

Modelos Matemáticos e Aplicações (2024/2025)

Revisão dos fundamentos de Probabilidade e Estatística com
apoio do *software* 

Módulo I – Abril 2025

Manuela Neves

ISA/ULisboa

- Breve introdução ao ambiente .
- Os principais objectos do . Operações com números, vectores e matrizes.
- Estrutura e manipulação de dados no . Algumas funções estatísticas.
- Estatística Descritiva e Análise exploratória de dados com apoio do .
- Os principais modelos de probabilidade discretos e contínuos.
- Introdução à Teoria de Estimação. Noção de estimador e estimativa.
- Principais métodos de estimação: o método dos momentos e o método da máxima verosimilhança.
- Introdução à Inferência Estatística paramétrica e não paramétrica.
- Intervalos de Confiança e Testes de hipóteses.
- Resolução de exercícios de aplicação.

- 1 Tutoriais de 
- 2 Referências Bibliográficas
- 3 Breve introdução ao ambiente 
- 4 Estrutura e manipulação de dados
 - Objectos no  - Vector, Matrix, Factor, List, Data Frame
 - Leitura e escrita de ficheiros
- 5 Funções e Programação em 
- 6 Estatística Descritiva e Análise Exploratória de Dados
 - Alguns conceitos básicos. Indicadores
 - Gráficos no  com `plot()`

- Everitt, B.S. and Hothorn, T. (2006). *A Handbook of Statistical Analyses using  .* Chapman & Hall
- Kerns, G.J. (2010). *Introduction to Probability and Statistics using  .*
Disponível *on-line*
- Monteiro, L.R. (2006). *Introdução à Biometria usando o  .* Disponível *on-line*.
- Torgo, L. (2006). *Introdução à Programação em  .* Disponível *on-line*
- Venables, W. N., Smith D. M. & R Core Team (2024). *An Introduction to  .*
Notes on  : A Programming Environment for Data Analysis and Graphics.
(Version 4.5.0 (2025-04-11) (<http://cran.r-project.org/doc/manuals/R-intro.pdf>))

- Casella, G. and Berger, R.L.(2002). *Statistical Inference*. Wadsworth & Brooks
- Murteira, B. e Antunes, M. (2012). *Probabilidades e Estatística. Voll e II*. McGraw-Hill
- Neves, M. M. (2017). *Introdução à Estatística e à Probabilidade com utilização do R* ISAPress.



- Pestana, D. e Velosa, S. (2008). *Introdução à Probabilidade e à Estatística*. Fundação Calouste Gulbenkian

AULA 1

Breve introdução ao ambiente

- O que é o  ?

- É um conjunto integrado de ferramentas computacionais que permitem **a manipulação e análise estatística de dados, o cálculo numérico e a produção de gráficos.**
- É uma linguagem **interpretada** - os comandos são imediatamente executados.
- É uma linguagem orientada por **objectos** - os dados são armazenados na memória activa do computador na forma de objectos, têm nome e sobre eles aplicam-se acções.

● O que é o ?

- É uma linguagem de programação, permite por isso implementar e tratar novos algoritmos.
- Está em actualização permanente pela introdução constante de novos e diversos procedimentos estatísticos.
- É uma aplicação de distribuição gratuita e de código público disponível em (<http://cran.r-project.org/>) - - aqui existe toda a informação.
- Depois de fazer o *download* da versão adequada ao sistema operativo do computador (por ex. R 4.5.0 e deve escolher-se o sistema operativo). No ambiente de trabalho aparece **Rx64 4.5.0** e clicando estamos prontos para trabalhar.

- **Vantagens do R ?**
 - É mesmo totalmente gratuito.
 - Resulta de uma colaboração internacional de vários investigadores, que mantêm uma rede de discussão na *internet*.
 - É possível corrigir com mais facilidade os *bugs* detectados e até obter ajuda para tentar resolver algo mais específico.
 - Na *homepage* do R, www.r-project.org encontram-se vários tutoriais, em várias linguas.
 - É possível desenvolver um módulo (*package*) para uma aplicação de interesse, torná-la disponível no R e assim poder partilhar conhecimento.
 - O R tem várias *mailing-lists* para uma grande variedade de temas, ver a *homepage* do R.

- **Iniciar e terminar uma sessão de R**
 - Criar uma pasta (ex. **MMA.24.25**) onde irão guardar-se todos os ficheiros de trabalho - ficheiros de dados, ficheiros de resultados, ficheiros do R.
 - Iniciar o R - abre-se uma janela de trabalho.
 - Especificar logo a pasta de trabalho
menu: **File** → **Change dir ...**
 - Os comandos são dados à frente da **prompt >** e são executados após pressionar **“Enter”**.

