INSTITUTO SUPERIOR DE AGRONOMIA
Universidade de Lisboa

Departamento de Ciências e Engenharia de Biossistemas
Agrupamento de Matemática

Matemática II

Introdução à Aplicação R

2013/2014

(F. Valente e M. Mesquita)

O que é o R?

- É um conjunto integrado de ferramentas computacionais que permitem a manipulação e análise de dados, o cálculo numérico e a produção de gráficos de qualidade.
- É uma linguagem de programação bem desenvolvida e simples (uma variante da linguagem S).
- É uma linguagem orientada por objectos:
  - as estrutura de dados manipuladas são armazenadas na memória activa do computador na forma de objectos, que têm um nome e aos quais se podem aplicar acções.

Instalar o R

- O R é uma aplicação de distribuição gratuita e de código público (http://cran.r-project.org/), existindo versões já compiladas para execução nos principais sistemas operativos (Windows, Linux e Macintosh).
- Depois de fazer o *download* da versão adequada ao sistema operativo do computador (por ex. R-3.0.2-win.exe, para o Windows), para instalar o R basta executar esse ficheiro.

Iniciar uma sessão de R

- Criar, na respectiva área de trabalho, uma nova pasta (por exemplo, Mat2) onde irão ser guardados os ficheiros de dados – pasta de trabalho.
- Iniciar o R:

Start → All Programs → R → R 3.0.2

(*em alternativa, pode-se criar um atalho no desktop e alterar as suas propriedades de modo a iniciar a aplicação na pasta de trabalho*)

RGui

- O R funciona fundamentalmente pelo modelo "pergunta-resposta":
  - Os comandos introduzem-se a seguir à prompt (>) e são executados após pressionar Enter (↵)
- Edição de comandos:
  - Repetir comandos ou navegar entre comandos: ↑ ou ↓
  - Percorrer a linha de comandos: ← ou →
  - Colocar o cursor no início / fim da linha de comandos: Home / End
- Para terminar o R:
  - executar o comando `q()` ou fechar a janela da aplicação.

Comandos elementares

- Expressões aritméticas - são calculadas, o seu valor é impresso mas não é guardado.

```
> 2+3/4*7^2
[1] 38.75
> exp(-2)/log(sqrt(2))
[1] 0.3904951
```

- Atribuições - calcula o valor de uma expressão e guarda-o numa variável mas o resultado não é impresso.

```
> x <- 2     # <- é o operador de atribuição
> x
[1] 2
> x <- 2+3/4*7^2
> x
[1] 38.75
```

Designação das variáveis

- Os **nomes das variáveis** podem conter letras (maiúsculas são diferentes de minúsculas), algarismos e pontos. É recomendavel que os nomes das variáveis comecem por uma letra.
- Alguns nomes são utilizados pelo sistema, pelo que devem ser evitados (por ex., `c, q, t, C, D, F, I, T, diff, df, pt`).

Exemplo: `Pb.ISA`

Script

- Os comandos podem ser escritos e guardados num ficheiro de texto para facilitar a sua utilização posterior. Estes ficheiros devem ter extensão .R e ser guardados na pasta de trabalho.
- Criar um novo ficheiro de *script*

  File → New script
- Abrir um ficheiro de *script* existente

  File → Open script...

Ajuda

- Para obter informação sobre uma função específica, por exemplo a função `sin()`, o comando é

```
> help(sin)
> help("sin")
> ?sin
```

- Para pesquisar uma sequência de caracteres no help do R o comando é `help.search()`, por exemplo
- `> help.search("solve system")`
- Para navegar nas páginas de ajuda do R utilizando um browser de internet o comando é

```
> help.start()
```

(qualquer um dos comandos anteriores pode também ser executado através do menu Help)

Objectos

- As entidades que o R cria e manipula e que ficam guardadas na memória do computador.
  - Para mostrar os objectos disponíveis,

    `> ls()`
  - Para mostrar alguma informação sobre os objectos,

    `> ls.str()`
  - Para remover objectos,

    `> rm(objecto1, objecto2,...)`
  - Para remover todos os objectos,

    `> rm(list=ls())`

Objectos (*Workspace*)

- A colecção de todos os objectos disponíveis numa sessão pode ser guardada num ficheiro com vista à sua utilização em futuras sessões de R.
  - No final de cada sessão é perguntado ao utilizador se quer guardar o *workspace.* Caso a resposta seja afirmativa, todos os objectos disponíveis em memória são guardados no ficheiro ".RData", na pasta de trabalho actual, podendo ser carregados na próxima sessão de R.
    - Especificação da pasta de trabalho:

      menu File → Change dir...
    - Importar um ficheiro de objectos (por exemplo o ficheiro ".RData"):

      menu File → Load Workspace...

