(j,x) retorna resultado diferente de match(x,j):

```
> match(j,x)
 [1] NA NA  2 10
> match(x,j)
 [1] NA  3 NA NA NA NA NA NA NA  4 NA
```

Os resultados são:
- o elemento 3 e 4 de j são os mesmos que os elementos 2 e 10 de x.
- o elemento 2 e 10 de x são os mesmos que os elementos 3 e 4 de j.

Cobrimos os principais tipos de dados necessários para esta Apostila. Vamos agora ver como usar eles em R.

1.4 – Álgebra

Vetores e matrizes podem ser usados para computar novos vetores (e matrizes) com expressões algébricas simples e intuitivas.

1.4.1 - Álgebra de vetores

Vamos criar dois vetores:

```
> a <- 1:5
> b <- 6:10
```

Alguns exemplos de como efetuamos operações com esses vetores:

```
> d <- a*b
> a
[1] 1 2 3 4 5
> b
[1]  6  7  8  9 10
> d
[1]  6 14 24 36 50
> a+b
[1]  7  9 11 13 15
> a-b
[1] -5 -5 -5 -5 -5
> b-a
[1] 5 5 5 5 5
> a/b
[1] 0.1666667 0.2857143 0.3750000 0.4444444 0.5000000
> b %% a
[1] 0 1 2 1 0
```

A grande vantagem de R é que não precisamos criar ‘loops’ para efetuar operações em vetores (ou matrizes).

Podemos também efetuar operações de vetores com valores números isolados:

```
> a*3
[1]  3  6  9 12 15
> a/4
[1] 0.25 0.50 0.75 1.00 1.25
> b-5
[1] 1 2 3 4 5
```

1.4.2 - Comparações lógicas

R possui uma série de operadores lógicos:

==	***igual***
!=	***diferente***
>	***maior***
<	***menor***
>=	***maior ou igual***
<=	***menor ou igual***
&	***E (AND) lógico***
|	***OU (OR) lógico***

Vejamos alguns exemplos:

```
> b-5==a
[1] TRUE TRUE TRUE TRUE TRUE
> a!=b
[1] TRUE TRUE TRUE TRUE TRUE
> a > 2
[1] FALSE FALSE  TRUE  TRUE  TRUE
> a < 2
[1]  TRUE FALSE FALSE FALSE FALSE
> b >6 & b < 8
[1] FALSE  TRUE FALSE FALSE FALSE
> b > 9 | a < 2
[1]  TRUE FALSE FALSE FALSE  TRUE
> a >=4
[1] FALSE FALSE FALSE  TRUE  TRUE
```

1.4.3 - Funções matemáticas

R possui uma variedade de funções matemáticas e estatísticas que podem ser usadas para operações com arrays (vetores e matrizes).

Abaixo temos alguns exemplos: sqrt é raiz quadrada, min é o mínimo valor, range é a faixa de valores (min e max), mean é a média, prod é o produto de todos os valores, sd é o desvio padrão e exp é o exponencial:

```
> sqrt(a)
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
> sqrt(b)
[1] 2.449490 2.645751 2.828427 3.000000 3.162278
> min(a)
[1] 1
> min(b)
[1] 6
> range(b)
[1]  6 10
> range(a)
[1] 1 5
> mean(a)
[1] 3
> prod(a)
[1] 120
> sd(a)
[1] 1.581139
> exp(b)
[1]   403.4288  1096.6332  2980.9580  8103.0839 22026.4658
```

1.4.4 - Números aleatórios

É bastante comum a criação de vetores de números aleatórios na análise de dados, e também para criar um modelo ou entender se um procedimento funciona. Para obter 10 números aleatórios amostrados de forma uniforme entre 0 e 1 usamos:

```
> ale<-runif(10)
> ale
 [1] 0.93575073 0.75748417 0.11579646 0.41033991 0.65282156 0.61987880
 [7] 0.73218410 0.03723520 0.78086560 0.06122219
```

Para obtermos números distribuídos de forma Normal (com valores de média e desvio padrão pré-definidos) usamos rnorm:

```
> rno<-rnorm(10,mean=10.4001,sd=0.05005)
> mean(rno)
[1] 10.42109
> sd(rno)
[1] 0.0510869
> rno
 [1] 10.41448 10.38759 10.46205 10.47412 10.38051 10.37194 10.46489 10.47035
 [9] 10.45487 10.33007
```

Se executarmos novamente o comando obteremos diferentes valores, seria o esperado pois são aleatórios! Mas, números gerados em computação não são verdadeiramente aleatórios, mas pseudoaleatórios. Para trabalharmos com análise de dados muitas vezes precisamos de usar os mesmos números aleatórios todas as vezes que repetirmos o experimento ou procedimento. Podemos fazer isso com a função set.seed.

Veja abaixo como funciona:

```
> set.seed(12)
> runif(3)
[1] 0.06936092 0.81777520 0.94262173
> runif(4)
[1] 0.26938188 0.16934812 0.03389562 0.17878500
> runif(5)
[1] 0.641665366 0.022877743 0.008324827 0.392697197 0.813880559
> set.seed(12)
> runif(3)
[1] 0.06936092 0.81777520 0.94262173
> runif(5)
[1] 0.26938188 0.16934812 0.03389562 0.17878500 0.64166537
> set.seed(12)
> runif(3)
[1] 0.06936092 0.81777520 0.94262173
> runif(5)
[1] 0.26938188 0.16934812 0.03389562 0.17878500 0.64166537
```

Note que toda vez que set.seed é chamada a mesma sequência aleatória é executada, conforme queríamos.

O número é arbitrário e uma sequência de números diferente ocorre se mudarmos o número do set.seed:

```
> set.seed(12)
> runif(5)
[1] 0.06936092 0.81777520 0.94262173 0.26938188 0.16934812
> set.seed(120)
> runif(5)
[1] 0.3919077 0.1260168 0.3446131 0.7954961 0.1960155
```

1.4.5 - Álgebra de matrix

A computação de matrizes é também vetorizada, ou seja, dada uma matriz m podemos efetuar m * 5 para multiplicar todos os valores da matriz por 5 ou executar m^2 para elevar a mesma ao quadrado.

Vejamos alguns exemplos:

```
> m<- matrix(1:6, ncol=3,nrow=2,byrow=TRUE)
> m
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> m * 2
     [,1] [,2] [,3]
[1,]    2    4    6
[2,]    8   10   12
> m^2
     [,1] [,2] [,3]
[1,]    1    4    9
[2,]   16   25   36
```

Nós podemos também multiplicar uma matriz por um vetor lembrando que se o número de elementos do vetor for menor o mesmo será reciclado (sentido operação coluna – próxima coluna):

```
> m * c(2,3,4)
     [,1] [,2] [,3]
[1,]    2    8    9
[2,]    12  10   24
```

Podemos multiplicar duas matrizes, mas a multiplicação será célula por célula:

```
> m*m
     [,1] [,2] [,3]
[1,]    1    4    9
[2,]   16   25   36
```

Para usar álgebra linear, ou seja, multiplicação de matrizes usamos %*%.

Nesse caso multiplicando a matriz m pela sua transposta :

```
> m %*% t(m)
     [,1] [,2]
[1,]   14   32
[2,]   32   77
```

Dando prosseguimento, vamos ver como ler e escrever arquivos usando R.

1.5 – Lendo e escrevendo arquivos

Em muitos casos, o primeiro passo na análise de dados é ler arquivos. Isto pode ser bastante complicado devido à grande variação de formatos de arquivos. Aqui mostraremos como ler arquivos do tipo texto no formato planilha que poderão ser colocados numa estrutura do tipo data.frame, que é o procedimento mais comum. Pegamos um arquivo csv ou tab e colocamos na estrutura de dados (data.frame).

Para ler um arquivo, primeiro precisamos saber o seu nome e em qual diretório ele está localizado.

Usamos uma das seguintes formas: a) Colocando essa informação ('path' ou caminho para o arquivo) em uma variável como uma string (não use simples \, use / ou \\ ):

```
> f1 <- "C:/projects/research/data/test.csv"
> f2 <- "C:\\projects\\research\\data\\test.csv"
```

b) Na prática trabalhamos com arquivos localizados no diretório de trabalho corrente, desta forma podemos abrir o arquivo sem o caminho completo, apenas fornecendo o seu nome diretamente.

Obtemos o diretório corrente com a função getwd:

```
> getwd()
[1] "/home/lanjal/workR"
```

Criaremos no diretório de trabalho um arquivo chamado furos.csv com dados sobre furos de sonda. Copie e cole no seu editor de texto e salve como furos.csv:

```
furo,easting,northing,elev,dip,az
AT-001,300300,7989500,400,-90,0
AT-002,300656,7989345,410,-60,123
AT-003,300567,7989789,380,-60,180
AT-004,300123,7989843,350,-45,270
```

Primeiro checamos se o arquivo existe e depois leremos este dado usando:

```
> file.exists('furos.csv')
[1] TRUE
> d <- read.csv('furos.csv', stringsAsFactors=FALSE)
> head(d)
    furo easting northing elev dip  az
1 AT-001  300300  7989500  400 -90   0
2 AT-002  300656  7989345  410 -60 123
3 AT-003  300567  7989789  380 -60 180
4 AT-004  300123  7989843  350 -45 270
> class(d)
[1] "data.frame"
```

Vemos que os dados são carregados automaticamente em um data.frame.

Escrevemos um arquivo a partir de uma data.frame usando:

```
> write.csv(d, 'furos2.csv', row.names=FALSE)
```

Podemos ler um arquivo linha por linha usando:

```
> webpage<- readLines('http://www.r-project.org')
> webpage
 [1] "<!DOCTYPE html>"
 [2] "<html lang=\"en\">"
.
.
.

[123] "
</body>"
[124] "</html>"
```

Trabalhando com arquivos csv é a forma mais eficiente e rápida de importar a informação dentro do ambiente R.

Outros formatos podem ser carregados usando funções específicas para o tal.

No próximo capítulo vamos importar os dados de um arquivo e mostrar funções que nos auxiliam a inspecionar os dados nele contido.

1.6 – Inspecionando os dados

Depois de abrirmos dados de um arquivo, o próximo passo é dar uma inspecionada neles usando algumas funções estatísticas. Primeiro dê uma olhada nos dados em geral para ver se algo estranho ou incoerente aparece. Isso é muito importante pois evitará problemas mais adiante. Vamos ver algumas ferramentas de inspeção.

Criaremos no diretório de trabalho um arquivo chamado amostra.csv com dados sobre análises. Copie e cole no seu editor de texto e salve como amostra.csv:

```
amostra,batch, Au(ppm),Ag(ppm), Cu(ppm)
smp-01,c3,0.03,0.05,1.5
smp-02,c3,0.13,0.15,5.5
smp-02,c3,0.14,0.25,6.5
smp-04,c3,0.10,0.25,9.5
smp-05,c3,0.08,0.24,9.5
smp-06,c3,0.04,0.25,11.5
smp-07,c1,0.03,0.23,8.5
smp-08,c3,0.03,0.22,5.5
smp-08,c3,-999.00,0.05,2.5
smp-10,c3,0.05,0.15,2.5
```

Depois use:

```
> smp <- read.csv('amostra.csv', stringsAsFactors=FALSE)
> smp
   amostra batch Au.ppm. Ag.ppm. Cu.ppm.
1   smp-01    c3    0.03    0.05     1.5
2   smp-02    c3    0.13    0.15     5.5
3   smp-02    c3    0.14    0.25     6.5
4   smp-04    c3    0.10    0.25     9.5
5   smp-05    c3    0.08    0.24     9.5
6   smp-06    c3    0.04    0.25    11.5
7   smp-07    c1    0.03    0.23     8.5
8   smp-08    c3    0.03    0.22     5.5
9   smp-08    c3 -999.00    0.05     2.5
10  smp-10    c3    0.05    0.15     2.5
```

Com a função summary vemos estatística dos dados coluna por coluna (se aplicável):

```
> summary(smp)
   amostra             batch              Au.ppm.           Ag.ppm.
 Length:10          Length:10          Min.   :-999.000   Min.   :0.0500
 Class :character   Class :character   1st Qu.:   0.030   1st Qu.:0.1500
 Mode  :character   Mode  :character   Median :   0.045   Median :0.2250
                                       Mean   : -99.837   Mean   :0.1840
                                       3rd Qu.:   0.095   3rd Qu.:0.2475
                                       Max.   :   0.140   Max.   :0.2500

    Cu.ppm.
 Min.   : 1.50
 1st Qu.: 3.25
 Median : 6.00
 Mean   : 6.30
 3rd Qu.: 9.25
 Max.   :11.50
```

Observamos o seguinte:
- O batch deveria ser c1 para todas as amostras;
- Temos duas amostras 02;
- Temos um valor -999 que deveria ser mudado para NA.

Podemos acessar as colunas do data.frame usando o operador $. Dessa forma podemos mudar os valores conforme necessário.

Mudando o número da amostra:

```
> smp$amostra[3]<- 'smp-03'
> smp$amostra
 [1] "smp-01" "smp-02" "smp-03" "smp-04" "smp-05" "smp-06" "smp-07" "smp-08"
 [9] "smp-08" "smp-10"
```

Mudando o batch:

```
> i<- smp$batch == 'c3' #retorna o(s) índice(s) do elemento do vetor = ‘c3’
> smp$batch[i]<- 'c1'
> smp$batch
 [1] "c1" "c1" "c1" "c1" "c1" "c1" "c1" "c1" "c1" "c1"
```

E mudaremos o valor -999 para NA

```
> smp$Au.ppm.[9]<- NA
> smp$Au.ppm.
 [1] 0.03 0.13 0.14 0.10 0.08 0.04 0.03 0.03   NA 0.05
```

Vendo todos os dados usando novamente summary:

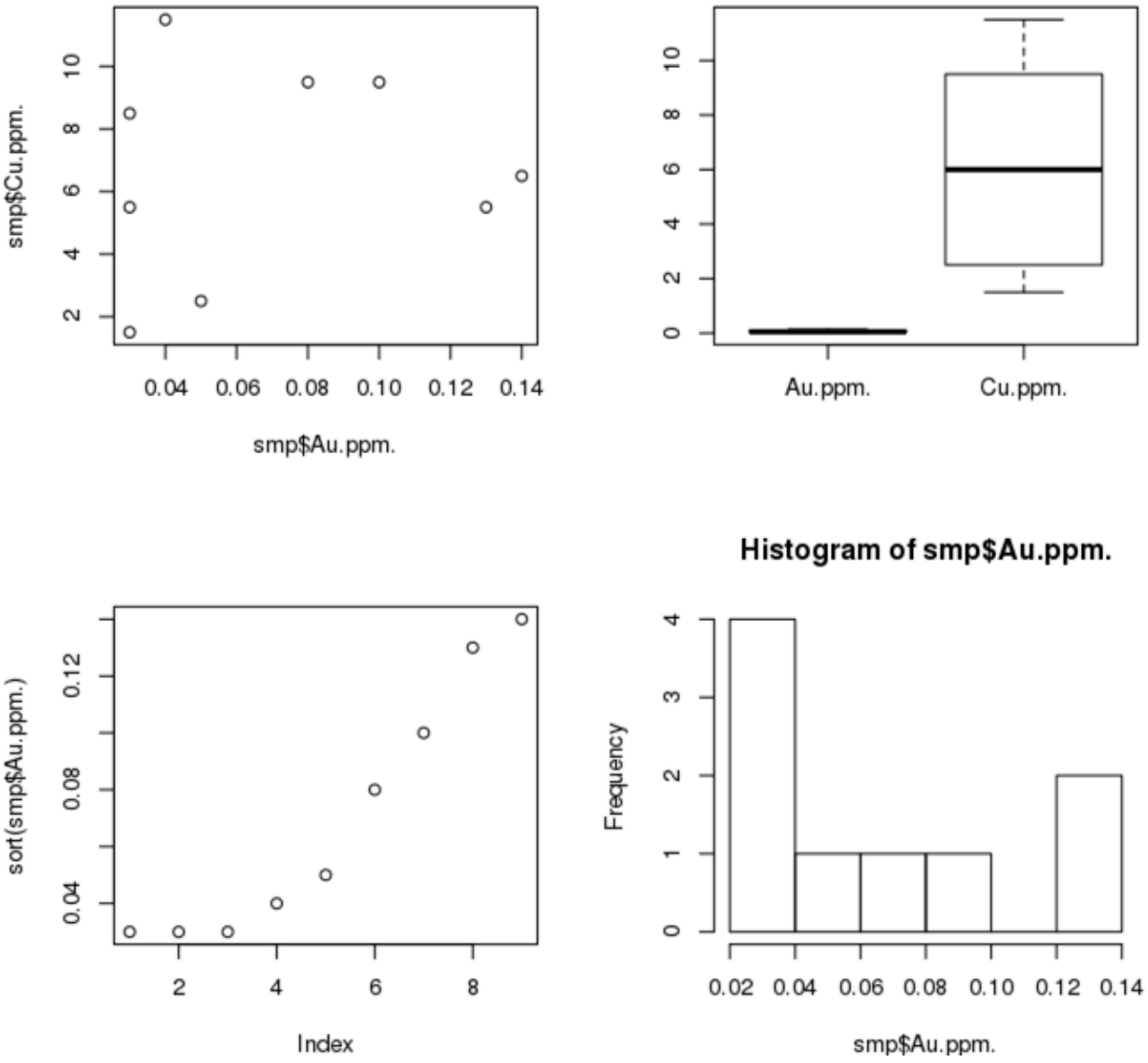
```
> smp
   amostra batch Au.ppm. Ag.ppm. Cu.ppm.
1   smp-01    c1    0.03    0.05     1.5
2   smp-02    c1    0.13    0.15     5.5
3   smp-03    c1    0.14    0.25     6.5
4   smp-04    c1    0.10    0.25     9.5
5   smp-05    c1    0.08    0.24     9.5
6   smp-06    c1    0.04    0.25    11.5
7   smp-07    c1    0.03    0.23     8.5
8   smp-08    c1    0.03    0.22     5.5
9   smp-08    c1      NA    0.05     2.5
10  smp-10    c1    0.05    0.15     2.5
> summary(smp)
   amostra             batch              Au.ppm.        Ag.ppm.
 Length:10          Length:10          Min.   :0.03   Min.   :0.0500
 Class :character   Class :character   1st Qu.:0.03   1st Qu.:0.1500
 Mode  :character   Mode  :character   Median :0.05   Median :0.2250
                                       Mean   :0.07   Mean   :0.1840
                                       3rd Qu.:0.10   3rd Qu.:0.2475
                                       Max.   :0.14   Max.   :0.2500
                                       NA's   :1
    Cu.ppm.
 Min.   : 1.50
 1st Qu.: 3.25
 Median : 6.00
 Mean   : 6.30
 3rd Qu.: 9.25
 Max.   :11.50
```

Agora os dados estão OK.