- **Iniciar e terminar uma sessão de** 
- Para verificar se está na pasta pretendida fazer
`> getwd()`
- Para terminar uma sessão executar `>q()`.
Se pretender guardar o *workspace* (sessão de trabalho que contém o conjunto de objectos de trabalho) fazer **Yes** e fica guardado no ficheiro **.Rdata**

- Todas as funções e comandos do  estão armazenados em (módulos) *packages*.
 - **Para:**
 - ver quais os packages disponíveis `>(.packages())`
 - ver quais os packages instalados `>library()`
 - carregar em memória um package instalado
`>library(nome-package)` ou
menu: Packages → Load Package ...
 - Para instalar um *package* fazer
menu: Packages → Install Package ...
 - Numa sessão de  o conteúdo de um *package* **só fica disponível quando ele é carregado em memória.**

Ajudas no

- Sobre um *package*
`>help(package=datasets)`
- Sobre um conjunto de dados
`>help(InsectSprays)` ou `> ?InsectSprays`
- Sobre uma função, conhecendo o nome
`>help(mean)` ou `> ?mean`
- Para pesquisar uma sequência de caracteres
`>help.search("norm")` ou `>??"norm"`
Indica o *package* e comando onde aparece a sequência
stats::Normal The Normal Distribution

O  usado como calculadora:

- Expressões aritméticas (aqui o resultado é mostrado e não guardado):

```
> 2 + 3/4 * 7^2
```

```
[1] 38.75
```

```
> exp(-2)/log(sqrt(2))
```

```
[1] 0.3904951
```

```
> sin(pi)^2 + cos(pi)^2
```

```
[1] 1
```

```
> sin((pi)^2) + cos((pi)^2)
```

```
[1] -1.332987 # Note a diferença!!!!
```

- Atribuição (resultado guardado num objecto):

```
> x <- 3 # O resultado é guardado na variável
```

```
> x # para mostrar o conteúdo de x
```

As funções mais usuais no

- > `sqrt()` - raiz quadrada
- > `abs()` - valor absoluto
- > `log()` - logaritmo de base e
- > `log10()` - logaritmo de base 10
- > `exp()` - exponencial
- > `sin()` ; > `cos()` ; > `tg()`
- > `factorial(n)` - factorial, $n!$
- > `choose(n,k)` - devolve $\binom{n}{k}$

- Os **objectos** do  são entidades que o  cria, manipula e podem ser guardados num *workspace*.
- Para ver a lista dos objectos no *workspace*: `> ls()`
- Para ver a informação sobre os objectos no *workspace*:
`> ls.str()`
- Para apagar objectos: `> rm(x, y)`
- Para apagar **todos** os objectos existentes no *workspace*
`> rm(list=ls())`
- Para guardar o *workspace* num ficheiro: `> save.image()` ou
menu: **File** → **Save Workspace ...**
- O ficheiro *workspace* por omissão é `.RData`

Em vez de escrever os comandos directamente na consola do  podem ser escritos e guardados em **ficheiros de texto** já sem erros e até comentados, para facilitar a sua utilização posterior. Estes ficheiros devem ter extensão **.R** e devem ser guardados na pasta de trabalho.

Para:

- Criar um ficheiro de *script*
menu: `File` → `New script ...`
- Utilizar um ficheiro de *script*
menu: `File` → `Open script ...`

Estrutura e manipulação de dados

O que são Dados?

Dados são representações simbólicas de observações, medições ou factos recolhidos em fenómenos, indivíduos ou processos, expressos de forma quantitativa ou qualitativa. Os dados são registos de valores de variáveis.

Uma **variável** é uma característica, atributo ou propriedade de um fenómeno ou indivíduo que pode assumir diferentes valores ou estados.

As variáveis podem ser de natureza qualitativa (categórica) ou quantitativa (numérica).

Na investigação, é fundamental compreender e definir corretamente os tipos de variáveis – voltaremos a este assunto.

Algumas boas Práticas na Organização de Dados:

- Genericamente devem ser organizados numa tabela bidimensional:
 - Cada **linha** representa um registo (por exemplo, um indivíduo ou uma observação).
 - Cada **coluna** representa uma variável.
- Apenas a primeira linha deve conter os nomes das variáveis.
- Deve usar-se nomes adequados e descritivos para as variáveis.
- Evitar células vazias
- Escrever uma informação por célula: não combine múltiplos dados numa só célula.
- Não deve realizar cálculos nos ficheiros de dados brutos: realize cálculos apenas em análises posteriores.

Cuidados Adicionais

- Evite espaços desnecessários no início ou no fim dos nomes (ex.: 'Peso', ' Peso', 'Peso ').
- Nomes de variáveis e ficheiros não utilize espaços; prefira o sublinhado (_) ou o ponto (.) para separar palavras.
- Evite caracteres especiais como \$, , %, #, &, *, !, /, etc., podem causar problemas em linguagens de programação (ex.: R).
- Não misture números e unidades na mesma célula (ex.: escreva "45" numa célula e indique a unidade "g" numa coluna separada).
- Não escreva notas nas células
- Para utilização no  aconselha-se guardar os dados em ficheiros de texto simples: Formatos como CSV (.csv) ou TXT (.txt).