Exercício R1

a) Importar os ficheiros de objectos
DadosMeteo.RData
serras.RData
ExSlides_Desc.RData

b) Ver todos os objectos em memória.

Alguns objectos

- *Vector*
- *Matrix*
- *List*
- *Data Frame*
- *Function*

- Nota: deve distinguir-se os objectos do R *vector* e *matrix* das entidades matemáticas vector e matriz. Esta distinção justifica-se por ser possível no R realizar certas operações que não estão definidas na Álgebra Linear. Por exemplo, enquanto no R é possível somar vectors com diferente número de componentes, na Matemática esta operação só está definida para vectores da mesma dimensão.

Vector

- É uma colecção ordenada de elementos do mesmo tipo (valores numéricos, lógicos, alfanuméricos, ...).
- Uma das maneiras de criar um vector no R é através da função `c()`.

```
> x <- c(5.4, -3.7, 11.2, 0.78, 21.6)
> x
[1]  5.40 -3.70 11.20  0.78 21.60
> y <- c(FALSE,  TRUE,  TRUE,  TRUE, FALSE)
> y
[1] FALSE  TRUE  TRUE  TRUE FALSE
> nomes <- c("Ana","Paulo","Zé")
> nomes
[1] "Ana"   "Paulo" "Zé"
```


Vector

- Um vector pode conter os símbolos especiais `NA` e `NaN`, que designam valor desconhecido e valor não numérico, respectivamente.

```
> y <- c(NA, 53, 31, 15, 62)
> sqrt(c(-1,1,2))
[1]       NaN 1.000000 1.414214
Warning message:
NaNs produced in: sqrt(c(-1, 1, 2))
```


Vector

- `c()` é a função de concatenação. Os argumentos desta função podem ser eles próprios *vectors*, pelo que `c(v1,v2)` devolve a concatenação de `v1` e `v2`. Esta função pode ter um qualquer número de argumentos.

```
> c(x,0,x)
[1] 5.40 -3.70 11.20 0.78 21.60 0.00
  5.40 -3.70 11.20 0.78 21.60
> nomes <- c(nomes,"Manel")
> nomes
[1] "Ana"   "Paulo" "Zé"    "Manel"
```


Vector numérico (operações aritméticas)

- As operações são realizadas elemento a elemento.
- Alguns operadores aritméticos: `+`, `-`, `*`, `/`, `^`

```
> v1 <- c(5,7)
> v2 <- c(10,11,12,13)
> 1/v1
[1] 0.2000000 0.1428571
> v1+v2 # o vector de menor dimensão é repetido
        (total ou parcialmente) até ficar com o
        mesmo número de elementos do maior
> v1*4 # cada componente de v1 é multiplicada por 4
> v1*v2
[1] 50 77 60 91
```


Vector numérico (algumas funções básicas)

- `log`, `exp`, `sin`, `cos`, `tan`, `atan`, `sqrt`, `abs`, ...
- `length(x)` devolve o número de elementos do vector `x`,
- `sum(x)` devolve a soma dos elementos do 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)` devolve um vector cujos elementos são o produto acumulado dos elementos do vector `x`,
- `sort(x)` devolve um vector com os elementos do vector `x` ordenados por ordem crescente,
- `choose(n,k)` devolve o número de combinações ^nC_k

Exercício R2

a) Crie o vector `notas` com as classificações que obteve nas UC do primeiro semestre.
b) Determine a média aritmética das notas do primeiro semestre.
c) Crie o vector `creditos` com os créditos correspondentes às UC da alínea a).
d) Determine a média das notas do primeiro semestre, ponderadas pelos respectivos créditos.

Exercício R3

Indique uma instrução em R que permita calcular o factorial de 10.