Vamos ver algumas outras ferramentas gráficas para inspeção visual de dados:

```
> par(mfrow=c(2,2)) #cria área de plotagem de 2 linhas e duas colunas
> plot(smp$Au.ppm.,smp$Cu.ppm.)
> boxplot(smp[,c('Au.ppm.','Cu.ppm.')])
> plot(sort(smp$Au.ppm.))
> hist(smp$Au.ppm.)
```

Gráficos gerados:



Podemos começar a ver o poder de R com relação a análise de dados. Vamos ver agora funções no ambiente R.

1.7 – Funções

Até o momento nós usamos várias funções em R. Por exemplo c, matrix, read.csv e sum. Funções são usadas ou chamadas sempre digitando-se o nome seguido por parêntesis. Muitas vezes, mas nem sempre, fornecemos argumentos dentro dos parêntesis. Se você não digita o parêntesis, as propriedades e definições desta função são descritas.

Vejamos:

```
> nrow
function (x)
dim(x)[1L]
<bytecode: 0x230ca50>
<environment: namespace:base>
```

Nessa descrição vemos que a função nrow usa um único argumento x. Ela chama outra função chamada dim e retorna o seu primeiro elemento no formato inteiro (1L).

Agora vamos ver a função dim que é uma função primitiva de R que usa um argumento:

```
> dim
function (x)  .Primitive("dim")
```

Agora vamos executar a função nrow mas primeiro precisamos de criar o objeto argumento que possui linhas e colunas (um objeto matrix por exemplo).

Criando uma matrix e chamando a função nrow:

```
> m<- matrix(1:6,nrow=2,ncol=3,byrow=TRUE)
> nrow(m)
[1] 2
```

1.7.1 - Escrevendo funções novas

R vem com milhares de funções mas em alguns casos podemos criar funções quando:
- precisamos descrever melhor e de forma mais clara uma tarefa em particular no fluxo de sua análise de dados;
- reutilizar código em vez de repetir sempre os mesmos passos várias vezes;
- criar uma função que é o argumento de uma outra função, é uma prática comum em R.

Escrever as nossas próprias funções não é difícil. Abaixo temos uma bem simples chamada f. Este é um nome arbitrário. Ela não leva nenhum argumento e sempre retorna 'Use R'.

Criando função f:

```
> f <- function() {
+     return('Use R')
+ }
> f
function() {
    return('Use R')
}
> f()
[1] "Use R"
```

Esta função f é meio sem graça, vamos implementar ela para usar um argumento e retornar esse argumento de alguma forma.

Implementando função f para usar um argumento:

```
> f <-  function(nome) {
+     x<-  paste('Use', nome)
+     return(x)
+ }
> f('R')
[1] "Use R"
> f('GDAL também')
[1] "Use GDAL também"
```

Vamos agora criar uma função que retorna a hipotenusa:

```
> hipotenusa<-function(a,b){
+     h <-  sqrt(a^2+b^2)
+     return(h)
+ }
> hipotenusa(2,5)
[1] 5.385165
> hipotenusa(1,1)
[1] 1.414214
> hipotenusa(3,4)
[1] 5
```

Podemos definir os valores padrões para os argumentos usados:

```
> hipotenusa<-function(a=1,b=1){
+     h <-  sqrt(a^2+b^2)
+     return(h)
+ }
> hipotenusa()
[1] 1.414214
> hipotenusa(8)
[1] 8.062258
> hipotenusa(2,5)
[1] 5.385165
```

A função seguinte é um jogo pedra papel e tesoura em R.

```
> p.p.t<-function(aposta){
+ mine<-floor(runif(n=1,min=1,max=3.9999))
+ print(paste("Sua aposta é",aposta))
+ if(mine==1){
+ print("Minha aposta é pedra")
+ if(aposta=='pedra')print('Empatou')
+ if(aposta=='papel')print('Você ganhou!!!')
+ if(aposta=='tesoura')print('Eu ganhei!!!!!!!!!!!')
+ }
+ if(mine==2){
+ print("Minha oposta é tesoura")
+ if(aposta=='pedra')print('Você ganhou!!')
+ if(aposta=='tesoura')print('Empatou')
+ if(aposta=='papel')print('Eu ganhei!!!!!!!!!!')
+ }
+ if(mine==3){
+ print("Minha aposta é papel")
```

```
+ if(aposta=='papel')print('Empatou')
+ if(aposta=='tesoura')print('Você ganhou!!')
+ if(aposta=='pedra')print('Eu ganhei!!!!!!!!!!')
+ }
}
> p.p.t('pedra')
[1] "Sua aposta é pedra"
[1] "Minha aposta é pedra"
[1] "Empatou"
> p.p.t('tesoura')
[1] "Sua aposta é tesoura"
[1] "Minha aposta é pedra"
[1] "Eu Ganhei!!!!!!!!!!"
> p.p.t('papel')
[1] "Sua aposta é papel"
[1] "Minha aposta é papel"
[1] "Empatou"
```

Um exemplo de uma função que retorna o número de valores únicos de um vetor:

```
> único <-  function(x) {
+     length(unique(x))
+ }
> único(c('a','a','a','b','c','b'))
[1] 3
> único(c(1,1,3,4,5,1))
[1] 4
```

Para obter informações de ajuda sobre uma função existente em R digite ***?*** Antes do nome da função mas sem parêntesis:

```
> ?sum
```

Estes foram apenas alguns exemplos de demonstração de funções em R, você pode criar funções melhores para exercer tarefas mais complexas. Pratique sempre.

1.8 – A Família ‘apply’

Vamos mostrar agora uma família de funções muito importante em R, essa família de funções englobam as funções apply, tapply, aggregate, sapply e lapply.

Estas funções fornecem um meio eficiente e elegante de efetuar operações em porções de uma array ou data.frame, seja em uma coluna ou uma linha em uma matriz e data.frame ou num elemento de uma lista.

Criando uma matrix m:

```
> m <-  matrix(1:15, nrow=5, ncol=3)
> m
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
```

1.8.1 - apply

Cálculo computacional em matrizes é ‘vetorizado’, Por exemplo m * 5 para multiplicar todos os valores da matriz m por 5 ou m ^ 2 para elevar todos os valores ao quadrado. Mas frequentemente precisamos de computar valores somente para partes de uma matriz, uma simples operação para cada linha ou coluna.

A função apply faz isso:

```
> apply(m,1,sum)
[1] 18 21 24 27 30
> apply(m,1,mean)
[1]  6  7  8  9 10
> apply(m,2,sum)
[1] 15 40 65
> apply(m,2,mean)
[1]  3  8 13
```

Observe que apply usa pelo menos 3 argumentos: uma matriz, 1 para linhas ou 2 para colunas, e uma função para calcular o novo valor (ou valores) para cada linha ou coluna conforme especificado em 1 ou 2. Em muitos casos você pode adicionar o parâmetro na.rm=TRUE para remover qualquer NA porque ***em R uma operação que envolve NA sempre vai resultar em NA***.

No exemplo acima usamos as funções sum (soma) e mean (média) mas poderíamos ter usado uma criada por nós mesmo, nesse exemplo nossa função retorna um vetor de valor da soma de cada coluna adicionando o valor um:

```
> maisUm<- function(x){
+ return(sum(x)+1)
+ }
> apply(m,1,maisUm)
[1] 19 22 25 28 31
```

1.8.2 -tapply

Está função pode ser usada para fornecer um sumário estatístico (ex.: média) para grupos de linhas num data.frame. Você precisa de uma coluna que especifique o grupo e assim você pode calcular valores estatísticos para cada grupo.

Criaremos um data.frame onde mostramos os resultados de duas fontes:

```
> colnames(m) <-  c('valor1', 'valor2', 'valor3')
> d <- data.frame(nome=c('Fonte 1', 'Fonte 1', 'Fonte 1', 'Fonte 2', 'Fonte
2'), m, stringsAsFactors=FALSE)
> d$valor2[1]<-NA
> d
     nome valor1 valor2 valor3
1 Fonte 1      1     NA     11
2 Fonte 1      2      7     12
3 Fonte 1      3      8     13
4 Fonte 2      4      9     14
5 Fonte 2      5     10     15
```

Imagine agora que você queira calcular a média para cada valor em cada grupo individual (Fonte):

```
> tapply(d$valor1,d$nome,mean)
Fonte 1 Fonte 2
    2.0     4.5
> tapply(d$valor2,d$nome,mean)
Fonte 1 Fonte 2
     NA     9.5
> tapply(d$valor2,d$nome,mean,na.rm=TRUE)
Fonte 1 Fonte 2
    7.5     9.5
> tapply(d$valor3,d$nome,mean)
Fonte 1 Fonte 2
   12.0    14.5
```

1.8.3 - aggregate

Esta função é similar a tapply mas apresenta o resultado para várias colunas ao mesmo tempo, o segundo argumento deve ser uma lista (não pode ser um vetor)

Veja exemplo com média e desvio padrão:

```
> aggregate(d[, c('valor1', 'valor2', 'valor3')], list(d$nome), mean,
na.rm=TRUE)
  Group.1 valor1 valor2 valor3
1 Fonte 1    2.0    7.5   12.0
2 Fonte 2    4.5    9.5   14.5
> aggregate(d[, c('valor1', 'valor2', 'valor3')], list(d$nome), sd, na.rm=TRUE)
  Group.1    valor1    valor2    valor3
1 Fonte 1 1.0000000 0.7071068 1.0000000
2 Fonte 2 0.7071068 0.7071068 0.7071068
```

1.8.4 - sapply e lapply

Para interagir com uma lista podemos usar lapply ou sapply. A diferença é que lapply retorna uma lista e sapply tenta simplificar o resultado num vetor ou matrix.

Contando o número de caracteres de cada elemento da lista com a função nchar:

```
> minerios <- list('Ouro', 'Prata', 'Urânio', 'Cassiterita', 'Ferro')
> lapply(minerios,nchar)
[[1]]
[1] 4

[[2]]
[1] 5

[[3]]
[1] 6

[[4]]
[1] 11

[[5]]
[1] 5

> sapply(minerios,nchar)
[1]  4  5  6 11  5
```

Pratique bastante essas funções. Vamos a seguir ver controles de fluxo em R.

1.9 – Controles de fluxo

A maioria dos programas possuem dois tipos de controle de fluxo: interação e alternação. Interação é feito via ‘loops’ e alteração via ramificações ‘if-then-else’.

Vamos aqui mostrar rapidamente como usar esses controles de fluxo em R:

1.9.1 - loop-for

Um loop **for** básico é mostrado abaixo. O que o exemplo faz é: para cada valor associado a i entre 1 e 3 executa o que está entre os colchetes {}:

```
> for (i in 1:3) {
+     print('Olá!')
+ }
[1] "Olá!"
[1] "Olá!"
[1] "Olá!"
```

O exemplo abaixo ilustra melhor: soma os valores 1, 2 e 3 e carrega o resultado em j.

```
> j<- 0
> for (i in 1:3){
+ print(i)
+ j<- j+i
+ }
[1] 1
[1] 2
[1] 3
> j
[1] 6
```

Muitas vezes queremos incluir uma condição para parar o loop ou para pular um passo. Isso é feito usando break e next.

Veja o exemplo onde pulamos os valores ímpares e paramos quando passamos 8 no loop que vai até 10:

```
> for (i in 1:10) {
+     if (i %in% c(1,3,5,7)) {
+         next
+     }
+     if (i > 8) {
+         break
+     }
+     print(i)
+ }
[1] 2
[1] 4
[1] 6
[1] 8
```

1.9.2 - loop-while

O loop **while** executa uma tarefa repetidamente até o critério de parada ou break ser encontrado.

Nesse exemplo vemos como funciona o loop: enquanto i < 4 execute:

```
> i <-  0
> while (i < 4) {
+    print(paste(i, 'e contando ...'))
+    i <-  i + 1
+ }
[1] "0 e contando ..."
[1] "1 e contando ..."
[1] "2 e contando ..."
[1] "3 e contando ..."
```

1.9.3 - if-them-else

Ramificações do tipo **if then else** são importantes em qualquer linguagem de programação e não seria diferente em R.

Pegamos duas variáveis x e y e queremos mudar o valor de y dependendo do valor x. Checamos o valor de x se este é igual (==) a cinco, como é verdade (TRUE) o novo valor de y e atribuído:

```
> x<- 5
> y<- 10
> if (x==5){
+ y<- 15
+ }
> y
[1] 15
```

Agora usamos **if-then-else** para testar o valor de x e atribuir o novo valor de y, o último else é executado somente se os parâmetros acima não forem verdadeiros:

```
> if (x > 20) {
+     y <-  y + 2
+ } else if (x > 5 & x < 10) {
+     y <-  y - 1
+ } else {
+     y <-  x
+ }
> y
[1] 5
```

Agora combinando **for** e **if-then-else**:

```
> a <-  1:5
> f <-  vector(length=length(a))
> for (i in 1:length(a)) {
+     if (a[i] > 2) {
+         f[i] <-  a[i] / 2
+     } else {
+         f[i] <-  a[i] * 2
+     }
+ }
> f
[1] 2.0 4.0 1.5 2.0 2.5
```

1.10 – Preparação de dados

Um dos pontos importantes em análise espacial é como organizar os dados existentes, gastamos muito tempo nesta etapa e aqui vamos ver algumas funções em R que nos ajudarão a preparar nossos dados.

Estas funções são reshape e merge.