- Os **objectos** no  são caracterizados por:
 - nome;
 - tipo - algumas estruturas de dados são por exemplo **vector**, **matrix**, **factor**, **data frame**, **list**, **function**;
 - atributos:
 - *mode*: numeric, character, complex, logical;
 - *length*: número de elementos no objecto;

`>str(x)` - mostra a estrutura interna do objecto `x`

- **Nome**

- Deve começar com uma letra (A-Z ou a-z);
- Pode conter dígitos e/ou pontos;
- **Case-sensitive** (maiúsculas são diferentes de minúsculas).
- **Nomes a evitar** (porque são usados internamente pelo ). Alguns:
c, q, t, C, D, F, I, T, pi, diff, df, pt, if, else, for, in, next, repeat, else, while, break, NULL, NA, NaN, Inf, FALSE, TRUE

Vector: estrutura de dados do mesmo tipo (numérico ou caracteres), armazenados enumerados – é o tipo de objecto mais comum.

- Criação de um **vector** - o uso de `c()`

```
> x <- c(1.2, 5.7, 6.3, 8, 14)
```

```
> cores <- c("Red", "Green", "Blue")
```

```
> u <- c(F, T, F)
```

```
> mais.cores <- c(cores, "Yellow", "Black")
```

- Um vector pode conter símbolos especiais: **NA** (valor desconhecido, *missing value*), **NaN** (*Not a Number*), **Inf**, **- Inf**.

```
z <- c(log(0), NA, Inf); z
```

```
[1] -Inf NA Inf
```

- Geração de **sequências** (permite criar certos vectores)

```
> y <- 1:5
```

```
> w <- seq(1, 1.4, by = 0.1)
```

```
> w1 <- rep(1,7)
```

```
> w2 <- rep(1:3,2)
```

- **Operações com vectores**

```
> v1 <- c(1,3,-1,2); v2 <- c(2,4,5,1)
```

Nota: operações realizadas elemento a elemento – se um dos vectores tiver menor dimensão que o outro é concatenado consigo próprio

```
> v1+v2; v1*v2; v1*2; 2/v1
```

- **Operadores lógicos**

- > `x>4; x>4 & x<6` (& conjunção)
- > `x<5 | x >= 8` (| disjunção)
- > `2==sqrt(4)` [1] TRUE

- **Seleção de elementos de um vector** – usa-se []

- > `cores[1]` - devolve a 1ª componente do vector `cores`
- > `cores[-c(1,3)]` - mostra o vector resultante da remoção dos elementos na posição 1 e 3 do vector `cores`
- > `x[u]` - devolve as componentes de `x` correspondentes às componentes TRUE de `u`
- > `x[x>2 & x<14]` - devolve as componentes de `x` entre 2 e 14

Algumas funções realizadas sobre um vector x

- > `length(x)` - devolve o numero de elementos do vector x
- > `sort(x)` - devolve um vector com os elementos do vector x ordenados por ordem crescente
- > `sum(x)` - devolve a soma dos elementos do vector vector x
- > `prod(x)` - devolve o produto dos elementos do vector x
- > `cumsum(x)` - devolve um vector cujos elementos são a soma acumulada dos elementos do vector x
- > `cumprod(x)` - procedimento análogo ao anterior, com o produto
- > `max(x)`; `min(x)` - devolve máximo e mínimo dos elementos do vector vector x
- > `factorial(x)` - devolve, para cada componente x_i , $\Gamma(x_i + 1)$
- > `sample(x)` - faz uma permutação dos elementos do vector x

Objetos no R - Matrix

Uma **matriz** é uma estrutura de dados, do mesmo tipo, referenciados por dois índices (a duas dimensões). Define-se pelo número de linhas **nrow** e número de colunas **ncol** e um conjunto de **nrow** × **ncol** valores.

```
>M <- matrix(1:12,nrow=3,ncol=4);M
>rownames(M)<-c("L1","L2","L3")
>colnames(M)<-c("C1","C2","C3","C4")
```

M

	[,1]	[,2]	[,3]	[,4]		C1	C2	C3	C4
[1,]	1	4	7	10	L1	1	4	7	10
[2,]	2	5	8	11	L2	2	5	8	11
[3,]	3	6	9	12	L3	3	6	9	12

Os valores são dispostos por coluna, a menos que seja indicado

```
> M <- matrix(1:12,3,4,byrow=T)
```

Seleção de elementos de uma matriz

```
> M[3,1]      # elemento que está na linha 3 e coluna 1
> M[3,]      # vector com os elementos da linha 3
> M[,1]      # vector com os elementos da coluna 1
> M[c(1,4),] # elementos da linha 1 e linha 4
              # de todas as colunas
> M[-2,c(1,4)] # submatriz constituída pelas
              # linhas 1 e 3 e colunas 1 e 4
```