Geração de sequências regulares

- O R permite gerar sequências de valores numéricos

```
> 1:15     # é o vector c(1,2,...,14,15)
> 15:1     # é o vector c(15,14,...,2,1)
> 2*1:15     # é o vector c(2,4,...,28,30)
> (2*1):15  # é o vector c(2,3,...,14,15)
> seq(1,15)
> seq(to=15,from=1)
> seq(-5,-1,by=0.5)
[1] -5.0 -4.5 -4.0 -3.5 -3.0 -2.5 -2.0 -1.5 -1.0
> seq(length=9, from=-5, by=0.5)
> rep(1:3,2) # é o vector c(1,2,3,1,2,3)
> rep(1:3,each=2) # é o vector c(1,1,2,2,3,3)
```


Exercício R4

Considere as folhas de exercícios de Estatística Descritiva (http://www.isa.utl.pt/dm/mat2AP/mat2AP/Exercicios_MatII_EstDesc.pdf).

Para os exercícios 1. e 2., crie vectores com as observações de cada um dos conjuntos de dados.

Indicadores numéricos no R

- `mean(x)` devolve a média dos elementos do vector `x`
- `median(x)` devolve a mediana dos elementos do vector `x`
- `quantile(x,probs=p)` devolve o quantil de ordem `p` dos elementos do vector `x`; por omissão `p=seq(0, 1, 0.25)`, isto é, devolve os extremos e os quartis de `x`
- `max(x)/min(x)` devolve o máximo/mínimo dos elementos do vector `x`
- `range(x)` devolve o vector `c(min(x),max(x))`
- `IQR(x)` devolve a amplitude inter-quartil dos elementos do vector `x`

Indicadores numéricos no R (cont.)

- `var(x)` devolve a variância dos elementos do vector `x`; é igual a `sum((x-mean(x))^2)/(length(x)-1)`
- `sd(x)` devolve o desvio padrão dos elementos do vector `x`; é igual a `sqrt(var(x))`
- `summary` devolve os extremos, os quartis e a média do vector `x`

Exercício R5

Os vectores `precip`, `temp` e `vento` contêm dados para o ano de 2006 de precipitação (mm), temperatura do ar (ºC) e velocidade do vento (ms^{-1}), respectivamente, medidos de 10 em 10 minutos numa estação meteorológica em Évora.

a) Determine para cada vector o número total de elementos.
b) Determine indicadores numéricos de localização e de dispersão para estes dados meteorológicos.

List

- Colecção ordenada de elementos que podem ser de tipos diferentes: vectores numéricos, vectores lógicos, matrizes, listas, funções, ...
- As componentes de uma lista são numeradas e podem ter um nome associado a elas.

```
> estudante<-list(nr=12345,nome="José Silva",
+ notas=c(12,14,10,14,11))
```

- O resultado de muitas funções é uma lista.

List (selecção de componentes)

- As componentes de uma lista são indexadas e podem ser referidas pelo seu índice, utilizando parêntesis rectos duplos; se tiverem nome, também podem ser referidas pelo nome.

```
> estudante[[2]]
> estudante$nome
```


Data frame

- As *data frames* são casos especiais de listas (com formato de tabela).

```
> serras <- data.frame(nome=serras.nome,
+ altitude=serras.altitude)
> iris
> ?iris
> str(iris)
```


Data frame (selecção de componentes)

- Os elementos de uma *data frame* podem ser referidos indicando a linha e a coluna:

```
> serras[4,1]
> serras[,2]
```

- A notação nomeDataFrame$nomeComponente (como numa *list*) também pode ser usada:

```
> serras$nome
> serras[[1]]
```


Data frame (função `attach()`)

- As consultas a *data frames* podem ser simplificadas utilizando a função `attach()`
- A função `attach()` permite aceder directamente às colunas de uma *data frame*, sem necessidade referir o nome da *data frame*.

```
> Species
Error: object "Species" not found
> attach(iris)
> Species

> detach(iris)
> Species
Error: object "Species" not found
```


Gráficos no R

- O R permite construir uma grande variedade de gráficos.
- Os comandos para criar gráficos dividem-se fundamentalmente em dois grupos:
  - **funções gráficas de alto nível** (que permitem criar um novo gráfico);
  - **funções gráficas de baixo nível** (que permitem acrescentar informação a um gráfico existente).
- A função `par` permite ver e modificar uma lista de parâmetros gráficos.

```
> par()  # devolve a lista dos valores dos parâmetros
> ?par   # informação sobre a função par
```