1.10.1 - reshape

A função reshape nos permite formatar dados usando duas opções: largo para longo e longo para largo. Qual o formato que melhor se adapta à sua análise de dados é escolha sua. Geralmente largo para longo é a melhor opção mas vamos ver as duas abaixo. Copie estes dados abaixo e grave no diretório de trabalho com lab_res.csv.

amostra, batch, lab, au, ag, pt, cu, as, zn, pb
1, 23-A, LHY,0.03,0.09,0.03,0.84,0.30,0.4,0.5
2, 23-A, LHY,0.04,0.07,0.02,0.54,0.35,0.5,0.4
3, 23-A, LHY,0.05,0.04,0.02,0.64,0.34,0.6,0.5
4, 23-A, LHY,0.06,0.01,0.01,0.74,0.27,0.7,0.6
5, 23-A, LHY,0.06,0.02,0.03,0.56,0.14,0.8,0.5
6, 23-A, LHY,0.05,0.03,0.05,0.64,0.21,0.9,0.7
7, 23-A, LHY,0.04,0.05,0.07,0.74,0.34,0.8,0.4
8, 24-A, LHY,0.03,0.09,0.03,0.84,0.56,0.7,0.6
9, 24-A, LHY,0.02,0.07,0.05,0.92,0.35,0.6,0.4
10, 24-A, LHY,0.01,0.06,0.05,0.56,0.36,0.4,0.3
11, 24-A, LHY,0.03,0.05,0.03,0.67,0.32,0.45,0.8
12, 24-A, LHY,0.05,0.06,0.02,0.78,0.54,0.3,0.76
13, 24-A, LHY,0.07,0.07,0.01,0.23,0.34,0.2,0.65
14, 24-A, LHY,0.08,0.08,0.03,0.34,0.24,0.3,0.46
15, 24-A, LHY,0.09,0.09,0.02,0.45,0.34,0.4,0.61
16, 24-A, LHY,0.09,0.10,0.02,0.78,0.23,0.5,0.63

Importe os dados :

```
> lr<- read.csv('lab_res.csv',header=TRUE)
> lr
   amostra batch  lab   au   ag   pt   cu   as   zn   pb
1        1  23-A  LHY 0.03 0.09 0.03 0.84 0.30 0.40 0.50
2        2  23-A  LHY 0.04 0.07 0.02 0.54 0.35 0.50 0.40
3        3  23-A  LHY 0.05 0.04 0.02 0.64 0.34 0.60 0.50
4        4  23-A  LHY 0.06 0.01 0.01 0.74 0.27 0.70 0.60
5        5  23-A  LHY 0.06 0.02 0.03 0.56 0.14 0.80 0.50
6        6  23-A  LHY 0.05 0.03 0.05 0.64 0.21 0.90 0.70
7        7  23-A  LHY 0.04 0.05 0.07 0.74 0.34 0.80 0.40
8        8  24-A  LHY 0.03 0.09 0.03 0.84 0.56 0.70 0.60
9        9  24-A  LHY 0.02 0.07 0.05 0.92 0.35 0.60 0.40
10      10  24-A  LHY 0.01 0.06 0.05 0.56 0.36 0.40 0.30
11      11  24-A  LHY 0.03 0.05 0.03 0.67 0.32 0.45 0.80
12      12  24-A  LHY 0.05 0.06 0.02 0.78 0.54 0.30 0.76
13      13  24-A  LHY 0.07 0.07 0.01 0.23 0.34 0.20 0.65
14      14  24-A  LHY 0.08 0.08 0.03 0.34 0.24 0.30 0.46
15      15  24-A  LHY 0.09 0.09 0.02 0.45 0.34 0.40 0.61
16      16  24-A  LHY 0.09 0.10 0.02 0.78 0.23 0.50 0.63
```

Criaremos um data.frame longo com um elemento e resultado por linha, cada amostra repetirá 7 vezes, uma para cada elemento:

```
> longlr <- reshape(lr,
+   varying = c("au", "ag", "pt", "cu", "as","zn","pb"),
+   v.names = "ppm",
+   timevar= “elem”,
+   times = c("au", "ag", "pt", "cu", "as","zn","pb"),
+   new.row.names = 1:1000,
+   direction = "long")
```

E como resultado temos:

```
> longlr[order(longlr$id),]
    amostra batch  lab elem  ppm id
1         1  23-A  LHY   au 0.03  1
17        1  23-A  LHY   ag 0.09  1
33        1  23-A  LHY   pt 0.03  1
49        1  23-A  LHY   cu 0.84  1
65        1  23-A  LHY   as 0.30  1
81        1  23-A  LHY   zn 0.40  1
97        1  23-A  LHY   pb 0.50  1
2         2  23-A  LHY   au 0.04  2
18        2  23-A  LHY   ag 0.07  2
34        2  23-A  LHY   pt 0.02  2
50        2  23-A  LHY   cu 0.54  2
66        2  23-A  LHY   as 0.35  2
82        2  23-A  LHY   zn 0.50  2
98        2  23-A  LHY   pb 0.40  2
3         3  23-A  LHY   au 0.05  3
19        3  23-A  LHY   ag 0.04  3

.
.
.

16       16  24-A  LHY   au 0.09 16
32       16  24-A  LHY   ag 0.10 16
48       16  24-A  LHY   pt 0.02 16
64       16  24-A  LHY   cu 0.78 16
80       16  24-A  LHY   as 0.23 16
96       16  24-A  LHY   zn 0.50 16
112      16  24-A  LHY   pb 0.63 16
```

Comparando os dois data.frames usando dim:

```
> dim(lr)
[1] 16 10
> dim(longlr)
[1] 112   6
```

Agora vamos transformar este data.frame longo para largo:

```
> larglr <- reshape(longlr,
+   timevar = "elem",
+   idvar = c("id","amostra", "batch", "lab"),
+   direction = "wide")
```

E o resultado é:

```
> larglr
   amostra batch  lab id ppm.au ppm.ag ppm.pt ppm.cu ppm.as ppm.zn ppm.pb
1        1  23-A  LHY  1   0.03   0.09   0.03   0.84   0.30   0.40   0.50
2        2  23-A  LHY  2   0.04   0.07   0.02   0.54   0.35   0.50   0.40
3        3  23-A  LHY  3   0.05   0.04   0.02   0.64   0.34   0.60   0.50
4        4  23-A  LHY  4   0.06   0.01   0.01   0.74   0.27   0.70   0.60
5        5  23-A  LHY  5   0.06   0.02   0.03   0.56   0.14   0.80   0.50
6        6  23-A  LHY  6   0.05   0.03   0.05   0.64   0.21   0.90   0.70
7        7  23-A  LHY  7   0.04   0.05   0.07   0.74   0.34   0.80   0.40
8        8  24-A  LHY  8   0.03   0.09   0.03   0.84   0.56   0.70   0.60
9        9  24-A  LHY  9   0.02   0.07   0.05   0.92   0.35   0.60   0.40
10      10  24-A  LHY 10   0.01   0.06   0.05   0.56   0.36   0.40   0.30
11      11  24-A  LHY 11   0.03   0.05   0.03   0.67   0.32   0.45   0.80
12      12  24-A  LHY 12   0.05   0.06   0.02   0.78   0.54   0.30   0.76
13      13  24-A  LHY 13   0.07   0.07   0.01   0.23   0.34   0.20   0.65
14      14  24-A  LHY 14   0.08   0.08   0.03   0.34   0.24   0.30   0.46
15      15  24-A  LHY 15   0.09   0.09   0.02   0.45   0.34   0.40   0.61
16      16  24-A  LHY 16   0.09   0.10   0.02   0.78   0.23   0.50   0.63
```

Comparando o original com o largo usando dim voltamos a 16 linhas de dados:

```
> dim(larglr)
[1] 16 11
> dim(lr)
[1] 16 10
```

1.10.2 - merge

Podemos unir dois data.frames com mesmo número de linhas com um campo em comum usando merge, primeiro vamos criar os data.frame(s):

```
> furos <-  data.frame(
+     furo = c("At-001", "At-002", "At-003", "At-004", "At-005"),
+     sonda = c("lf", "bs", "lf", "bs", "bs"),
+     completo = c("yes", rep("no", 4)))
> survey <-  data.frame(
+     name = c("At-001", "At-002", "At-003", "At-004", "At-005"),
+     dip = c(-60,-60,-60,-45,-90),
+     az = c(45,270,180,270,0))
```

E agora usamos merge para unir os dois:

```
> mg<- merge(furos, survey, by.x="furo", by.y="name")
> mg
    furo sonda completo dip  az
1 At-001    lf      yes -60  45
2 At-002    bs       no -60 270
3 At-003    lf       no -60 180
4 At-004    bs       no -45 270
5 At-005    bs       no -90   0
```

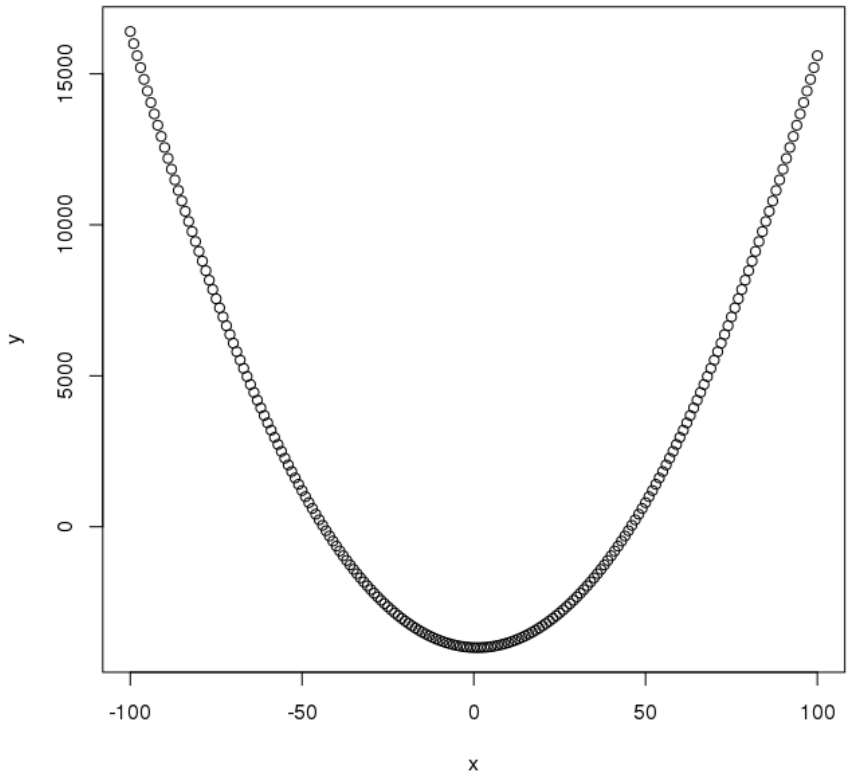
1.11 – Gráficos

Em R podemos gerar diversos tipos de gráficos, vamos aqui mostrar como fazemos isso.

Existem várias maneiras diferentes para fazer gráficos. Vamos aqui lidar com gráficos de pontos usando o pacote “base”.

Vamos utilizar o seguinte dado para criar os primeiros gráficos:

```
> x<-c(-100:100)
> y<-2*x^2-4*x-4000
> plot(x,y)
```



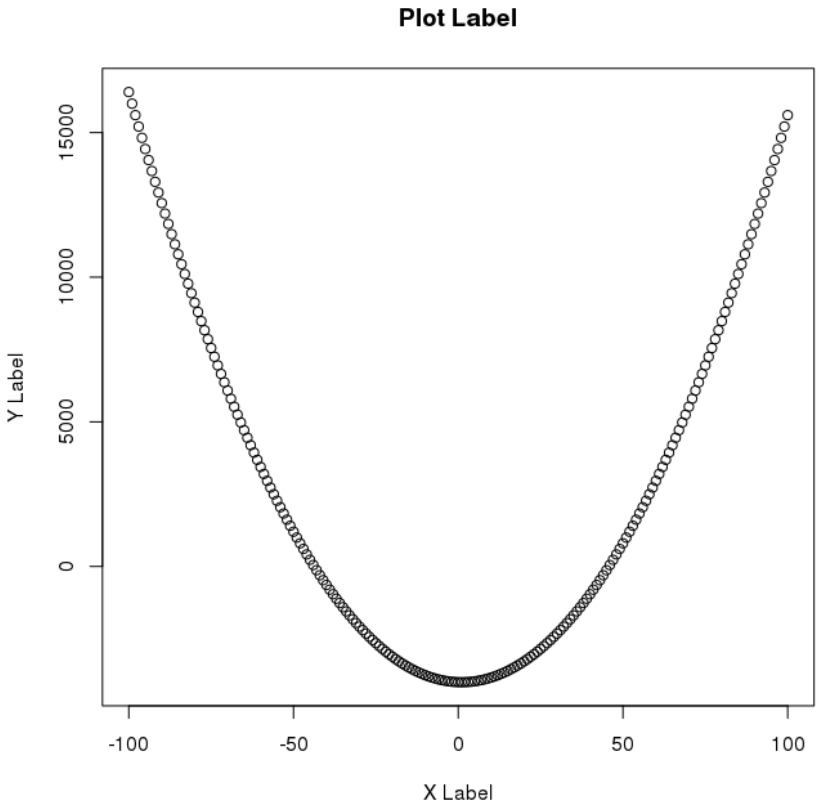
No ambiente R os gráficos são criados em uma nova janela quando chamamos as funções de plotagem.

1.11.1 - Parâmetros da função plot

A função plot tem vários parâmetros. Vamos ver alguns deles.

Parâmetros de rótulos. Use main para rotular o gráfico. Para rotular o eixo X use xlab e use ylab para rotular o eixo Y.

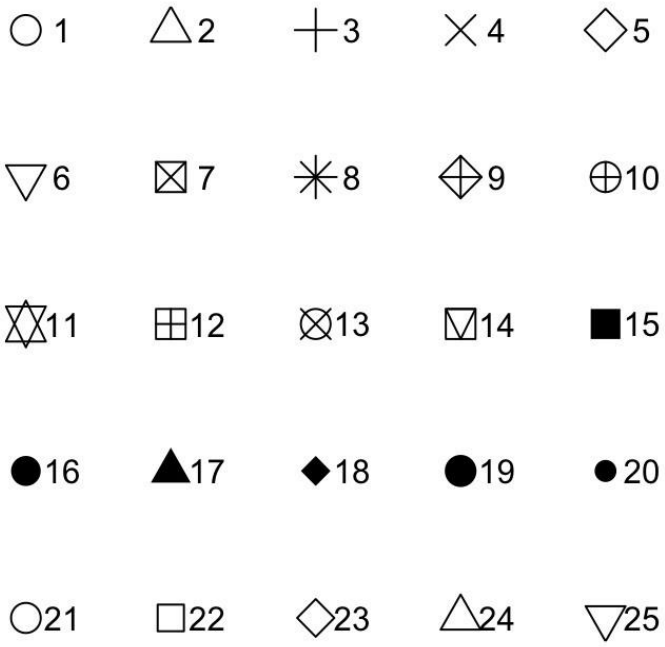
```
> plot(x,y,xlab='X Label',ylab='Y Label',main='Plot Label')
```



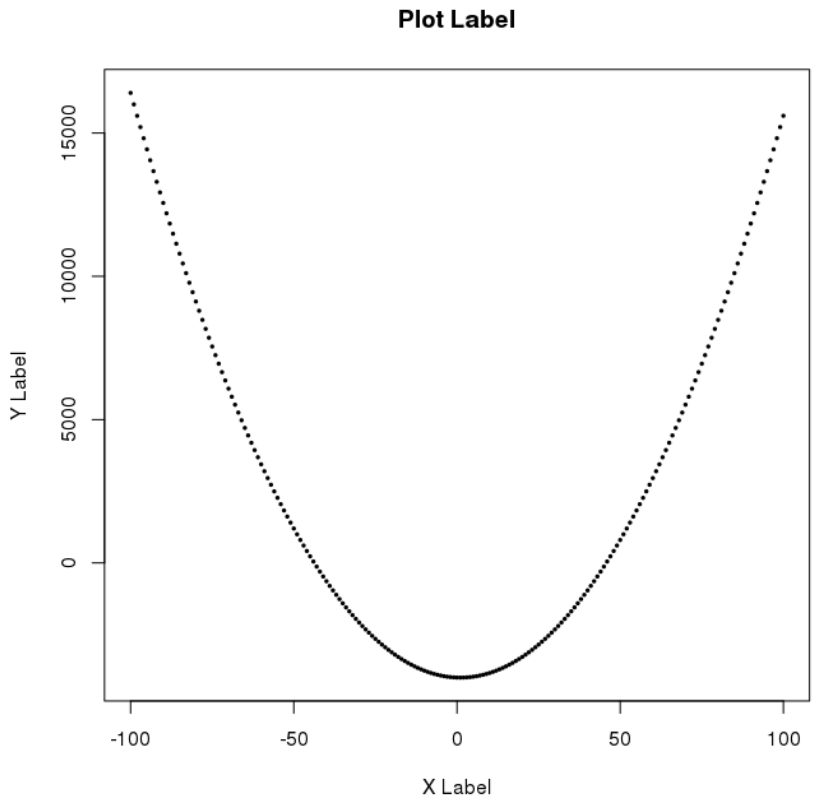
Para fechar a janela podemos usar a função dev.off:

```
> dev.off()
null device
          1
```

Use os parâmetros cex para definir o tamanho do símbolo a ser usado e pch para mudar o formato do símbolo. Veja abaixo uma tabela com os pch dos símbolos.