Operações com matrizes

```
> A<-matrix(c(3,2,-4,0,-1,8, 2,1,3,4,2,0),nc=4);A
> A * M      # faz o produto elemento a elemento
              # (verifique o que dá!!!)
> A %*% M    # faz o produto usual de matrizes
> t(A)       # transposta de A
> diag(k)    # faz a matriz identidade de dimensão k
> rowMeans(A) # devolve o vector das médias por linha
> solve(A)   # inversa de A
> solve(A,b) # devolve o vector x na equação Ax=b

> vec <- rep(2,4)
> M * vec    # por replicação tem o vector (2,2,2,2)
              # e faz produto componente a componente
> M %*% vec  # faz o produto usual (matemático)
              # da matriz M pelo vector (2,2,2,2)
```

- **List**: colecção ordenada de objectos (componentes da lista) que podem ser de tipos diferentes (vectores numéricos, vectores lógicos, matrizes, funções,...)

```
> aluno <- list(num=12345, nome="Manuel",  
+ notas=c(12.5,13.4,12.1,14.3),curso="Eng.Florestal")  
> str(aluno) #dá designação e mode
```

- As componentes de uma lista podem ser referidas pelo seu índice, com `[[]]` ou pela sua designação

```
> aluno[[2]]           # segunda componente  
> aluno$nome          # componente designada "nome"  
> aluno[2:3]          # resulta uma sublista
```

Nota: O resultado de muitas funções é uma lista.

Um **data frame** é semelhante a uma matriz em que as colunas podem conter dados de diferentes tipos.

Um **data frame** pode ser visto como **uma tabela de dados**: colunas – são as variáveis; linhas – são os registos (as observações em cada variável)

É a estrutura usual para armazenar tabelas de dados

Leitura de um data frame já existente no 

```
> data() # mostra vários data frame existentes
        # no package "datasets"
> data(ToothGrowth) # carrega os dados para
                   # a memória do R
> ToothGrowth      # mostra os valores contidos
                   # no data frame
```

Como lidar com um Data Frame

```
> str(ToothGrowth)
      # mostra a estrutura do data frame
> head(ToothGrowth)
      # para visualizar as 6 primeiras linhas
> names(ToothGrowth)
      # dá os nomes das variáveis (colunas)
> dim(ToothGrowth)
      # dá a dimensão (n.linhas, n. colunas)
> ToothGrowth[,2]
      # dá a 2ª coluna
> ToothGrowth$len
      # mostra os valores da variável "len"
```

Consultar um `data frame` pode ser mais simples utilizando a função `attach()`.

Permite aceder directamente às *colunas de um `data frame`*, sem necessidade de referir o nome do `data frame`.

```
> len                                # objecto desconhecido
> attach(ToothGrowth)
> len # permite aceder às colunas do data frame
> detach(ToothGrowth) # operação inversa de attach
> len                                # objecto de novo desconhecido
```

Um Data Frame famoso

Um base de dados famosa é `iris`, existente no . Disponibiliza 4 medidas observadas em 3 espécies de `iris` (é um `data frame` com 150 linhas e 5 colunas)

```
>iris
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
....					

```
> names(iris)
```

```
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"
```

```
> dim(iris)
```

```
[1] 150  5
```

Um Data Frame famoso

```
> iris[,1]
```

```
[1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0  
[9] 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 ...
```

```
> mean(iris[,1])
```

```
[1] 5.843333
```

```
> summary(iris[,1]) # o mesmo que summary(iris$Sepal.Length)
```

```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.  
4.300  5.100   5.800   5.843  6.400   7.900
```

```
> table(iris$Species)
```

```
  setosa versicolor  virginica  
    50         50         50
```

Como criar um Data Frame

```
> pauta <- data.frame(N.Aluno = c(18355, 17456, 19334, 17756),  
+   turma = c("T1", "T2", "T3", "T3"),  
+   notas.Est = c(10.3,9.3, 14.2, 15))
```

```
> pauta; pauta$notas.Est
```

	N.Aluno	turma	notas.Est	
1	18355	T1	10.3	
2	17456	T2	9.3	
3	19334	T3	14.2	
4	17756	T3	15.0	
...				
[1]	10.3	9.3	14.2	15.0

Uma das formas mais comuns de armazenar dados para trabalhar no  é usar **ficheiros de texto**.

Tendo um ficheiro no formato **txt** ou **dat** ou **csv** (*Comma Separated Values*), i.e., os valores em cada linha estão separados por vírgulas ou ponto e vírgula, **deve**

- 1 abrir-se o ficheiro com um editor de texto (Notepad, Wordpad) para visualizar a estrutura
- 2 para ler usar o comando `read.table()`