Função gráfica: `plot`

- `plot(f,from=a,to=b)`, com `f` uma função, traça o gráfico de `f` entre `a` e `b` (por omissão, `a=0` e `b=1`):

```
> plot(sin)
> plot(sin,-2*pi,2*pi)
```

- `plot(x)`, com `x` um vector numérico, produz um diagrama dos valores de `x` versus o seu índice:

```
> plot(serras.altitude)
> plot(sort(serras.altitude))
```

- `plot(x,y)`, com `x` e `y` vectores numéricos, produz um diagrama de pontos de `y` versus `x`:

```
> x<-seq(0.1,100,0.1)
> fx<-sin(x)/x
> plot(x,fx)
```


Função gráfica: `plot` (argumentos)

- Com a função `plot` podem ainda ser utilizados outros argumentos. O argumento `type` controla o tipo de gráfico produzido. O seu valor pode ser `"p"` (pontos) (valor por omissão), `"l"` (linhas), `"b"` (pontos ligados por linhas), `"h"` (linhas verticais dos pontos ao eixo das abcissas), `"s"` e `"S"` (função em escada).

```
> plot(serras.altitude,type="h")
> plot(temp,type="l")
> plot(x,fx,type="l")
```


Função gráfica: `plot` (argumentos)

- O título, sub-título e as legendas dos eixos num gráfico podem ser criados/alterados através dos argumentos `main`, `sub`, `xlab` e `ylab`, respectivamente. Os extremos dos eixos podem ser definidos com os argumentos `xlim` e `ylim`, cujos valores são vectores de duas componentes.

```
> plot(x,fx,type="l",main=
+ "Gráfico da função f(x)=sin(x)/x",
+ xlab="x",ylab="f(x)")
```

- Qualquer um dos argumento da função `par` pode ser também utilizado como argumento da função `plot` para modificar as características do gráfico produzido.

Visualização de vários gráficos

- As funções gráficas de alto nível criam, por omissão, um novo gráfico, apagando, se necessário, aquele que se encontra na janela gráfica.
  - Em alguns casos, a utilização do argumento `add=TRUE` permite sobrepor diferentes gráficos na mesma janela gráfica.

```
> plot(sin,-pi,pi)
> plot(cos,-pi,pi,add=TRUE,col="red",lty=2)
```

- Pode-se utilizar a função `par` para visualizar simultaneamente vários gráficos.

```
> op<-par(mfrow=c(2,1))
> plot(sin,-pi,pi);plot(cos,-pi,pi)
> par(op) # recupera os valores iniciais dos parâmetros
```


Funções gráficas de baixo nível

- Permitem acrescentar informação a um gráfico existente na janela gráfica:
  - as funções `points(x,y)` e `lines(x,y)` permitem acrescentar, respectivamente, pontos e pontos ligados por linhas;
  - a função `abline(a,b)` acrescenta uma recta de declive `b` e ordenada na origem `a`;
  - as funções `abline(v=x)` e `abline(h=y)` permitem adicionar rectas verticais (de abcissa `x`) e horizontais (de ordenada `y`), respectivamente;
  - a função `legend` permite acrescentar uma legenda ao gráfico.

Funções gráficas de baixo nível (exemplo)

```
> plot(sin,-pi,pi)
> plot(cos,-pi,pi,add=T,col="red",lty=2)
> legend(-3,1,c("sin","cos"),
+ col=c("black","red"),lty=1:2)
> abline(h=0,v=0)
```

Relembrando os exemplos 1 e 2

```
> sucesso
> ozono
```


Representação gráfica no R

- No caso de dados de natureza qualitativa ou quantitativa discreta, a função `table(x)` devolve uma tabela com a frequência absoluta para cada um dos elementos distintos do vector `x`.

```
> table(sucesso)
```

- A função `plot(x)`, em que `x` é uma tabela de frequências obtida com a função table, cria um diagrama de barras com os dados de `x`.

```
> plot(table(sucesso))
```