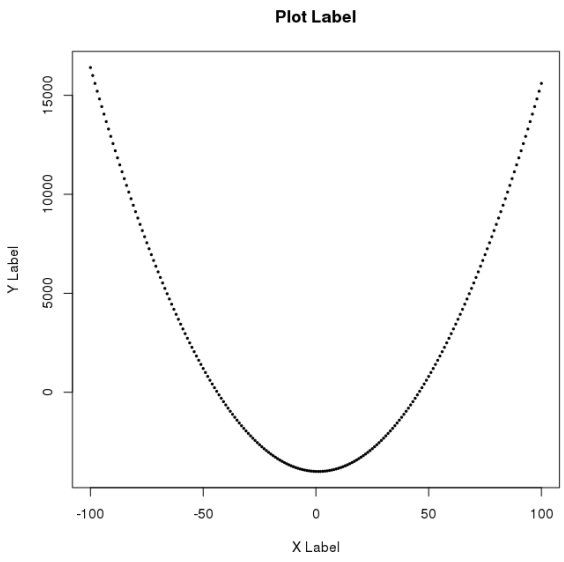


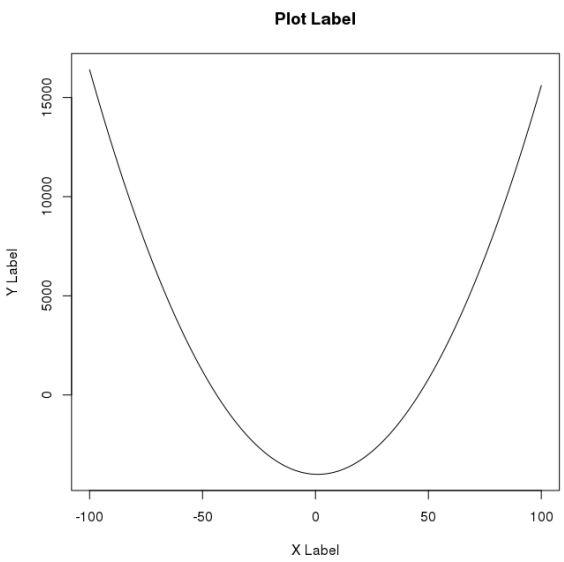
```
> plot(x,y,xlab='X Label',ylab='Y Label',main='Plot Label',pch=20,cex=0.5)
```

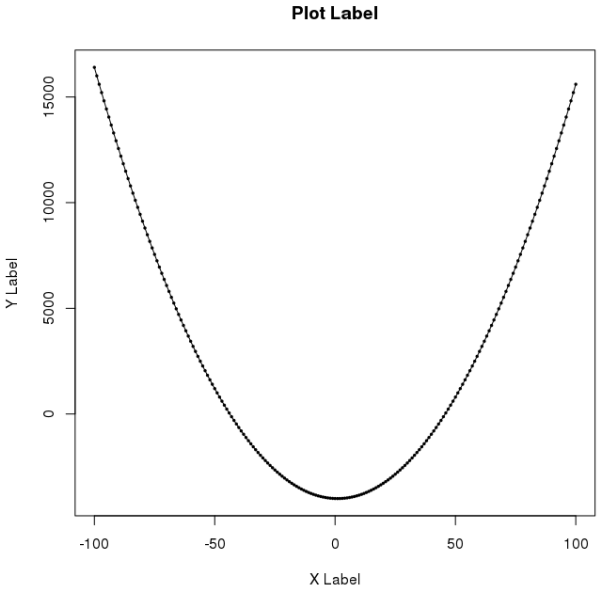


Use o parâmetro type para selecionar entre símbolo, linha ou os dois.

```
> plot(x,y,xlab='X Label',ylab='Y Label',main='Plot ',pch=20,cex=0.5,type= 'p')
> plot(x,y,xlab='X Label',ylab='Y Label',main='Plot ',pch=20,cex=0.5,type= 'l')
> plot(x,y,xlab='X Label',ylab='Y Label',main='Plot ',pch=20,cex=0.5,type= 'o')
```

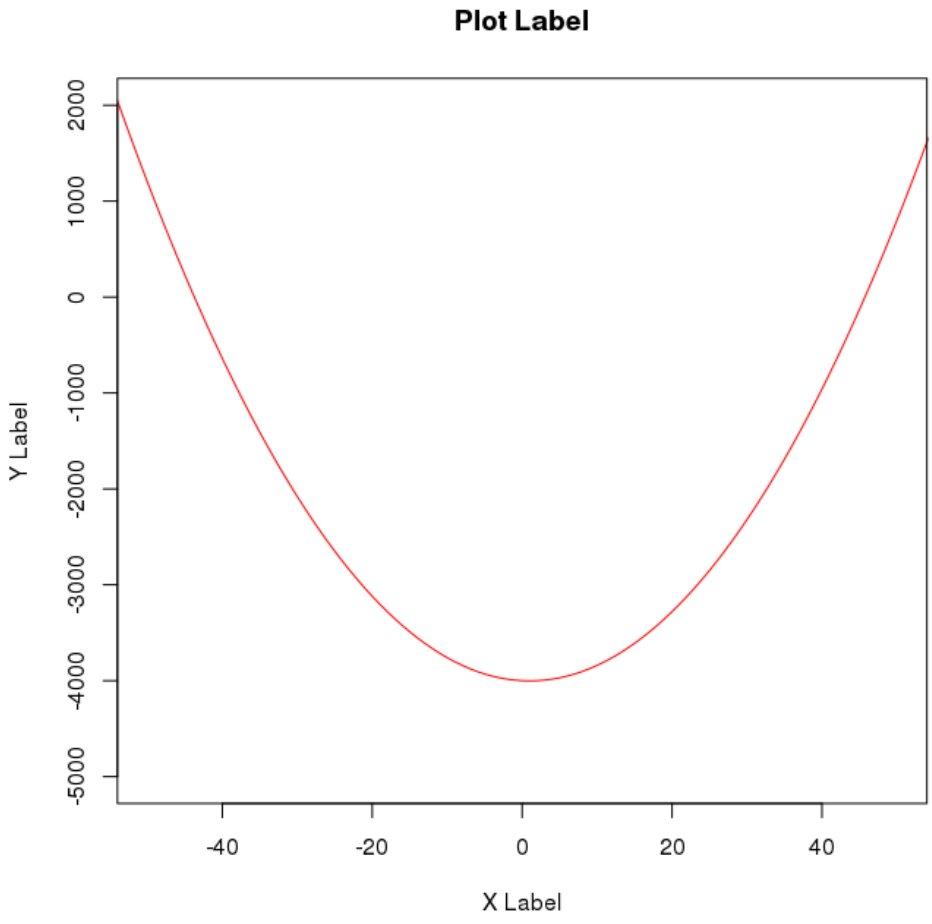






Use col para definir uma cor e xlim e ylim para definir a faixa de X e Y a ser plotada.

```
> plot(x,y,xlab='X Label',ylab='Y Label',main='Plot Label',pch=20, cex=0.5,
type='l',col='red',xlim=c(-50,50),ylim=c(-5000,2000))
```

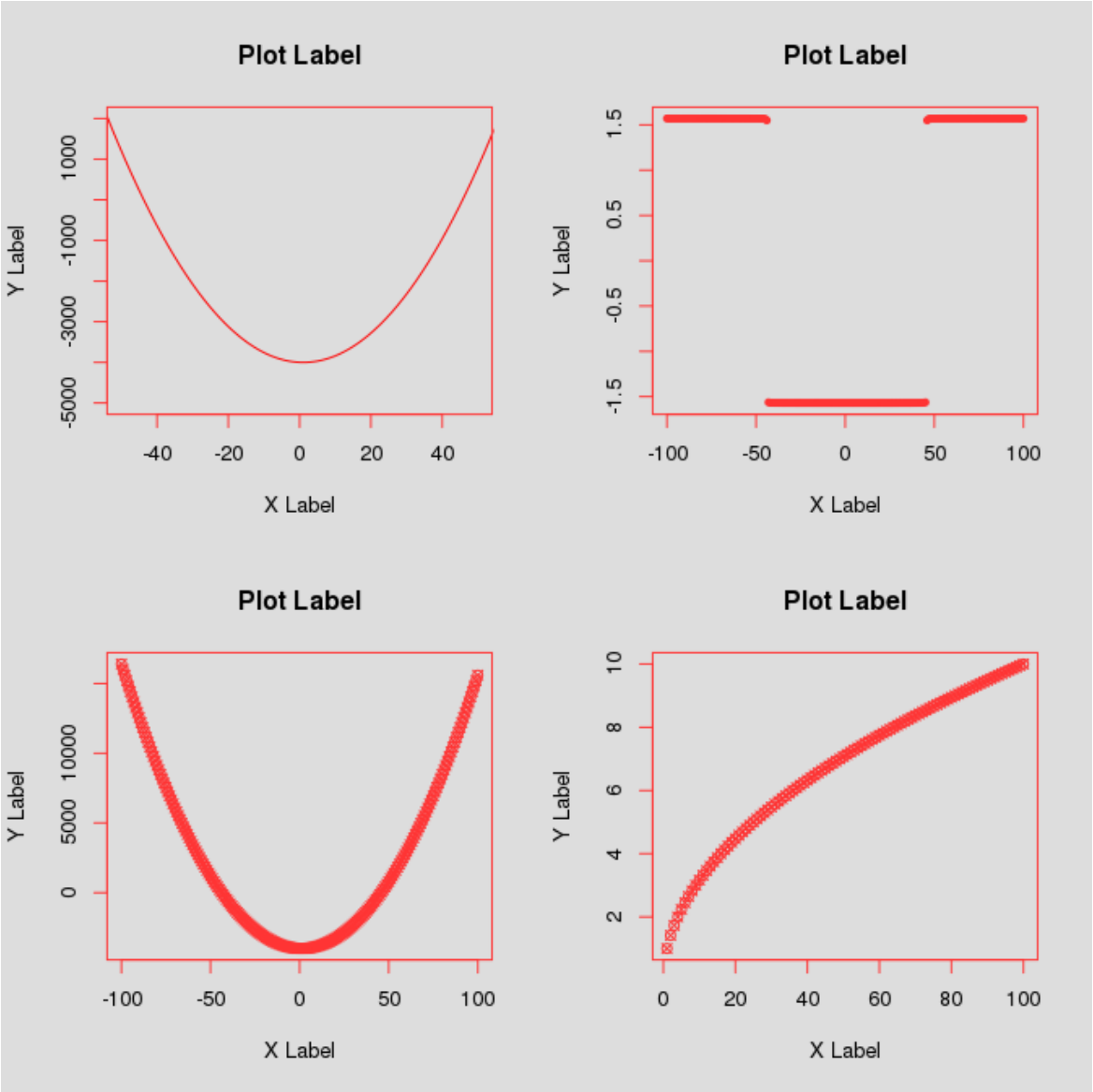


1.11.2 - Parâmetros da função par

A função par prepara a janela de plotagem. Existem vários ajustes nela que podem ser manipulados. Vamos ver alguns deles:

O parâmetro mfrow define o número de subplots organizados em linhas e colunas e o parâmetro bg e fg definem as cores de fundo e frente. Para fechar a janela de plotagem ao seu original use dev.off()

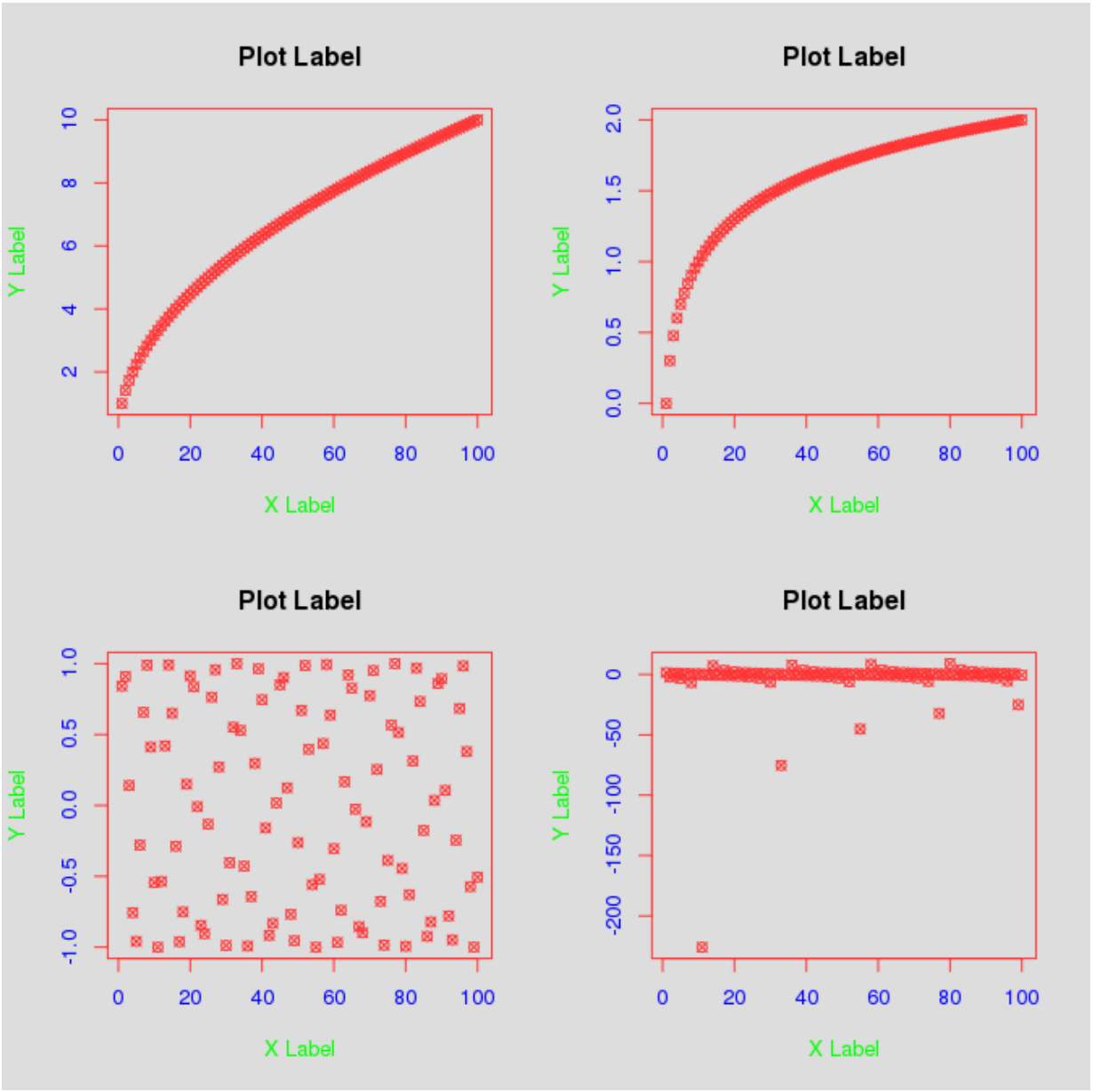
```
> par(mfrow=c(2,2),bg='#dddddd',fg='#ff3434')
> plot(x,y,xlab='X Label',ylab='Y Label',main='Plot Label',pch=20, cex=0.5,
type='l',
col='red',xlim=c> plot(x,atan(y),xlab='X Label',ylab='Y Label',main='Plot
Label',pch=20)
> plot(x,y,xlab='X Label',ylab='Y Label',main='Plot Label',pch=13)
> plot(c(1:100),sqrt(c(1:100)),xlab='X Label',ylab='Y Label',main='Plot Label',
pch=13)
```



Mudando as cores de rótulos e eixos usando par

```
> par(mfrow=c(2,2),bg='#dddddd',fg='#ff3434',col.axis='blue',col.lab='green')
```

```
> plot(c(1:100),sqrt(c(1:100)),xlab='X Label',ylab='Y Label',main='Plot Label',
pch=13)
> plot(c(1:100),log10(c(1:100)),xlab='X Label',ylab='Y Label',main='Plot
Label', pch=13)
> plot(c(1:100),sin(c(1:100)),xlab='X Label',ylab='Y Label',main='Plot Label',
pch=13)
> plot(c(1:100),tan(c(1:100)),xlab='X Label',ylab='Y Label',main='Plot Label',
pch=13)
```



1.12 – Modelos estatísticos

Neste capítulo vamos fazer um 'curso relâmpago' de estatística elementar. Vamos cobrir bem superficialmente todos os pontos da estatística básica usando R (fonte: www.r-tutor.com/elementary-statistics, modificado).

Primeiramente vamos importar a biblioteca (library) MASS que contém os data.frames necessários:

```
> library(MASS)
```

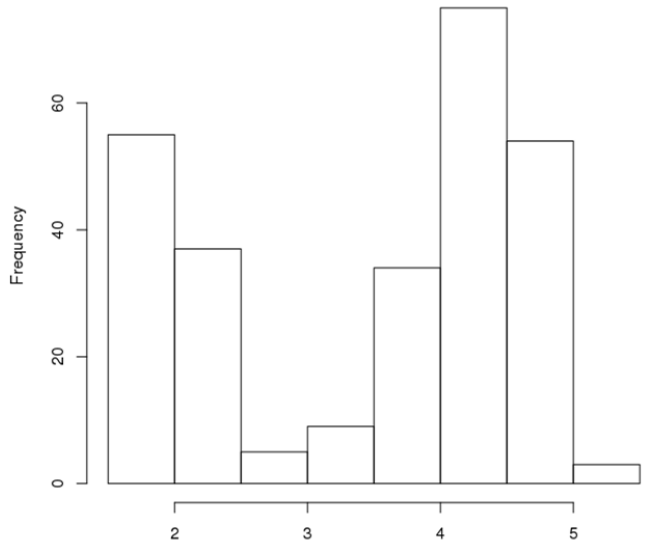
1.12.1 - Dados quantitativos

A frequência de distribuição de um dado é um resumo da ocorrência destes em categorias não sobrepostas. Ou seja, pegamos a distribuição total destes dados e distribuímos em faixas.

```
> duracao<- faithful$eruptions #carrega a duração das erupções
> range(duracao) # exibe a faixa de duração das erupções
[1] 1.6 5.1
> quebar = seq(1.5,5.5,by=0.5) #cria a quebra em intervalos de 0.5
> duracao.dividido<-cut(duracao,quebar,right=FALSE) #classifica as erupções nas
quebras
> duracao.freq<- table(duracao.dividido) # cria as faixas de frequência dos
dados
> cbind(duracao.freq) #mostra na forma de coluna
        duracao.freq
[1.5,2)           51
[2,2.5)           41
[2.5,3)            5
[3,3.5)            7
[3.5,4)           30
[4,4.5)           73
[4.5,5)           61
[5,5.5)            4
```

Ou podemos criar diretamente o histograma com:

```
> hist(faithful$eruptions)
```



A frequência relativa é a frequência dividida pelo tamanho da amostragem.