```
>read.table("ficheiro",header=TRUE)
```

dependendo da estrutura dos dados.

Quando se tem dados em Excel **deve guardar-se cada folha num arquivo csv**. Dependendo das configurações do computador as colunas virão separadas por vírgula (,) ou ponto e vírgula (;)

Exemplo

```
>semente<-read.table("sementes.csv",header=TRUE,  
+ dec = ".",sep=";",as.is = TRUE)  
>head(semente)
```

	melhorada	tradicional
1	3.46	3.18
2	3.48	3.67
3	2.74	2.92
4	2.83	3.10

Existem outras funções de leitura, semelhantes a `read.table()`, cujas diferenças residem no **separador** que é usado por omissão

`read.csv()` – separador decimal é **ponto**;

`read.csv2()` – separador decimal é **vírgula**

Para escrever o conteúdo de um `data frame`, "x", num ficheiro "output.csv", compatível com o Excel, usa-se a função

```
>write.table(x,file="output.csv",sep=";",  
+ dec=".",row.names=FALSE)
```

Para escrever num ficheiro `.txt` compatível com o *Notepad* fazer

```
>write.table(x,file="output.txt",sep=","  
+ ,row.names=FALSE)
```

Funções e Programação em

O  tem um vasto conjunto de funções já definidas, orientadas para o objecto

Estrutura:

`>função(argumentos obrigatórios, argumentos opcionais)`

- Exemplo de funções standard (já referimos algumas atrás)

```
abs() log() log10() sqrt() round(x,3)
exp() sin() cos() tan() gamma() choose(n,k)
```

- Funções de álgebra matricial

```
t(X) nrow(X) eigen(X) solve(A,b) det(X)
```

- Funções estatísticas

```
mean()  median()  quantile(x,prob=p)
var()   sd()      plot()  barplot()
summary() sample() hist()  boxplot()
predict() lm()    aov()   t.test()
```

Mais adiante veremos mais ...

- Uma função é definida por um **nome**, uma **lista de argumentos** separados por vírgulas e um **bloco de instruções** (corpo da função)

Expressão geral

```
>function(arguments) {  
  comandos  
}
```

Exemplo Cálculo do coeficiente de variação de um vector de dados

```
>coef.var<- function(x) {  
  cv<-sd(x)/mean(x)*100  
  return(cv)  
}  
  
>z<-c(2,4,6,2,4,8,9,1,3,2,7,8,3,2)  
>coef.var(z)  
  
[1] 62.1123
```

A função `for ()`. Sintaxe

```
> for (indice in sequencia) {  
  expressão a executar  
}
```

Exemplo

```
> x<-c(2,4,6,2,4,8,9,1,3,2,7,8,3,2)  
> soma<-0  
> for (i in 1:length(x))  
  {soma<-soma+x[i]}  
> soma
```

A estrutura `if()`

```
>xval=c(-0.54, 1.35, -1.09, 0.23, -0.76, 0.85, -0.55,  
        -0.17, -0.31, 0.91)  
>conta<-0; soma<-0  
  
>for (i in 1:10)  
{ >if (xval[i]<0) {conta<-conta+1}  
  else  
    {soma<-soma+xval[i]}}  
}  
>conta;soma  
>print(conta);print(soma)
```

Estatística Descritiva e Análise Exploratória de Dados

Análise exploratória de dados—alguns conceitos básicos

População ou universo e **Unidade Estatística**

Variável: característica de interesse

Amostra: subconjunto da população ou conjunto de dados observados

A **análise exploratória de dados** tem como objectivo: organizar, sumariar, apresentar e extrair informação de um conjunto de dados.

As variáveis que interessa estudar podem ser de **natureza qualitativa (nominal ou ordinal)** ou de **natureza quantitativa (discreta ou contínua)**

Aos dados (matéria prima da Estatística Descritiva) costuma atribuir-se a mesma classificação, i.e. diz-se serem de **natureza qualitativa (nominal ou ordinal)** ou de **natureza quantitativa (discreta ou contínua)**.

Análise exploratória de dados—alguns conceitos básicos

Os procedimentos que a Estatística Descritiva pode usar dependem da natureza dos dados.

Para os dados de **natureza qualitativa** os procedimentos habituais para o seu estudo descritivo são: usar **tabelas de frequências**, **diagramas de barras** e podendo ainda calcular-se a **moda** (se existir).

Para os dados de **natureza quantitativa** pode ser feito o seu estudo descritivo com **tabelas de frequências** e **métodos gráficos**: **histogramas** ou **diagramas de barras** e o cálculo de vários **indicadores numéricos**: **média**, **mediana** e **quantis**, **moda**, **variância** e **desvio padrão**, etc....

- Consideremos os dados `InsectSprays` existente no *package* `datasets` do .