- A função `hist(x)`, cria um histograma com os elementos do vector `x`.

```
> hist(ozono)
```


A função `hist()`

- O parâmetro `breaks=x` da função `hist()` permite definir o número de rectângulos do histograma (por omissão, calculado pela regra de Sturges) em que :
  - `x` é o número de classes em que se pretende agrupar os dados, ou
  - `x` é um vector numérico com os limites das várias classes.

```
> hist(ozono,breaks=7)
> hist(ozono,breaks=seq(0,12,1.5))
> hist(ozono,breaks=seq(1,13,1.5))
> hist(ozono,breaks=seq(1,13,1.5),plot=F)
```


Exercício R6

a) Construa um histograma para os dados do vector `serras.altitude`.

b) Ainda com os dados do vector `serras.altitude`, construa três novos histogramas: um com classes de amplitude de 100m, outro com classes de amplitude de 200m e outro também com classes de amplitude 200m mas cujo o primeiro intervalo de classe é]100,300].

c) Dos quatro histogramas anteriores, qual permite ter uma melhor representação gráfica da distribuição das altitudes das serras de Portugal Continental? Justifique.

d) Apresente os quatro histogramas construidos nas alíneas anteriores na mesma janela gráfica (sugestão: utilize a função `par`).

Caixa-de-bigodes no R

- Estes gráficos podem ser obtidos em R utilizando a função `boxplot()`.

```
> boxplot(serras.altitude)
```


Exercício R7

1. Construa um diagrama de barras com os dados do vector `sucesso` colocando um título e legendas nos eixos.
2. Para os conjuntos de dados dos exemplos 1 e 2 (vectores `sucesso` e `ozono`), construa as respectivas caixas-de-bigodes. Analise os resultados obtidos, comparando-os com as representações gráficas efectuadas anteriormente.

Leitura de ficheiros

Uma tabela de valores, registada num ficheiro de texto (ASCII), pode ser atribuída a uma *data frame*, através da função

```
read.table(ficheiro,header=FALSE,sep="",
  dec = ".",row.names, col.names, as.is = FALSE,
  na.strings = "NA")
```

em que,

ficheiro é o nome do ficheiro de dados; se este não se encontrar na pasta de trabalho, é necessário indicar o endereço completo (utilizando / ou \\ como separador de pastas);

header é uma variável lógica que indica se o ficheiro tem ou não os nomes das variáveis na primeira linha. Por omissão o valor é FALSE;

sep é o caracter que separa os valores em cada linha; por omissão é `""` que corresponde a um ou mais espaços em branco;

Leitura de ficheiros (cont.)

dec é o caracter utilizado como separador decimal. Por omissão é o ponto;

row.names é um vector com os nomes das linhas. Por omissão `1,2,3,...`;

col.names é um vector com os nomes das colunas. Por omissão `V1,V2,V3,...`;

as.is é uma variável lógica que controla a conversão das variáveis alfanuméricas em factores. Se o seu valor é TRUE a leitura mantém o tipo original dos dados;

na.strings é o valor usado para os dados desconhecidos no ficheiro e que será convertido em `NA`.

Dados bivariados

- **Indicadores numéricos**
  - `cov(x,y)` devolve a covariância entre `x` e `y`
  - `cor(x,y)` devolve o coeficiente de correlação de `x` e `y`
- **Regressão linear simples** – método dos mínimos quadrados:
  - `lm(y~x)` devolve os coeficientes da recta de regressão de `y` sobre `x`
- Exemplo:

```
> x<-c(20,50,30,100,70); y<-c(3,5,3,10,8)
> cov(x,y);cor(x,y)
[1] 98.5
[1] 0.9854435
> lm(y~x)
Coefficients:
(Intercept)            x
    0.63592      0.09563
```


Dados bivariados: representação gráfica

- `plot(x,y)` ou `plot(y~x)`, com `x` e `y` vectores numéricos, produz um diagrama de dispersão (nuvem de pontos) de `y` versus `x`
- `abline(lm(y~x))` acrescenta ao diagrama de dispersão anterior, a recta de regressão dos mínimos quadrados de `y` sobre `x` .
- Exemplo:

```
> x<-c(20,50,30,100,70); y<-c(3,5,3,10,8)
> plot(y~x)
> abline(lm(y~x))
```