Assim a frequência relativa seria:

```
> duracao.freqrel<- duracao.freq/nrow(faithful)
> old<- options(digits=1)
> cbind(duracao.freq,duracao.freqrel)
        duracao.freq duracao.freqrel
[1.5,2)           51            0.19
[2,2.5)           41            0.15
[2.5,3)            5            0.02
[3,3.5)            7            0.03
[3.5,4)           30            0.11
[4,4.5)           73            0.27
[4.5,5)           61            0.22
[5,5.5)            4            0.01
```

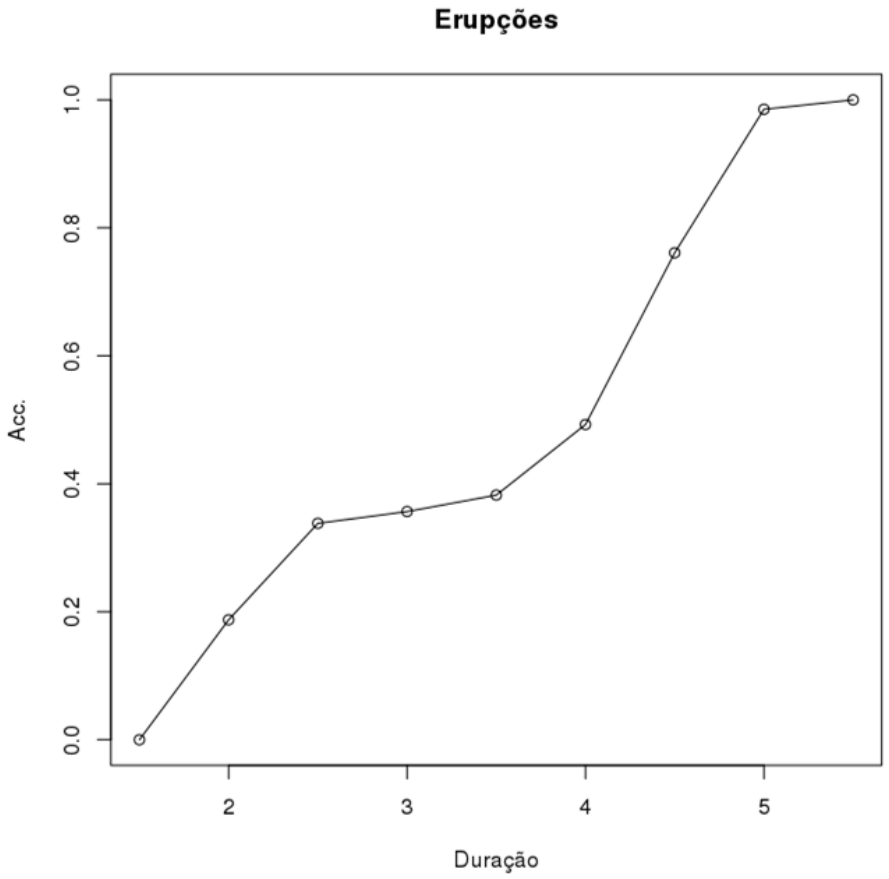
Agora veremos estas mesmas frequências distribuídas na forma acumulada.

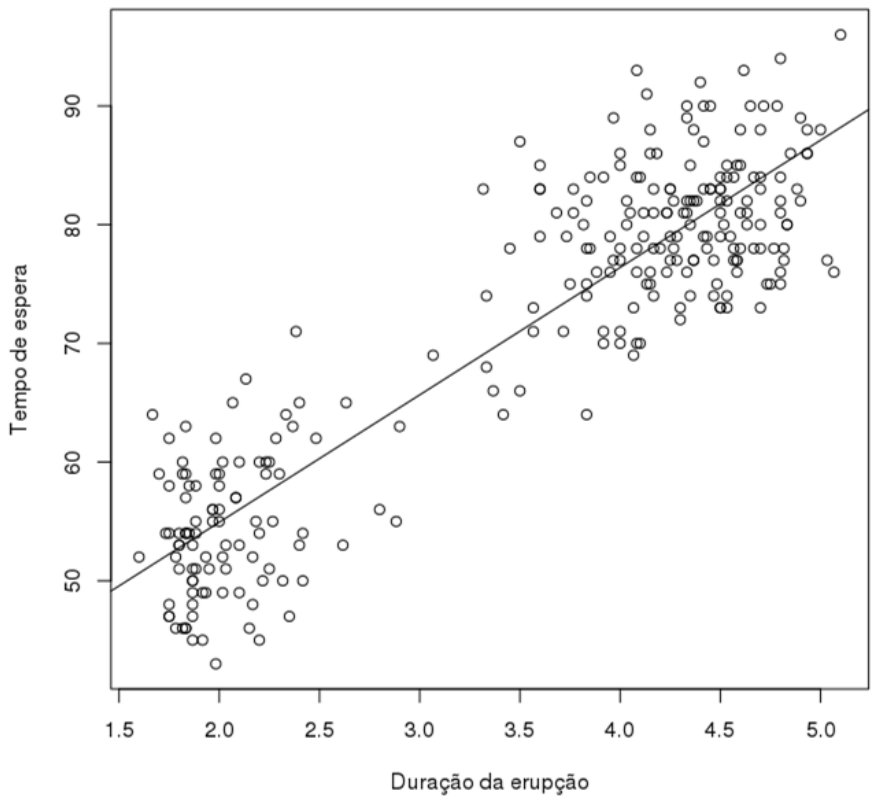
```
> duracao.freqcum<- cumsum(duracao.freq)
> cbind(duracao.freqcum)
        duracao.freqcum
[1.5,2)              51
[2,2.5)              92
[2.5,3)              97
[3,3.5)             104
[3.5,4)             134
[4,4.5)             207
[4.5,5)             268
[5,5.5)             272

> duracao.freqrelcum<- duracao.freqcum/nrow(faithful)
> old<- options(digits=2)
> cbind(duracao.freqcum,duracao.freqrelcum)
        duracao.freqcum duracao.freqrelcum
[1.5,2)              51               0.19
[2,2.5)              92               0.34
[2.5,3)              97               0.36
[3,3.5)             104               0.38
[3.5,4)             134               0.49
[4,4.5)             207               0.76
[4.5,5)             268               0.99
[5,5.5)             272               1.00
```

Criando o gráfico das frequências relativas acumuladas e gráfico de erupções x espera com ‘trend’ de regressão linear:

```
> cumrelfreq<- c(0,duracao.freqrelcum)
> plot(quebra,cumrelfreq,main="Erupções",xlab="Duração",ylab="Acc.")
> lines(quebra,cumrelfreq)
> plot(faithful[1:2],xlab="Duração da erupção",ylab="Tempo de espera")
> abline(lm(faithful$waiting~faithful$eruptions))
```


Erupções
Acc.
0.0
0.2
0.4
0.6
0.8
1.0
2
3
4
5
Duração


Tempo de espera
50
60
70
80
90
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Duração da erupção

1.12.2 - Medidas numéricas

Média é a soma de todos os valores dividido pelo número de amostragem:

```
> mean(duracao)
[1] 3.487783
```

Mediana é o valor central dos dados organizados de forma ascendente:

```
> median(duracao)
[1] 4
```

Quartis são os valores que cortam os 25%, 50% (mediana) , 75% e 100% dos dados.

```
> quantile(duracao)
     0%     25%     50%     75%    100%
1.60000 2.16275 4.00000 4.45425 5.10000
```

Percentil: valor que corta n% dos dados.

```
> quantile(duracao,c(.05,.95))
   5%   95%
1.800 4.817
```

Extensão é o valor da diferença entre o maior e o menor valor

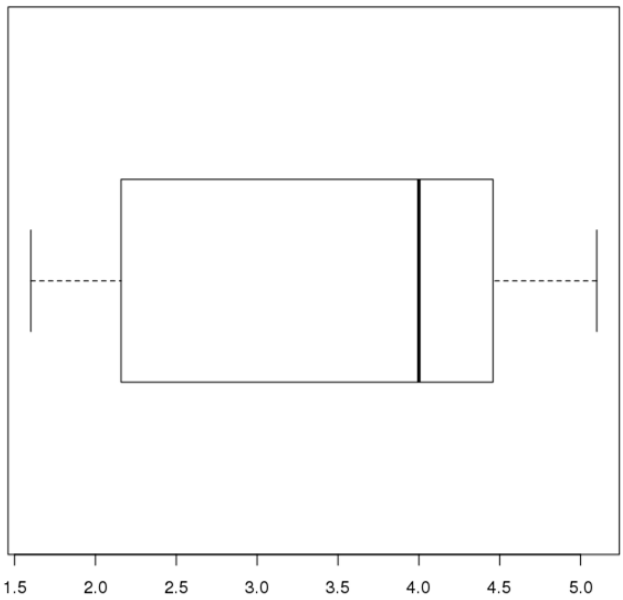
```
> max(duracao)-min(duracao)
[1] 3.5
```

Extensão interquartil é a diferença entre o valor do quartil 75% menos o valor do quartil 25%:

```
> IQR(duracao)
[1] 2.2915
```

Boxplot é a representação dos quartis e da extensão dos dados

```
> boxplot(duracao, horizontal=TRUE)
```



Variância é a medida de como os valores estão dispersos em volta da média;

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

```
> var(duracao)
[1] 1.302728
```

Desvio padrão é a raiz quadrada da variância:

```
> sd(duracao)
[1] 1.141371
```

A covariância de duas variáveis num conjunto de dados mede como estes estão linearmente relacionados.

$$s_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

Uma covariância positiva indica uma correlação positiva entre as duas variáveis:

```
> cov(duracao,faithful$waiting)
[1] 13.97781
```

O coeficiente de correlação entre duas variáveis em um conjunto de dados é igual a covariância dividida por seus desvios padrão. É uma medida normalizada de como as duas variáveis são linearmente relacionadas.

$$r_{xy} = \frac{s_{xy}}{s_x s_y}$$

Um coeficiente próximo a 1 indica que as duas variáveis são positivamente relacionadas.

```
> cor(duracao,faithful$waiting)
[1] 0.9008112
```

O momento central pode ser calculado usando a seguinte fórmula:

$$m_k = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^k$$

Em particular, o segundo momento central de uma amostragem é praticamente a sua variância, abaixo calculamos o momento central de terceira ordem:

```
> library(e1071)
> moment(duracao, order=3, center=TRUE)
[1] -0.6149059
```

Skewness (assimetria ou obliquidade) é uma medida de simetria dos dados. Como regra, valores negativos indicam média menor que a mediana (mais para esquerda) e valores positivos indicam média maior que a mediana (mais para direita).

$$\gamma_1 = \mu_3 / \mu_2^{3/2}$$

Encontrando o skewness:

```
> library(e1071)
> skewness(duracao)
[1] -0.4135498
```

Curtose é uma medida de forma que caracteriza o achatamento da curva da função de distribuição de probabilidade, se negativo é chamada platicúrtica (achatada), se 0 é mesocúrtica (normal) e se positiva é chamada leptocúrtica (comprida):

$$\gamma_2 = \mu_4 / \mu_2^2 - 3$$

Achando a curtose:

```
> library(e1071)
> kurtosis(duracao)
[1] -1.511605
```

1.12.3 - Distribuição probabilística

A **distribuição binomial** é uma distribuição de probabilidade discreta. Ela descreve o resultado de tentativas em um experimento e cada tentativa tem dois resultados possíveis (binomial). Podemos expressar pela função abaixo:

$$f(x) = \begin{pmatrix} n \\ x \end{pmatrix} p^x (1-p)^{(n-x)} \quad where\ x = 0, 1, 2, ..., n$$

Qual seria a probabilidade de termos 4 respostas certas em 12 tentativas onde cada tentativa temos 20% de acerto (5 opções)? Em R calculamos assim e temos 92.7% de chances de acertarmos 4 vezes:

```
> pbinom(4,size=12,prob=0.2)
[1] 0.9274445
```

A **distribuição de Poisson** é a distribuição de probabilidade da ocorrência de eventos independentes em um intervalo. A função abaixo mostra essa distribuição onde λ é a média:

$$f(x) = \frac{\lambda^x e^{-\lambda}}{x!} \quad where\ x = 0, 1, 2, 3, ...$$

Temos 12 carros cruzando uma ponte em média, qual a probabilidade de termos 17 carros cruzando a ponte em determinado minuto? Em R calculamos assim e temos como resposta 10,13% de chances disso acontecer:

```
> ppois(16, lambda=12, lower=FALSE)
[1] 0.101291
```

A **distribuição contínua uniforme** é a distribuição de probabilidade de números aleatórios selecionados de um intervalo contínuo entre a e b. Sua função é definida por:

$$f(x) = \begin{cases} \frac{1}{b-a} & when\ a \leq x \leq b \\ 0 & when\ x < a\ or\ x > b \end{cases}$$

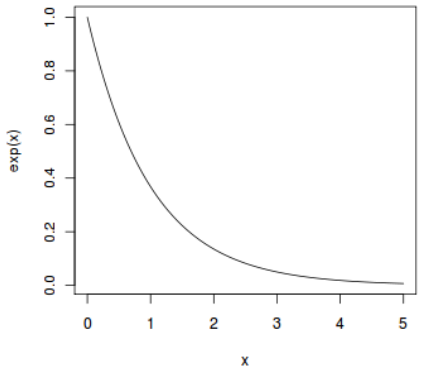
Em R geramos 8 números aleatórios num intervalo (1 e 3) usando:

```
> runif(8, min=1, max=3)
[1] 1.578376 1.893886 2.518432 2.450899 1.095584 1.096872 1.046974 2.052164
```

A **distribuição exponencial** descreve o tempo de chegada de uma sequência de eventos independentes aleatoriamente recorrentes. Se μ é a media do tempo de espera para o próximo evento recorrer, sua função probabilística será:

$$f(x) = \begin{cases} \frac{1}{\mu} e^{-x/\mu} & when\ x \geq 0 \\ 0 & when\ x < 0 \end{cases}$$

Abaixo está um gráfico de distribuição exponencial com μ=1:



Suponhamos que o tempo que um caixa leva para cobrar uma compra seja 3 minutos, qual seria a probabilidade do caixa cobrar em menos de 2 minutos? Em R calculamos assim (1/3 de compras por minuto), a resposta seria 48.7% de chances de concluir a compra em menos de 2 minutos:

```
> pexp(2,rate=1/3)
[1] 0.4865829
```

A **distribuição normal** é definida pela seguinte função, onde μ é média da população e σ^2 é a variância:

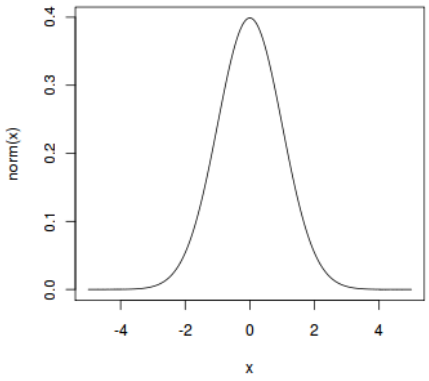
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

Se uma variável aleatória X segue uma distribuição normal, escrevemos:

$$X \sim N(\mu, \sigma^2)$$

A distribuição normal é importante devido ao **Teorema do Limite Central**, que diz que a população de todas as amostras possíveis de tamanho n de uma população de média μ e variância σ^2 se aproxima da distribuição normal com média μ e σ^2/n quando n se aproxima do infinito.