```
>help(InsectSprays)
```

```
>data(InsectSprays)
```

```
>head(InsectSprays)
```

```
>str(InsectSprays)
```

```
'data.frame':  72 obs. of  2 variables:
```

```
$ count: num  10 7 20 14 14 12 10 23 17 20 ...
```

```
$ spray: Factor w/ 6 levels "A","B","C","D",...:...
```

A variável `spray` é um **factor** com 6 níveis

- Análise descritiva básica: função `summary()`

```
>summary(InsectSprays)
```

count	spray
Min. : 0.00	A:12
1st Qu.: 3.00	B:12
Median : 7.00	C:12
Mean : 9.50	D:12
3rd Qu.:14.25	E:12
Max. :26.00	F:12

- Análise descritiva básica **por subgrupos**: função `by()`

```
> by(InsectSprays$count, InsectSprays$spray, summary)
```

Análise exploratória de dados

InsectSprays\$spray: A

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
7.00	11.50	14.00	14.50	17.75	23.00

InsectSprays\$spray: B

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
7.00	12.50	16.50	15.33	17.50	21.00

InsectSprays\$spray: C

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.000	1.000	1.500	2.083	3.000	7.000

InsectSprays\$spray: D

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
2.000	3.750	5.000	4.917	5.000	12.000

...

Utilização de indexação

```
>summary(InsectSprays[InsectSprays$count>10,])
```

	count	spray
Min.	:11.00	A: 9
1st Qu.	:13.00	B:11
Median	:16.00	C: 0
Mean	:16.68	D: 1
3rd Qu.	:20.00	E: 0
Max.	:26.00	F:10

Alternativamente pode usar-se o comando `subset()`

```
>summary(subset(InsectSprays, count>10))
```

Tabela de frequências - caso seja uma **variável qualitativa** ou **quantitativa discreta** com poucos valores distintos

Vamos considerar um subconjunto dos dados do exemplo

```
> parte.dados<-subset(InsectSprays, count>10)
```

```
> head(parte.dados)
```

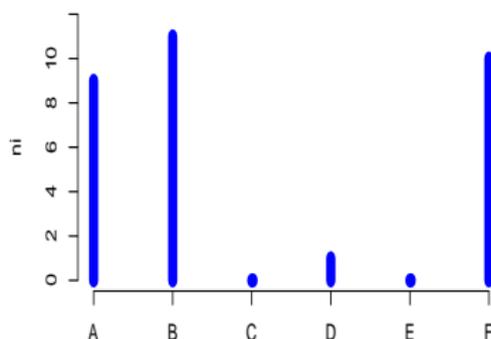
	count	spray
3	20	A
4	14	A
5	14	A
6	12	A
8	23	A
9	17	A

Dados da variável spray no subconjunto *parte.dados*

```
> ni<-table(parte.dados$spray) #freq. absoluta  
> fi<-ni/sum(ni)  
> fi.ar<-round(fi,3)
```

```
> cbind(ni,fi,fi.ar)
```

	ni	fi	fi.ar
A	9	0.29032258	0.290
B	11	0.35483871	0.355
C	0	0.00000000	0.000
D	1	0.03225806	0.032
E	0	0.00000000	0.000
F	10	0.32258065	0.323



Análise exploratória de dados–tabelas

Vamos considerar todas as contagens da *dataframe* `InsectSprays`

Tabela de frequências - variável quantitativa

```
> table(InsectSprays$count)
```

0	1	2	3	4	5	6	7	9	10	11	12	13	14	15	16	17	19	20	21	22
2	6	4	8	4	7	3	3	1	3	3	2	4	4	2	2	4	1	2	2	1

A variável `count` do `data frame` `InsectSprays` é **discreta mas** como vemos apresenta **muitos valores distintos**. Neste caso, **a tabela não tem interesse** e tal como se faz com uma variável contínua, a construção de uma tabela de frequências para sumariar os dados obtém-se com recurso ao comando `hist()`.

Os dados vão ser agrupados em classes (uso da Regra de Sturges) ou de classes por nós definidas.

```
> ?hist
```

```
> attach(InsectSprays)
> hist(count,plot=F) #devolve uma lista
```

```
$breaks
```

```
[1] 0 5 10 15 20 25 30
```

```
$counts
```

```
[1] 31 10 15 9 5 2
```

```
$density
```

```
[1] 0.086111111 0.027777778 0.041666667 0.025000000 0.013888889 0.005555556
```