Abaixo está um gráfico de distribuição normal com μ=0 e σ=1:



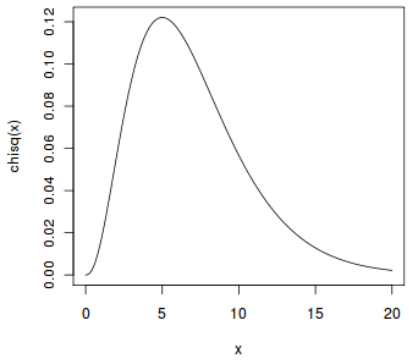
Vamos assumir que as notas de entrada num curso obedeçam a distribuição normal e a média de entrada seja 72 e o desvio padrão 15.2. Qual seria a porcentagem de estudantes que teriam nota acima de 84? Em R resolvemos assim e vemos que 21,5% teriam nota acima de 84:

```
> pnorm(84, mean=72, sd=15.2, lower.tail=FALSE)
[1] 0.2149176
```

Se $X_1, X_2, \ldots, X_m$ são m variáveis aleatórias independentes em uma distribuição normal padrão então a quantidade seguinte segue uma **distribuição Chi quadrática** com m graus de liberdade. Sua média será m e sua variância 2m.

$$V = X_1^2 + X_2^2 + \cdots + X_m^2 \sim \chi^2_{(m)}$$

Abaixo temos um gráfico de uma distribuição chi quadrática com 7 graus de liberdade:



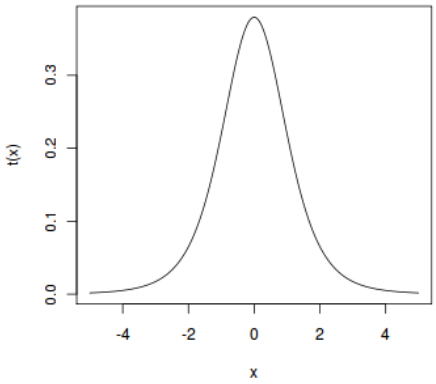
Encontre o percentil 95 de uma distribuição chi quadrática com 7 graus de liberdade. Em R acharíamos assim:

```
> qchisq(.95, df=7)
[1] 14.06714
```

Assumindo que uma variável aleatória Z tem distribuição normal padrão e outra variável V tem distribuição chi quadrática com m graus de liberdade e assumindo que as duas são independentes, então a quantidade seguinte segue uma **distribuição T student** com m graus de liberdade.

$$t = \frac{Z}{\sqrt{V/m}} \sim t_{(m)}$$

Abaixo temos um gráfico de uma distribuição T student com 5 graus de liberdade:



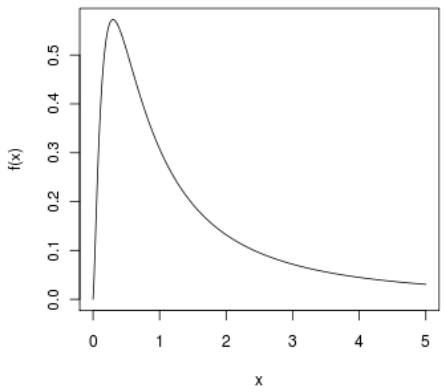
Encontre o percentil 2.5 e o 97.5 de uma distribuição student com 5 graus de liberdade. Em R achamos assim:

```
> qt(c(.025, .975), df=5)
[1] -2.570582  2.570582
```

Se V_1 e V_2 são duas variáveis aleatórias independentes com distribuição chi quadrática com grau de liberdade m_1 e m_2 respectivamente, então a seguinte quantidade segue uma **distribuição F** com numerador m_1 grau de liberdade e denominador m_2 grau de liberdade (m_1,m_2).

$$F = \frac{V_1/m_1}{V_2/m_2} \sim F_{(m_1,m_2)}$$

Abaixo temos um gráfico de uma distribuição F com (5,2) graus de liberdade:



Encontre o percentil 95 de uma distribuição F com (5,2) graus de liberdade. Em R achamos assim:

```
> qf(.95, df1=5, df2=2)
[1] 19.29641
```

1.12.4 - Estimativas de intervalo

A estimativa pontual da média de uma população (exemplo survey de library MASS) pode ser feita facilmente em R com:

```
> library(MASS)
> altura<-survey$Height
> mean(altura, na.rm=TRUE)
[1] 172.3809
```

Agora vamos estimar a média de uma população com variância conhecida para estimarmos sua precisão.

Assumindo que o desvio padrão da população 'survey' é 9.48, encontre a margem de erro e estimativa de intervalo para um nível de confiança de 95%. Em R executamos:

```
> altura<-na.omit(survey$Height) #se livra dos NA
> n = length(altura)
> sigma<-9.48
> sem<-sigma/sqrt(n) #desvio padrão da média
> sem
[1] 0.6557453
```

```
> E<-qnorm(0.975)*sem
> E  #margem de erro
[1] 1.285237
> xbar<-mean(altura)
> xbar +c(-E, E)
[1] 171.0956 173.6661
```

E agora para o caso de desconhecermos a variância

Sem assumir o desvio padrão da população altura dos estudantes, encontraremos a margem de erro com 95% de confiança usando R assim:

```
> altura<-na.omit(survey$Height) #se livra dos NA
> n = length(altura)
> s<-sd(altura)
> SE<- s/sqrt(n)
> SE
[1] 0.6811677
> E<-qt(0.975,df=n-1)*SE
> E
[1] 1.342878
> xbar<-mean(altura)
> xbar +c(-E, E)
[1] 171.0380 173.7237
```

A qualidade de uma amostragem pode ser melhorada pelo aumento do tamanho da amostragem.

Assumindo um desvio padrão de 9.48 qual seria o tamanho da amostragem para atingir uma margem de erro de 1.2 centímetros com nível de confiança de 95%? Usando R chegamos a 240:

```
> z<-qnorm(0.975)
> sigma<- 9.48
> E<-1.2
> z^2*sigma^2/E^2
[1] 239.7454
```

É possível estimar a proporção de uma característica populacional com base numa amostragem. Vamos encontrar agora a proporção de estudantes mulheres numa universidade com base na amostragem survey:

```
> library(MASS)
> sexo<-na.omit(survey$Sex)
> n<-length(sexo)
> k <-sum(sexo =="Female")
> pbar<- k/n
> pbar
[1] 0.5
```

O resultado foi 50%.

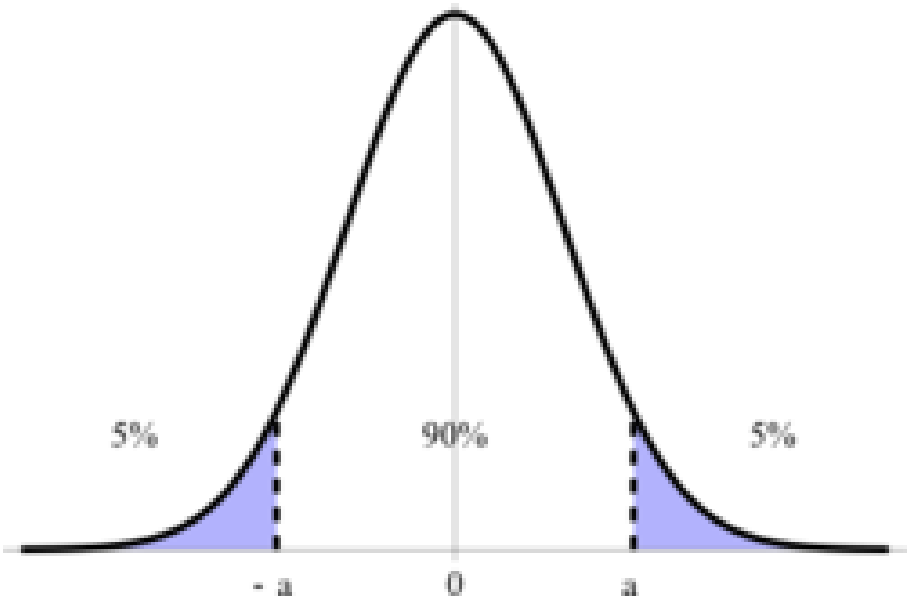
E agora para estimar o intervalo de uma proporção populacional usamos com 95% de nível de confiança:

```
> SE<-sqrt(pbar*(1-pbar)/n) #Erro padrão
> SE
[1] 0.03254723
> E<- qnorm(0.975)*SE
> E
[1] 0.06379139
```

```
> pbar + c(-E, E)
[1] 0.4362086 0.5637914
```
A 95% de nível de confiança entre 43.6% e 56.3% dos estudantes da universidade são mulheres.

Finalizando vamos calcular o tamanho da amostragem necessária para termos uma margem de erro de 5% num nível de confiança de 95%:
```
> z<-qnorm(0.975)
> p<-0.5
> E<-0.05
> z^2*p*(1-p)/E^2
[1] 384.1459
```
Precisaríamos 385 amostras para garantir margem de erro de 5% a um nível de confiança de 95% .

1.12.5 - Testando hipóteses



Cauda inferior (esquerda) e superior (direita) em cinza

A hipótese nula de um **teste de cauda inferior da média de população** pode ser expressa por:

$$\mu \geq \mu_0$$

onde μ_0 o limite inferior hipotético da média μ de uma população verdadeira.

Vamos definir o teste estatístico z em termos da média, do tamanho da amostra e do desvio padrão σ.

$$z = \frac{\bar{x} - \mu_0}{\sigma/\sqrt{n}}$$

Então a hipótese nula do teste da cauda inferior é para ser rejeitada se $z <= -z_\alpha$ onde z_α é o percentil $100(1-\alpha)$ de uma distribuição normal padrão.

Ex.: Foi dito que o teor médio de um jazimento é acima de 10%. Em uma amostragem de 30 pontos deste jazimento chegamos a um valor médio de 9.9%. Assumindo um desvio padrão de 0.12%. Num nível de significância de 0.05, podemos rejeitar o teor anunciado? Vamos usar R nesse teste:

```
> teor<- 9.9     #teor medio das 30 amostras
> mi0<-  10      #teor a ser aceito ou rejeitado
> sigma<- 0.12   #desvio padrão
> n<- 30         #tamanho da amostra
> z<- (teor-mi0)/(sigma/sqrt(n))
> z             #teste estatístico
[1] -4.564355
> alfa<- 0.05
> z.alfa<- qnorm(1-alfa)
> -z.alfa       #valor crítico
[1] -1.644854
```

Como $z<=-z_\alpha$ a hipótese de o teor ser >= 10% está rejeitada a nível de significância de 0.05 .

Da mesma forma, a hipótese nula de um **teste de cauda superior da média de população** pode ser expressa por:

$$\mu \leq \mu_0$$

onde μ_0 o limite inferior hipotético da média μ de uma população verdadeira.

Então a hipótese nula do teste da cauda superior é para ser rejeitada se $z>=z_\alpha$ onde z_α é o percentil $100(1-\alpha)$ de uma distribuição normal padrão.

Ex: Em uma frente de lavra a porcentagem de determinado contaminante de minério foi estipulada em ser no máximo 2%. Em uma amostra de 35 pontos encontramos uma média de 2.1% deste contaminante, assumindo um desvio padrão de 0.25% da população. A 0.05 de nível de significância, podemos rejeitar o valor de no máximo 2% de contaminante?

```
> teor<-2.1    #teor médio das 35 amostras
> mi0<- 2      #teor máximo a ser aceito ou rejeitado
> sigma<-0.25  #desvio padrão
> n=35         #tamanho da amostra
> z<-(teor-mi0)/(sigma/sqrt(n))
> z            #teste estatístico
[1] 2.366432
> alfa<-0.05
> z.alfa<-qnorm(1-alfa)
> z.alfa      #valor crítico
[1] 1.644854
```

Como $z>=z_\alpha$ a hipótese de o teor ser <= 2% está rejeitada a nível de significância de 0.05 .

Agora veremos a hipótese nula de um **teste das duas caudas da média de população** pode ser expressa por:

$$\mu = \mu_0$$

onde μ_0 o limite inferior hipotético da média μ de uma população verdadeira.

Então a hipótese nula do teste das duas caudas é para ser rejeitada se $z>=z_{\alpha/2}$ ou $z<=-z_{\alpha/2}$ onde $z_{\alpha/2}$ é o percentil $100(1-\alpha/2)$ de uma distribuição normal padrão.

Ex.: Numa mina de ouro foi definido um teor médio de 15.4 g/ton de ouro. Abrindo uma nova frente de lavra obtivemos 35 amostras que geraram um teor médio de 14.6 g/ton de ouro. Sabemos

que o desvio padrão da reserva é de 2.5 g/ton. A 0.05 de nível de significância podemos rejeitar a hipótese de que o teor encontrado não difere do teor da reserva?

```
> teor<-14.6    #teor médio das 35 amostras
> mi0<- 15.4    #teor a ser aceito ou rejeitado
> sigma<-2.5    #desvio padrão
> n=35          #tamanho da amostra
> z<-(teor-mi0)/(sigma/sqrt(n))
> z             #teste estatístico
[1] -1.893146
> alpha<-0.05
> z.meio_alfa<-qnorm(1-alfa/2)
> c(-z.meio_alfa, z.meio_alfa) # faixa crítica
[1] -1.959964  1.959964
```

Como z está entre z_α e $-z_\alpha$ não podemos rejeitar a hipótese e o teor encontrado na nova frente é válido.

Caso não saibamos a variância da população original podemos usar os testes acima ***usando o desvio padrão da sua amostra ao invés do da população*** e usar Student T (função qt(1-alfa, df=n-1)) no lugar de Normal (função qnorm(1-alfa)) para determinar a faixa ou valor crítico do teste.

A hipótese nula do **teste de cauda inferior da proporção populacional** pode ser expressa por:

$$p \geq p_0$$

onde p_0 seria o limite inferior hipotético da proporção populacional p.

Vamos definir o teste estatístico z em termos da proporção populacional e do tamanho da amostra.

$$z = \frac{\bar{p} - p_0}{\sqrt{p_0(1-p_0)/n}}$$

Então a hipótese nula do teste da cauda inferior é para ser rejeitada se $z<=-z_\alpha$ onde z_α é o percentil 100(1-α) de uma distribuição normal padrão.

Ex.: Em uma campanha de exploração mineral tivemos 60% de furos bem-sucedidos no ano passado. Aleatoriamente escolhemos 148 furos feitos esse ano e 85 foram positivos. A um nível de significância de 0.05 podemos rejeitar que atingimos mais de 60% de sucesso esse ano?

```
> pp<- 85/148    #proporção da amostra
> p0<- 0.6       # valor hipotético
> n<- 148        #tamanho da amostra
> z<- (pp-p0)/sqrt(p0*(1-p0)/n)
> z              # teste estatístico
[1] -0.6375983
> alfa<- 0.05
> z.alfa<-qnorm(1-alfa)
> -z.alfa       # valor crítico
[1] -1.644854
```

O valor de z não é menor que o de z_0, assim não rejeitamos a hipótese e o sucesso nos furos está acima de 60% este ano usando um nível de significância de 0.05.

A hipótese nula do **teste de cauda superior da proporção populacional** pode ser expressa por:

$$p \leq p_0$$

onde p_0 seria o limite inferior hipotético da proporção populacional p.

Então a hipótese nula do teste da cauda inferior é para ser rejeitada se $z>=z_\alpha$ onde z_α é o percentil $100(1-\alpha)$ de uma distribuição normal padrão.

Ex.: Numa sondagem de delineação de um depósito de carvão a perda de testemunho não deve ser maior que 12% dos furos. Foram selecionados aleatoriamente 214 furos e nestes observou-se perda em 30 furos. Para um nível de 0.05 podemos rejeitar termos menos de 12% de perda?

```
> pp<- 30/214    #proporção da amostra
> p0<- 0.12       # valor hipotético
> n<- 214        #tamanho da amostra
> z<- (pp-p0)/sqrt(p0*(1-p0)/n)
> z              # teste estatístico
[1] 0.908751
> alfa<- 0.05
> z.alfa<-qnorm(1-alfa)
> z.alfa       # valor crítico
[1] 1.644854
```

Z não é maior que z_0 , não rejeitamos a hipótese e assim a perda de amostra em furos é menor que 12%.

A hipótese nula do **teste das duas caudas da proporção populacional** pode ser expressa por:

$$p = p_0$$

onde p_0 seria o limite inferior hipotético da proporção populacional p.

Então a hipótese nula do teste das duas caudas é para ser rejeitada se $z<=-z_\alpha$ ou $z>=z_C$ onde z_α é o percentil $100(1-\alpha)$ de uma distribuição normal padrão.

Ex.: Jogando cara ou coroa 20 vezes obtivemos 12 caras. Com um nível de significância de 0.05 pode-se rejeitar a hipótese de que o cara e coroa é justo?

```
> pp<- 12/20    #proporção da amostra
> p0<- 0.5       # valor hipotético
> n<- 20        #tamanho da amostra
> z<- (pp-p0)/sqrt(p0*(1-p0)/n)
> z              # teste estatístico
[1] 0.8944272
> alfa<- 0.05
> z.meio_alfa<-qnorm(1-alfa/2)
> c(-z.meio_alpha, z.meio_alpha)
[1] -1.959964  1.959964
```

O resultado está entre $-z_\alpha$ e z_α então não rejeitamos a hipótese de que o cara e coroa é justo a um nível de significância de 0.05.

1.12.6 - Erro do tipo II

Num **teste de cauda inferior da média de população** a hipótese nula diz que a média da população μ é maior que um valor hipotético μ_0 .

$$\mu \geq \mu_0$$

Um **erro tipo II** ocorre se o teste hipótese, baseado em uma amostragem aleatória, falha em ser rejeitado mesmo quando a verdadeira média da população μ é em fato menor que μ_0.

Assumimos que a população tem **variância σ^2 conhecida**. Pelo Teorema do Limite Central, a população de todas as médias das amostras possíveis quando aumentamos o tamanho da amostra n, se aproxima de uma distribuição normal. Assim nós podemos calcular a faixa de amostragem média para qual a hipótese nula não será rejeitada, e obter uma estimativa da probabilidade de erro do tipo II.

Ex.: Foi dito que o teor médio de um jazimento é acima de 10%. Assumindo um desvio padrão de 0.12% e um teor médio atual de 9.95%. Num nível de significância de 0.05, qual seria a probabilidade de termos um erro tipo II para uma amostragem aleatória de 30 pontos?

```
> n<- 30                # tamanho da amostragem
> sigma<- 0.12          # desvio padrão da população
> sem<- sigma/sqrt(n)   # erro padrão
> sem
[1] 0.0219089
> alfa<- 0.05           # nível de significância
> mi0<- 10              #limite inferior hipotético
> q<- qnorm(alfa,mean=mi0,sd=sem)
> q
[1] 9.963963
> ma<- 9.95             # média assumida
> pnorm(q,mean=ma,sd=sem,lower.tail=FALSE)
[1] 0.261957
```

Se a amostragem de 30 pontos com média de teor 9.95% e com um desvio padrão da população dado de 0.12%, então a probabilidade de erro tipo II testando a hipótese nula μ>=10% a 0.05 significância é 26.2% e a força do teste da hipótese e 73.8%.

Num **teste de cauda superior da média de população** a hipótese nula diz que a média da população μ é maior que um valor hipotético μ_0 .

$$\mu \leq \mu_0$$

Um **erro tipo II** ocorre se o teste hipótese, baseado em uma amostragem aleatória, falha em ser rejeitado mesmo quando a verdadeira média da população μ é em fato maior que μ_0.

Ex.: Em uma frente de lavra a porcentagem de determinado contaminante de minério foi estipulada em ser no máximo 2%. Assumindo que a média atual é de 2.09% deste contaminante e assumindo um desvio padrão da população original de 0.25%. A 0.05 de nível de significância qual é a probabilidade de termos erro Tipo II para uma amostra de 35 pontos?

```
> n<- 35                # tamanho da amostragem
> sigma<- 0.25          # desvio padrão da população
> sem<-  sigma/sqrt(n)  # erro padrão
> sem
[1] 0.04225771
> alfa<- 0.05           # nível de significância
> mi0<- 2                # limite superior hipotético
```

```
> q<- qnorm(alfa,mean=mi0,sd=sem,lower.tail=FALSE)
> q
[1] 2.069508
> ma<- 2.09              # média assumida
> pnorm(q,mean=ma,sd=sem)
[1] 0.3138612
```

Para uma amostra de 35 pontos com média de 2.09% de contaminante e com desvio padrão da população original de 0.25%, a probabilidade de erro tipo II testando a hipótese nula μ<=2 a 0.05 de nível de significância é 31.4% e a força do teste de hipótese é de 68.6%.

Num **teste de duas caudas da média de população** a hipótese nula diz que a média da população μ é maior que um valor hipotético μ_0 .

$$\mu = \mu_0$$

Um **erro tipo II** ocorre se o teste hipótese, baseado em uma amostragem aleatória, falha em ser rejeitado mesmo quando a verdadeira média da população μ é em fato diferente de μ_0.

Ex.: Numa mina de ouro foi definido um teor médio de 15.4 g/ton de ouro. Abrindo uma nova frente de lavra amostramos um teor médio de 14.6 g/ton de ouro, Sabemos que o desvio padrão da reserva é de 2.5 g/ton. A 0.05 de nível de significância qual a probabilidade de termos um erro tipo II para um tamanho de amostra de 35?

```
> n<- 35                 # tamanho da amostragem
> sigma<- 2.5            # desvio padrão da população
> sem<- sigma/sqrt(n)    # erro padrão
> sem
[1] 0.4225771
> alfa<- 0.05           # nível de significância
> mi0<- 15.4            # média hipotética
> I <- c(alfa/2, 1-alfa/2)
> q=qnorm(I,mean=mi0,sd=sem)
> q
[1] 14.57176 16.22824
> ma<- 14.6             # média assumida
> p<- pnorm(q,mean=ma,sd=sem)
> p
[1] 0.4733631 0.9999417
> diff(p)             # Diferença entre os dois pontos extremos
[1] 0.5265786
```

Para 35 amostras com média 14.6 g/ton e desvio padrão da população original de 2.5 g/ton a probabilidade de erro tipo II testando a hipótese nula μ=14.6 g/ton a um nível de significância de 0.05 é 52.6% e a força do teste de hipótese é 47.4%.

Quando não temos informação sobre a variância da população **podemos chegar à probabilidade de erro tipo II usando o desvio padrão da amostra** e usar distribuição Student T com n-1 grau de liberdade conforme:

$$\frac{\bar{x} - \mu}{s/\sqrt{n}}$$

Isso nos permitirá calcular a faixa das médias das amostras para as quais a hipótese nula não será rejeitada e obter a probabilidade do erro tipo II conforme mostraremos abaixo:

Teste de cauda inferior da média de população (variância desconhecida)

Foi dito que o teor médio de um jazimento é acima de 10%. Numa amostragem de 30 pontos obtivemos um desvio padrão de 0.125% e um teor médio atual de 9.95%. Num nível de significância de 0.05, qual seria a probabilidade de termos um erro tipo II?

```
> n <-  30                # tamanho da amostra
> s <-  0.125             # desvio padrão da amostra
> SE <-  s/sqrt(n)        # estimativa do erro padrão
> SE
[1] 0.02282177
> alfa <-  0.05           # nível de significância
> mi0 <-  10              # limite inferior hipotético
> q <-  mi0+qt(alfa,df=n-1)*SE
> q
[1] 9.961223
> mi <-  9.95             # média assumida atual
> pt((q-mi)/SE,df=(n-1),lower.tail=FALSE)
[1] 0.3132943
```

Neste caso a probabilidade do erro tipo II testando a hipótese nula $\mu>=10$ a 0.05 nível de significância é de 31.3% e a força do teste de hipótese é de 68.7%.

Teste de cauda superior da média de população (variância desconhecida)

Em uma frente de lavra a porcentagem de determinado contaminante de minério foi estipulada em ser no máximo 2%. Numa amostragem de 35 pontos a média atual é de 2.09% deste contaminante e temos na amostra um desvio padrão de 0.3%. A 0.05 de nível de significância qual seria a probabilidade de termos um erro tipo II?

```
> n <- 35                 #tamanho da amostra
> s <- 0.3                # desvio padrão da amostra
> SE <- s/sqrt(n)         # estimativa do erro padrão
> SE
[1] 0.05070926
> alfa <- 0.05           # nível de significância
> mi0 <- 2               # limite superior hipotético
> q <- mi0+qt(alfa,df=n-1,lower.tail=FALSE)*SE
> q
[1] 2.085746
> mi <- 2.09              # média assumida atual
> pt((q-mi)/SE,df=(n-1))
[1] 0.4668141
```

Neste caso a probabilidade do erro tipo II testando a hipótese nula $\mu<=2$ a 0.05 nível de significância é de 46.7% e a força do teste de hipótese é de 53.3%.

Teste das duas caudas da média de população (variância desconhecida)

Numa mina de ouro foi definido um teor médio de 15.4 g/ton de ouro. Abrindo uma nova frente de lavra em 35 amostras um teor médio de 14.6 g/ton de ouro e desvio padrão da amostra é de 2.5 g/ton. A 0.05 de nível de significância qual seria a probabilidade de termos um erro tipo II?

```
> n <-  35                #tamanho da amostra
> s <-  2.5               # desvio padrão da amostra
> SE <-  s/sqrt(n)        # estimativa do erro padrão
> SE
[1] 0.4225771
```

```
> alfa <-  0.05             # nível de significância
> mi0 <-  15.4              # média hipotética
> I <-  c(alfa/2,1-alfa/2)
> q <-  mi0+qt(I,df=n-1)*SE
> q
[1] 14.54122 16.25878
> mi <-  14.6
> p <-  pt((q-mi)/SE,df=n-1)
> p
[1] 0.4450963 0.9997996
> diff(p)         #diferença dos extremos
[1] 0.5547033
```

Neste caso a probabilidade do erro tipo II testando a hipótese nula μ=15.4 a 0.05 nível de significância é de 55.5% e a força do teste de hipótese é de 44.5%.

1.12.7 - Interferência entre duas populações

Duas amostras de dados são pareadas se elas vem de repetidas observações de um mesmo indivíduo. Aqui nós assumimos que o dado das populações obedecem uma distribuição normal. Usando o teste T pareado podemos obter uma estimativa de intervalo da diferença das médias das populações.

Ex.: s1 e s31 são imagens raster da banda 2 (sentinel2) de um mesmo local adquiridas em dias diferentes (dia 01/07/2018 e dia 31/07/2018). Extraímos os primeiros 10000 valores de cada raster e efetuamos o teste T pareado:

```
> xs1<-extract(s1,c(1:10000))
> xs31<-extract(s31,c(1:10000))
> t.test(xs1,xs31,paired=TRUE)

        Paired t-test

data:  xs1 and xs31
t = 36.935, df = 9999, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 101.0702 112.3996
sample estimates:
mean of the differences
               106.7349
```

O intervalo com 95% confiança da diferença na média das imagens dos dias 1 e 31 é o intervalo entre 101.0700 e 112.3996.

Duas amostras de dados são consideradas independentes se elas vem de populações não relacionáveis e as amostras não afetam cada uma delas. Aqui nós assumimos que o dado das populações obedecem uma distribuição normal. Usando o teste T não pareado nós podemos obter uma estimativa de intervalo da diferença entre as medias das duas populações.

Ex.: Suponhamos que temos dois depósitos minerais 0 e 1 em diferentes áreas e resultados de vários furos dos dois depósitos são apresentados conforme abaixo:

Furo	Espessura(m)	Teor/metro (%/m)	Depósito
BZ2	4.5	15	0
BZ3	4.1	14.8	0
BZ4	3.9	15.7	0
BZ5	5.5	12.6	0

BZ6	4.3	14.5	0
BZ7	2.9	17.4	0
BZ8	3.5	14.6	0
BZ9	4.0	12.5	0
BZ1	5.6	11.4	0
BZ10	3.4	15.7	0
AT2	2.2	22.0	1
AT3	2.2	28.3	1
AT4	1.8	26.5	1
AT5	3.6	20.7	1
AT6	2.2	22.3	1
AT7	1.5	25.4	1
AT8	2.7	22.4	1
AT9	2.2	24.3	1
AT1	3.5	22.2	1
AT10	1.2	23.1	1

Assumindo que os resultados seguem uma distribuição normal, encontre a estimativa do intervalo, 95% de confiança, da diferença entre a média dos teores entre os dois depósitos:

```
> depo$Depósito
 [1] 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
> depo$Teor
 [1] 15.0 14.8 15.7 12.6 14.5 17.4 14.6 12.5 11.4 15.7 22.0 28.3 26.5 20.7 22.3
[16] 25.4 22.4 24.3 22.2 23.1
> D<-depo$Depósito == 0
> teor.0<-depo[D,]$Teor      # teor do deposito 0
> teor.1<-depo[!D,]$Teor     # teor do deposito 1
> teor.0
 [1] 15.0 14.8 15.7 12.6 14.5 17.4 14.6 12.5 11.4 15.7
> teor.1
 [1] 22.0 28.3 26.5 20.7 22.3 25.4 22.4 24.3 22.2 23.1
> t.test(teor.0,teor.1)

        Welch Two Sample t-test

data:  teor.0 and teor.1
t = -9.9143, df = 16.754, p-value = 2.015e-08
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -11.281302  -7.318698
sample estimates:
mean of x mean of y
    14.42     23.72
```

Nestes resultados, a média do depósito 0 é 14.43%/m e a média do depósito 1 é 23.72%/m. O intervalo da diferença das médias de teores variam entre 7.32%/m e 11.28%/m, a 95% confiança.

Um levantamento em duas populações distintas produzirá resultados diferentes. É frequentemente necessário comparar a proporção da resposta do levantamento entre as duas populações. Aqui assumimos que os dados das populações seguem uma distribuição normal.

Ex.:Crianças de uma cidade australiana são classificadas por etnia (A,N), gênero(M,F), e outras informações. Assumindo que os dados seguem a uma distribuição normal, a 95% de confiança qual é a proporção de meninas entre as populações aborígenes(A) e não aborígenes(N).

```
> library(MASS)
> table(quine$Eth, quine$Sex)
     F  M
```

```
  A 38 31
  N 42 35
> prop.test(table(quine$Eth, quine$Sex), correct=FALSE)

        2-sample test for equality of proportions without continuity
        correction

data:  table(quine$Eth, quine$Sex)
X-squared = 0.0040803, df = 1, p-value = 0.9491
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.1564218  0.1669620
sample estimates:
   prop 1    prop 2
0.5507246 0.5454545
```

A proporção de meninas entre as populações aborígenes(A) e não aborígenes(N), a 95% confiança, está entre -15.6% e 16.7%.

1.12.8 - Testes de adequação (aderência)

Uma população é chamada multinomial se seus dados forem em categorias e pertencerem a uma coleção de classes discretas não sobrepostas.

A hipótese nula do **teste de aderência para distribuição multinomial** é que a frequência observada f, é igual a uma contagem esperada e, em cada categoria. Ela será rejeitada se 0 valor-p do teste chi quadrado é menor que o valor de significância fornecido α.

$$\chi^2 = \sum_i \frac{(f_i - e_i)^2}{e_i}$$

Ex.: No levantamento survey temos a coluna Smoke(fumante) com as opções de resposta, “Heavy” (pesado),”Regul” (regular), “Occas” (ocasional) e “Never” (nunca). Este dado é multinomial. Supondo uma estatística no campus mostrada abaixo, demonstre se o dado em survey confirma estatística em um nível de significância de 0.05.

Heavy Never Occas Regul
4.5% 79.5% 8.5% 7.5%

```
> library(MASS)
> levels(survey$Smoke)
[1] "Heavy" "Never" "Occas" "Regul"
> smoke.freq<-table(survey$Smoke)
> smoke.freq                          # resultado da tabela survey
Heavy Never Occas Regul
   11   189    19    17
> smoke.prob<-c(0.045,0.795,0.085,0.075) #dados da estatística
> chisq.test(smoke.freq,p=smoke.prob)    #test chi-quadrado

        Chi-squared test for given probabilities

data:  smoke.freq
X-squared = 0.10744, df = 3, p-value = 0.9909
```

Como o valor-p 0.9909 é maior que o valor de significância 0.05 nós não rejeitamos a hipótese nula de que os dados de survey suportam os resultados da estatística no campus.

Duas variáveis x e y são chamadas **independentes** se a distribuição de probabilidade de uma variável não é afetada pela presença da outra.

Assumindo que f_γ é a contagem da frequência observada dos eventos pertencentes a ambas categorias i-ésima de x e j-ésima de y. Também assumindo que e_γ ser a contagem correspondente esperada de x e y e são independentes. A hipótese nula do **teste de independência** será rejeitada se o valor-p do seguinte teste chi quadrado é menor que o valor de significância fornecido α.

$$\chi^2 = \sum_{i,\ j} \frac{(f_{ij} - e_{ij})^2}{e_{ij}}$$

Ex.: Vamos agora testar a independência usando os dados survey checando o hábito de fumar com o hábito de se exercitar em nível de significância de 0.05.

```
> library(MASS)
> tbl <- table(survey$Smoke, survey$Exer)
> tbl

       Freq None Some
  Heavy    7    1    3
  Never   87   18   84
  Occas   12    3    4
  Regul    9    1    7
>  ctbl<-cbind(tbl[,"Freq"],tbl[,"None"]+tbl[,"Some"])  #combinamos  os  valores
pequenos de None e Some
> ctbl
      [,1] [,2]
Heavy    7    4
Never   87  102
Occas   12    7
Regul    9    8
> chisq.test(ctbl)

        Pearson's Chi-squared test

data:  ctbl
X-squared = 3.2328, df = 3, p-value = 0.3571
```

Como o valor-p de 0.357 é maior que 0.05 de significância, nós não rejeitamos a hipótese de que os hábitos de fumar e se exercitar são independentes.

1.12.9 - Análise de Variância (ANOVA)

Análise da variância (ANOVA) é dependente do design.

No **design completamente aleatório**, existe somente um fator primário a ser considerado no experimento. Os indivíduos testados são designados para níveis de tratamento do fator primário aleatoriamente.

Ex.: Três elementos de um depósito são testados. 18 pontos de amostragem foram escolhidos aleatoriamente. Para estar de acordo com o **design completamente aleatório,** os resultados do primeiro elemento vieram de 6 pontos aleatoriamente escolhidos, o segundo elemento de outros 6 e mesmo para o terceiro elemento. A uma nível de significância de 0.05, teste se a média do resultado para os 3 elementos são iguais.

Copie e salve como anov1.csv os dados abaixo:

```
el1, el2, el3
22,52,16
42,33,24
44,8,19
52,57,18
45,43,34
37,32,39
```

```
> df1<-read.csv('anov1.csv')
> df1
  el1 el2 el3
1  22  52  16
2  42  33  24
3  44   8  19
4  52  47  18
5  45  43  34
6  37  32  39
> r<-c(t(as.matrix(df1))) #concatenando df1 como um vetor
> r
 [1] 22 52 16 42 33 24 44  8 19 52 47 18 45 43 34 37 32 39
> # designando as variáveis para o tratamento
> f<- c('el1','el2','el3')    #níveis do tratamento
> k<-3                        # número de níveis de tratamento
> n<-6                        # observações por tratamento
> tm<-gl(k,1,n*k,factor(f))   # cria vetor corresp. aos níveis de tratam.
> tm
 [1] el1 el2 el3 el1 el2 el3 el1 el2 el3 el1 el2 el3 el1 el2 el3 el1 el2 el3
Levels: el1 el2 el3
> ANOVA<-aov(r ~ tm)
> summary(ANOVA)
            Df Sum Sq Mean Sq F value Pr(>F)
tm           2  745.4   372.7   2.541  0.112
Residuals   15 2200.2   146.7
```

Como o valor-p de 0.112 é maior que 0.05 de nível de significância, nós não rejeitamos a hipótese nula de que as médias dos elementos são iguais.