```
$mids
```

```
[1] 2.5 7.5 12.5 17.5 22.5 27.5
```

```
$xname
```

```
[1] "count"
```

```
$equidist
```

```
[1] TRUE
```

```
attr(,"class")
```

```
[1] "histogram"
```

O resultado da função `hist(count, plot=F)` é então uma `List` com as seguintes componentes

`breaks` - limites das classes

`counts` - frequência absoluta de cada classe

`density` - (frequência relativa) / (amplitude de cada classe)

`mids` - ponto médio de cada classe

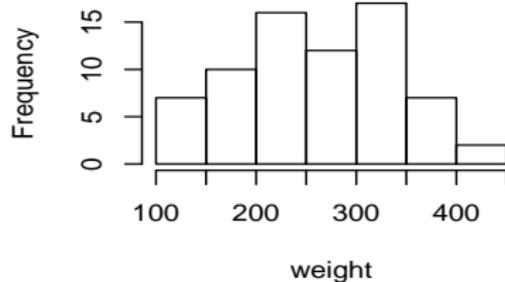
`equidist` - lógico que indica se as classes têm ou não amplitude constante

Nota: se as classes têm amplitude variável, a altura de cada rectângulo é a frequência relativa/amplitude da classe - feito por omissão, no 

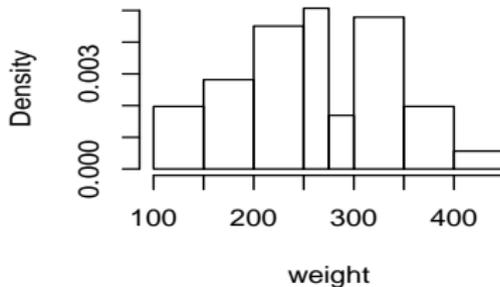
```
> data(chickwts)
> head(chickwts)
> par(mfrow=c(2,2)) # permite representar 4 gráficos
> hist(weight,breaks=
+   c(seq(100,250,50),275,seq(300,450,50)))
+   #comparar as alturas das classes 3 e 4
> hist(weight, freq=T,breaks=
+   c(seq(100,250,50),275,seq(300,450,50))) ##$mensagem
> hist(weight,col="grey",main="Hist. do peso",
+   freq=F,ylab="Freq. relat")
```

Análise exploratória de dados

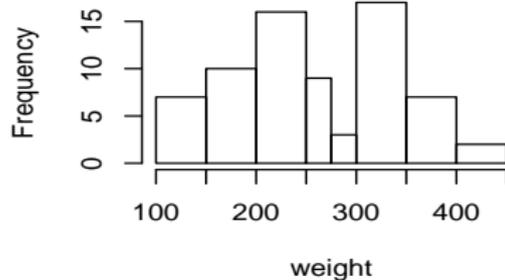
Histogram of weight



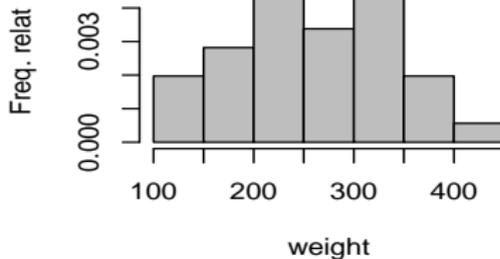
Histogram of weight



Histogram of weight



Hist. do peso



Análise exploratória de dados—Indicadores

Indicadores de forma: coeficientes de assimetria e de achatamento.

Vamos usar o *package* `fBasics`

```
>library(fBasics)
> x <- weight
> skewness(x) #coeficiente de assimetria
[1] -0.01136593
attr(,"method")
[1] "moment"
> kurtosis(x) #coeficiente de achatamento
[1] -0.9651994
attr(,"method")
[1] "excess"
```

```
>basicStats(x) #do package fBasics
```

```
> basicStats(x)
```

nobs	71.000000
NAs	0.000000
Minimum	108.000000
Maximum	423.000000
1. Quartile	204.500000
3. Quartile	323.500000
Mean	261.309859
Median	258.000000
Sum	18553.000000
SE Mean	9.265644
LCL Mean	242.830116
UCL Mean	279.789602
Variance	6095.502616
Stdev	78.073700
Skewness	-0.011366
Kurtosis	-0.965199

Gráficos no R com `plot()`

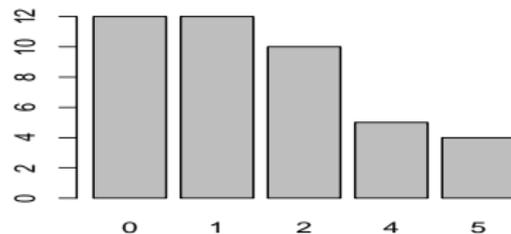
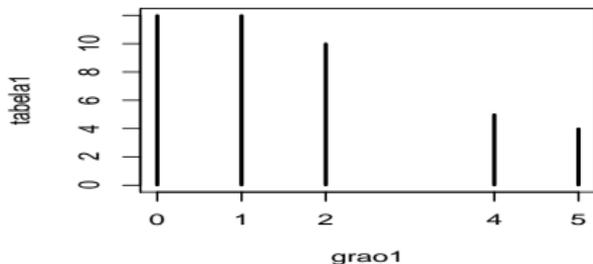
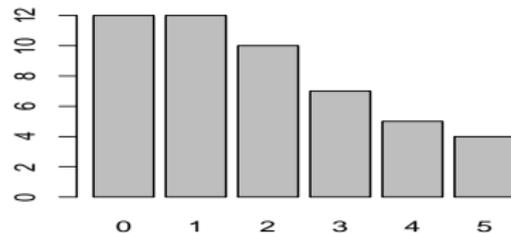
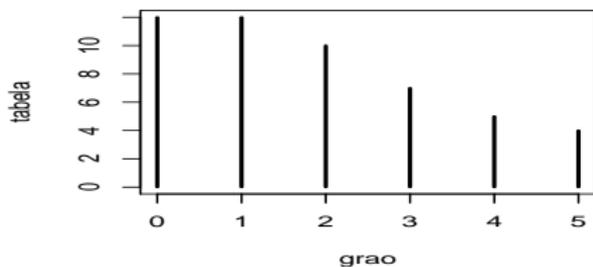
`plot(tab)` – produz um gráfico com barras se `tab` é uma tabela associada a um vector numérico