Em um **design aleatório em blocos**, existe somente um fator primário em consideração no experimento. Indivíduos similares do teste são agrupados em **blocos**. Cada bloco é testado contra todos os níveis de tratamento do fator primário em ordem aleatória com a intenção de eliminar possível influência de fatores externos.

Ex.: Três elementos de um depósito são testados. 6 pontos de amostragem foram escolhidos aleatoriamente. Para estar de acordo com o **design aleatório em blocos** cada ponto amostrará os três elementos. A um nível de significância de 0.05, teste se a média do resultado para os 3 elementos são iguais.

Copie e salve como anov2.csv os dados abaixo:

```
el1, el2, el3
31,27,24
31,28,31
45,29,46
21,18,48
42,36,46
32,17,40
```

```
> df1<-read.csv('anov2.csv')
> df1
  el1 el2 el3
1  31  27  24
2  31  28  31
3  45  29  46
4  21  18  48
5  42  36  46
6  32  17  40
> r<-c(t(as.matrix(df1))) #concatenando df1 como um vetor
> r
 [1] 31 27 24 31 28 31 45 29 46 21 18 48 42 36 46 32 17 40
> # designando as variáveis para o tratamento
> f<- c('el1','el2','el3')    #níveis do tratamento
> k<-3                        # número de níveis de tratamento
> n<-6                        # observações por tratamento
> tm<-gl(k,1,n*k,factor(f))   # cria vetor correp. aos níveis de tratam.
> tm
 [1] el1 el2 el3 el1 el2 el3 el1 el2 el3 el1 el2 el3 el1 el2 el3 el1 el2 el3
Levels: el1 el2 el3
> blk<-gl(n,k,k*n)           # fator bloco
> blk
 [1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5 6 6 6
Levels: 1 2 3 4 5 6
> ANOVA<- aov(r ~ tm + blk)
> summary(ANOVA)
            Df Sum Sq Mean Sq F value Pr(>F)
tm           2  538.8  269.39   4.959 0.0319 *
blk          5  559.8  111.96   2.061 0.1547
Residuals   10  543.2   54.32
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

Como o valor-p de 0.032 é menor que o nível de significância 0.05, nós rejeitamos a hipótese nula de que as médias dos resultados dos elementos são iguais.

No **design fatorial** existem mais de um fator em consideração no experimento. Os indivíduos testados são designados para o tratamento de cada fator em combinações aleatórias.

Ex.: Três elementos de dois depósitos similares são testados. 12 pontos de amostragem foram escolhidos aleatoriamente em cada depósito. Para estar de acordo com o **design fatorial** os resultados dos elementos vieram de 4 pontos aleatoriamente escolhidos, o segundo elemento de outros 4 e mesmo para o terceiro elemento. A um nível de significância de 0.05, teste se a média do resultado para os 3 elementos são iguais e conclua se as médias entre os dois depósitos diferem.

Copie e salve como anov3.csv os dados abaixo:

```
el1,el2,el3
A1,25,39,36
A2,36,42,24
A3,31,39,28
A4,26,35,29
B1,51,43,42
B2,47,39,36
B3,47,53,32
B4,52,46,33
```

```
> df1<-read.csv('anov3.csv')
> df1
   el1 el2 el3
A1  25  39  36
A2  36  42  24
A3  31  39  28
A4  26  35  29
B1  51  43  42
B2  47  39  36
B3  47  53  32
B4  52  46  33
> r<-c(t(as.matrix(df1))) #concatenando df1 como um vetor
> r
 [1] 25 39 36 36 42 24 31 39 28 26 35 29 51 43 42 47 39 36 47 53 32 52 46 33
> # designando as variáveis para o tratamento
> f1<- c('el1','el2','el3')    #níveis do tratamento
> f2<- c('depA','depB')        #níveis do tratamento
> k1<-length(f1)               # número de níveis de tratamento fator 1
> k2<-length(f2)               # número fator 2
> n <-4
> tm1<-gl(k1,1,n*k1*k2,factor(f1))
> tm1
 [1] el1 el2 el3 el1 el2 el3 el1 el2 el3 el1 el2 el3 el1 el2 el3 el1 el2 el3 el1
[20] el2 el3 el1 el2 el3
Levels: el1 el2 el3
> tm2<-gl(k2,n*k1,n*k1*k2,factor(f2))
> tm2
 [1] depA depA depA depA depA depA depA depA depA depA depA depA depB depB depB
[16] depB depB depB depB depB depB depB depB depB
Levels: depA depB
> ANOVA<-aov(r ~ tm1*tm2)
> summary(ANOVA)
            Df Sum Sq Mean Sq F value   Pr(>F)
tm1          2  385.1   192.5   9.554  0.00149 **
tm2          1  715.0   715.0  35.481 1.23e-05 ***
tm1:tm2      2  234.1   117.0   5.808  0.01132 *
Residuals   18  362.8    20.2
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Como o valor-p de 0.0015 é menor que o nível de significância 0.05, nós rejeitamos a hipótese nula de que as médias dos resultados dos elementos são iguais. Além do mais o valor de 1.23e-05 para a comparação depósito A e B é também bem menor que o nível de significância 0.05 mostrando que existe uma diferença geral entre os valores dos elementos.

1.12.10 - Regressão linear simples

Se escolhermos os parâmetros α e β no **modelo de regressão linear simples** para minimizar o erro das soma dos quadrados ε, nós teremos então de ter a **equação estimada de regressão simples**. Ela nos permitirá calcular valores ajustados de y em função do valor x:

$$\hat{y} = a + bx$$

Ex.: Vamos usar a base de dados ‘faithful’ e estimar a duração da próxima erupção com base num tempo de espera de 80 minutos:

```
> erupção.ml<-lm(eruptions ~ waiting, data=faithful)
> coeficiente<- coefficients(erupção.ml)
> coeficiente
(Intercept)     waiting
-1.87401599  0.07562795
> #agora vamos ajustar o duração da erupção usando a equação de regressão
> espera<-80            # o tempo de espera
> duração<- coeficiente[1] + coeficiente[2]*espera
> duração
(Intercept)
    4.17622
```

Com base no modelo de regressão linear simples, se o tempo de espera for de 80 minutos nós esperaremos que a próxima erupção dure 4.17622 minutos.

O **coeficiente de determinação** de um modelo de regressão linear é o quociente das variâncias dos valores ajustados e valores observados de uma variável dependente. Se denotarmos y_i como os valores observados de uma variável dependente, $\tilde{y}$ como sua média e $\hat{y}$ como o valor ajustado, então o coeficiente de determinação será:

$$r^2 = \frac{\sum(\hat{y_i} - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

Ex.: Encontre o **coeficiente de determinação** para a base de dados 'faithful':

```
> erupção.ml<-lm(eruptions ~ waiting, data=faithful)
> summary(erupção.ml)$r.squared
[1] 0.8114608
```

O coeficiente de determinação de 'faithful' é 0.8114608.

Assumindo que o erro ε no modelo de regressão linear é independente de x, e é normalmente distribuído com média zero e variância constante. Nós podemos definir se existe qualquer **relação de significância** entre x e y testando a hipótese nula que β=0.

```
> erupção.ml<-lm(eruptions ~ waiting, data=faithful)
> summary(erupção.ml)

Call:
lm(formula = eruptions ~ waiting, data = faithful)

Residuals:
     Min       1Q   Median       3Q      Max
-1.29917 -0.37689  0.03508  0.34909  1.19329

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.874016   0.160143  -11.70   <2e-16 ***
waiting      0.075628   0.002219   34.09   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4965 on 270 degrees of freedom
Multiple R-squared:  0.8115,    Adjusted R-squared:  0.8108
F-statistic:  1162 on 1 and 270 DF,  p-value: < 2.2e-16
```

Como o valor-p é muito menor que 0.05, nós rejeitamos a hipótese nula que β=0. Assim existindo uma relação de significância entre as variáveis no modelo de regressão linear da base de dados 'faithful'.

Assumindo que o erro ε no modelo de regressão linear é independente de x, e é normalmente distribuído com média zero e variância constante. Para um dado valor x, a estimativa do intervalo para a média da variável dependente, y, é chamado de **intervalo de confiança.**

Ex.: Para a base de dados 'faithful' , determine **intervalo com 95% confiança** para a media da duração da erupção para uma espera de 80 minutos.

```
> attach(faithful)
> erupção.ml<-lm(eruptions ~ waiting)
> novodado<- data.frame(waiting=80)
> predict(erupção.ml,novodado, interval="confidence")
       fit      lwr      upr
1 4.17622 4.104848 4.247592
> detach(faithful)
```

Para 95% intervalo de confiança, a média da duração da erupção para um tempo de espera de 80 minutos está entre 4.1048 e 4.2476.

Assumindo que o erro ε no modelo de regressão linear é independente de x, e é normalmente distribuído com média zero e variância constante. Para um dado valor x, a estimativa do intervalo da variável dependente, y, é chamado de **intervalo de predição.**

Ex.: Para a base de dados 'faithful' , determine **intervalo com 95% predição** da duração da erupção para uma espera de 80 minutos.

```
> attach(faithful)
> erupção.ml<-lm(eruptions ~ waiting)
> novodado<- data.frame(waiting=80)
> predict(erupção.ml,novodado, interval="predict")
       fit      lwr      upr
1 4.17622 3.196089 5.156351
> detach(faithful)
```

Para 95% intervalo de confiança, a duração da erupção para um tempo de espera de 80 minutos está entre 3.1961 e 5.1564.

O dado **residual** de um modelo de regressão linear simples é a diferença entre o dado observado da variável dependente y e o valor ajustado ŷ.

$$Residual = y - \hat{y}$$

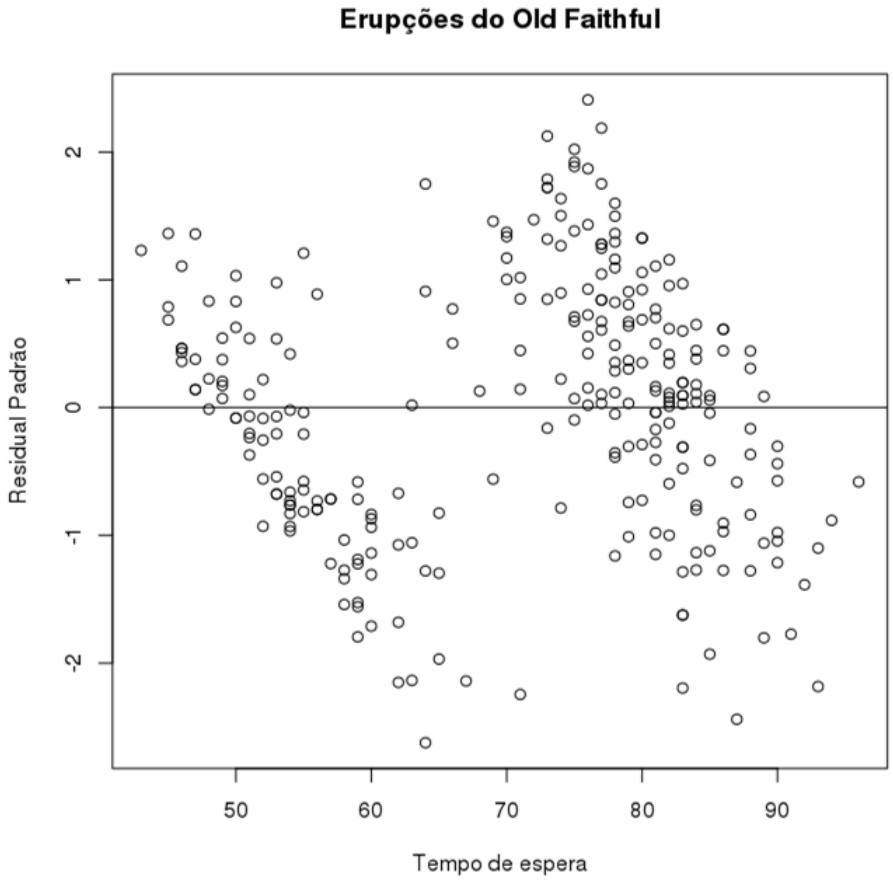
Ex,: Plote o residual do modelo de regressão linear simples da base de dados 'faithful' contra a variável dependente 'waiting' (espera).

```
> erupção.ml<-lm(eruptions ~ waiting, data=faithful)
> erupção.residual<-resid(erupção.ml)
> plot(faithful$waiting,erupção.residual,
+ ylab='Residual',xlab='Tempo de espera',
+ main='Erupções do Old Faithful')
> abline(0,0)
```

O **residual padrão** é o residual dividido por seu desvio padrão.

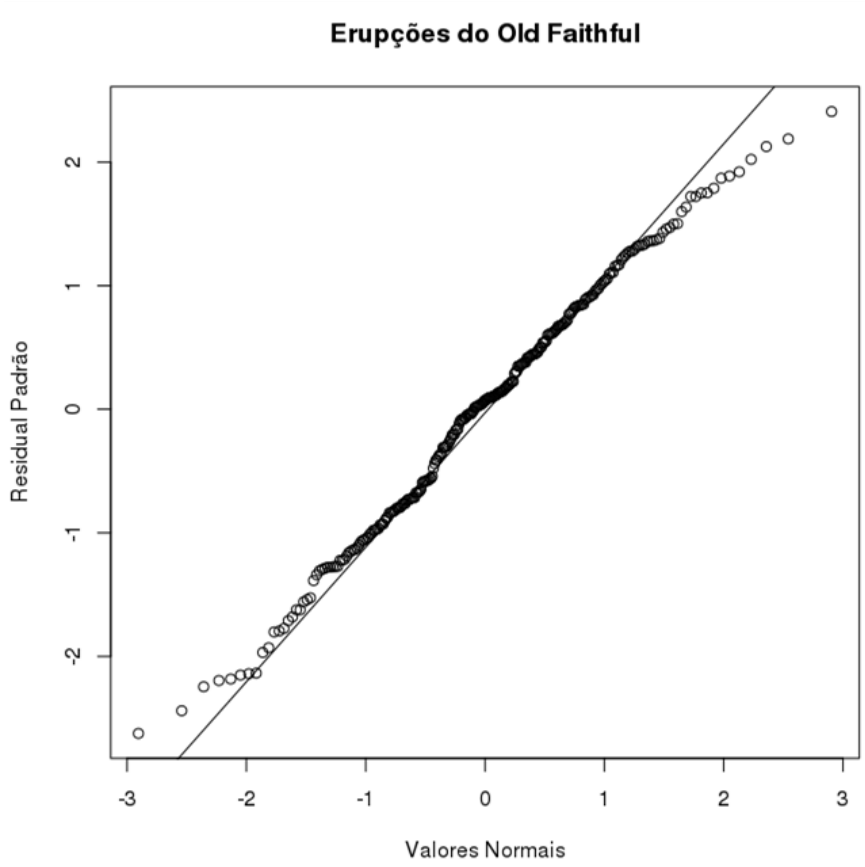
$$Standardized\ Residual\ i = \frac{Residual\ i}{Standard\ Deviation\ of\ Residual\ i}$$

```
> erupção.ml<-lm(eruptions ~ waiting, data=faithful)
> erupção.respadrão<-rstandard(erupção.ml)
> plot(faithful$waiting,erupção.respadrão,
+ ylab='Residual Padrão',xlab='Tempo de espera',
+ main='Erupções do Old Faithful')
> abline(0,0)
```



O **gráfico de probabilidade normal** é uma ferramenta gráfica para comparar uma base de dados com a distribuição normal. Nós podemos usar ele com o residual padrão do modelo de regressão linear e ver se o erro ε está de fato distribuído normalmente.

```
> erupção.ml<-lm(eruptions ~ waiting, data=faithful)
> erupção.respadrão<-rstandard(erupção.ml)
> qqnorm(erupção.respadrão,
+ ylab='Residual Padrão',xlab='Valores Normais',
+ main='Erupções do Old Faithful')
> qqline(erupção.respadrão)
```



Concluímos assim nossa revisão de estatística e como usamos ela com R.

1.13 – Informações finais

Na maioria dos casos se fará necessário instalar e usar um pacote de funções (package) adicional para trabalharmos em R.

Para instalar um pacote usamos (fazemos isso uma vez só)

```
> install.packages('nome_do_pacote')
```

Onde nome do pacote é substituído pelo nome real do pacote:

```
> install.packages('raster')
```

Carregamos o pacote instalado assim:

```
> library(raster)
```

Apenas demos uma pincelada no que diz respeito a R mas que permitirá a você desenvolver o seu conhecimento.