```
>grao<-c(1,2,0,0,1,4,2,5,1,1,5,0,2,2,3,2,1,0,0,3,3,3,
2,2,5,5,0,3,1,0,0,1,1,2,0,4,1,4,0,3,4,2,3,1,1,0,2,0,4,1)
>tabela<-table(grao)
>par(mfrow=c(2,2))
>plot(tabela)
```

Um outro gráfico também com barras, mas que pode ser usado quando se tem um vector não numérico é o `barplot()` . Vamos desenhá-lo para os mesmos dados

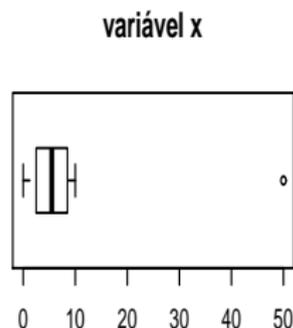
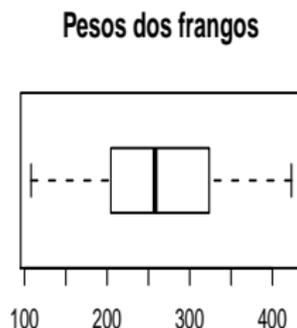
```
>barplot(tabela) #Note a diferença no gráfico
>grao1<-c(1,2,0,0,1,4,2,5,1,1,5,0,2,2,2,1,0,0,
 2,2,5,5,0,1,0,0,1,1,2,0,4,1,4,0,4,2,1,1,0,2,0,4,1)
>tabela1<-table(grao1)
>plot(tabela1)
>barplot(tabela1)
```

Gráficos com `plot()` e `barplot()` – veja a diferença



A “Caixa de bigodes” – `boxplot()`

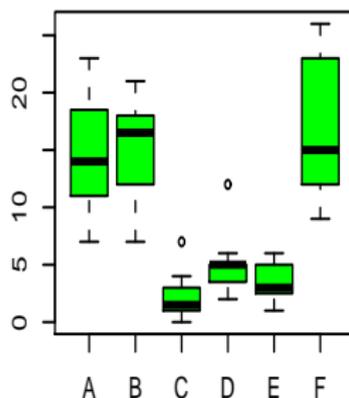
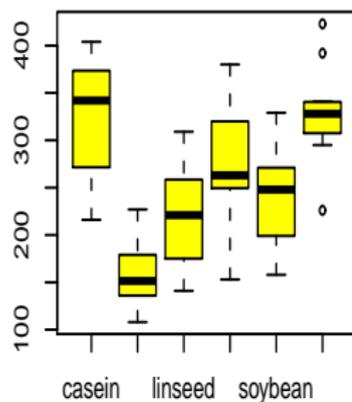
```
> par(mfrow=c(1,3))  
> boxplot(weight, main="Pesos dos frangos",col=3)  
> boxplot(weight, main="Pesos dos frangos",horizontal=T)  
> x <- c(0:10, 50)  
> boxplot(x,horizontal=TRUE,main="variável x") # observar outliers
```



“Caixas de bigodes” paralelas

```
>par(mfrow=c(1,3))
>boxplot(weight~feed, main="Pesos frangos/dieta",col = "yellow")
>boxplot(count~spray, col = "green",data=InsectSprays)
>boxplot(len ~ dose, data = ToothGrowth,
+       boxwex = 0.25, at = 1:3 - 0.2,
+       subset = supp == "VC", col = "yellow",
+       main = "ToothGrowth",
+       xlab = "Vitamina C dose mg",
+       ylab = "cresc dos dentes",
+       xlim = c(0.5, 3.5), ylim = c(0, 35), yaxs = "i")
>boxplot(len ~ dose, data = ToothGrowth, add = TRUE,
+       boxwex = 0.25, at = 1:3 + 0.2,
+       subset = supp == "OJ", col = "orange")
>legend("bottomright", c("Acido Asc.", "Sumo de laranja"),
+       fill = c("yellow", "orange"),cex=0.6)
```

Pesos frangos/dieta



ToothGrowth

