Operações e funções usuais.

As operações binárias usuais de adição, subtração, multiplicação, divisão, potenciação, resto de divisão inteira e quociente de divisão inteira, podem ser efetuadas através dos operadores `+, -, *, /, ^, %%, %/%`. Outras funções numéricas, tais como, raíz quadrada, valor absoluto, exponencial, logaritmo neperiano ou logaritmo de base b, arredondamento a um número especificado de dígitos, fatorial, podem ser efetuadas através das funções `sqrt, abs, exp, log, round, factorial` indicando entre parentesis curvos o argumento.

Algumas funções básicas do R.

Comando	Descrição
`help()`	retorna ajuda. Também pode-se usar `?`
`mode()` ou `typeof()`	tipo do objeto
`class()`	classe do objeto
`c()`	forma um vetor a partir da sequência indicada Ex: `c(1,-1,2,0.1)`, `c("A","b")`
`x<-objeto`	atribui o nome `x` ao objeto (ex: `x<-c(1,-1,2,0.1)`)
`ls()`	lista os objetos definidos (no ambiente de trabalho)
`rm(x,y)`	apaga os objetos `x` e `y`

Para introduzir um vetor númerico `x` com a função `scan` executa-se o comando `x<-scan()`, introduz-se os dados pelo teclado separando as componentes por um espaço ou fazendo um "copy-paste"a partir de outro documento e prima-se no "Enter" (1 ou 2 vezes) para terminar a introdução. No caso de vetores de caracteres usa-se `scan(what="")`.

Operadores lógicos.

`x==y`	`x` é igual a `y`
`x<y`	`x` é menor que `y`
`x<=y`	`x` é menor ou igual a `y`
`x>y`	`x` é maior que `y`
`x>=y`	`x` é maior ou igual a `y`
`x!=y`	`x` não é igual a `y`
`!b`	negação de `b`
`a|b`	`a` ou `b`
`a&b`	`a` e `b`

Algumas funções específicas para vetores.

`names(v)`	atribui nomes às componentes de `v`.
`length(v)`	número de componentes de `v`.
`max(v)  min(v)`	maior/menor valor em `v`.
`sort(v)`	ordena as componentes de `v` por ordem crescente.
`sum(v)  prod(v)`	efetua a soma/multiplicação das componentes de `v`.
`cumsum(v)  cumprod(v)`	somas/multiplicações acumuladas das componentes de `v`.
`numeric(n)`	vetor com `n` zeros.
`seq(n)`	vetor de `1` a `n` : alternativa a `1:n`.
`seq(m,n)`	vetor de `m` a `n` : alternativa a `m:n`.
`seq(m,n,by=p)`	vetor de `m` a `n` com passo `p`.
`rep(v,n)`	vetor com `n` cópias de `v` (ver também `rep(v,length.out=..)`, `rep(v,each=..)`, `rep(v,times=..)`)
`v[...]`	subvetor de `v` correspondente ao vetor de índices ou à condição dada entre [ ].

Outras funções usadas:

`table, paste, order, sample, which, mean, sd.`

Estruturas de controlo (documentação em `help("if")`)
`if(cond) {instruções} else {instruções}`
`for(var in seq) {instruções}`
`while(cond) {instruções}`
`repeat{instruções}` – deve existir uma instrução (*e.g.* `if(cond) break`)) para parar o ciclo.

Importante: Num bloco `{instruções}`, as instruções devem ser separadas por um ponto-vírgula (`;`) ou por mudança de linha. A condição `cond` deve ser um vetor lógico de comprimento 1. Em algumas situações (em particular na definição de funções por ramos), pode se usar o comando `ifelse` que permite uma condição dada por um vetor lógico de comprimento superior a 1 (ver `?ifelse`).

A sintaxa para a definição de uma **função** (documentação em `help("function")`) é

```
f<-function(arg1, arg2, ...){regra ou regras separadas por ; ou mudanca de linha}
```

Uma mensagem de erro pode ser incluída usando a função `stop()` em combinação com a instrução `if(cond) {instruções} else {instruções}`.

Comando	Descrição
`plot(x,y,opcoes)`	*(high level)* representa graficamente os pontos de coordenadas (x_i, y_i) onde x_i e y_i são as componentes de `x` e `y` Entre as opções: `main`, `xlab`, `ylab`, `type`... ver `help(plot)`.
`points(x,y)`	*(low level)* adiciona os pontos de coordenadas (x_i, y_i) a um gráfico já existente
`lines(x,y)`	*(low level)* num gráfico já existente junta os pontos (x_i, y_i) por segmentos de retas sucessivos
`abline(a,b)`	*(low level)* adiciona a um gráfico já existente a reta de equação $y = a + bx$
`plot(table(x),opcoes)`	*(high level)* diagrama de linhas para a amostra `x`
`barplot(table(x))`	*(high level)* gráfico de barras para a amostra `x`
`curve(f,a,b)`	*(high level)* representa o gráfico de f no intervalo $[a, b]$
`hist(x,opcoes)`	*(high level)* histograma para amostra `x`

Alguns parâmetros/Opcões	Descrição (documentação em `help(par)`, `help(plot)`...)
`col=`	código da cor ou nome da cor a usar
`lty=`	define o tipo de linha
`lwd=`	define a espessura da linha
`locator(n, ...)`	devolve as coordenadas de n pontos do gráfico selecionados com o rato
`main="nome"`	coloca o título `nome` no gráfico
`xlab="nome"` / `ylab="nome"`	coloca o texto `nome` no eixo dos `x`/`y`
`type=" "`	determina o tipo do gráfico (`"l"` - linhas, `"h"` - linhas verticais...)
`xlim=c(a,b)` / `ylim=c(a,b)`	fixa `[a,b]` como o intervalo do eixo dos `x`/`y`

Matrizes.

Comando	Descrição
`matrix(x,ncol= ,nrow= )`	produz a partir do vetor `x` uma matriz da dimensão indicada
`outer(x,y,FUN)`	produz a partir dos vetores `x` e `y` uma matriz em que a entrada (i, j) é dada por `FUN`(x_i, y_j)
`dim(A)`	dimensão (nº linhas/nº colunas) da matriz `A`
`nrow(A)`	número de linhas da matriz `A`
`ncol(A)`	número de colunas da matriz `A`
`A[ , ]`	seleciona as componentes de `A` correspondentes aos índices ou às condições indicado(a)s
`t(A)`	transposta de `A`
`%*%`	produto matricial
`row(A)`	matriz em que cada entrada é substituída pelo seu índice de linha
`col(A)`	matriz em que cada entrada é substituída pelo seu índice de coluna
`rowSums(A)`	devolve um vetor contendo as somas das linhas de `A`
`colSums(A)`	devolve um vetor contendo as somas das colunas de `A`
`apply(A, MARGIN, FUN)`	aplique a função `FUN` às linhas (se `MARGIN=1`) ou às colunas (se `MARGIN=2`) de `A`

Duas outras funções da família da função `apply` são `sapply` e `tapply`:

- `sapply(v,FUN)` aplica a função `FUN` às componentes do vetor `v`.
- Se o factor `fact` e o vetor `v` têm mesmo comprimento, `tapply(v,fact,FUN)` aplica a função `FUN` aos grupos de `v` determinados pelos níveis de `fact`. Se o argumento `fact` é um vetor númerico, este será automaticamente transformado num factor.

Listas/Data frames.

Comando	Descrição
`list(nome1=...,nome2=...)`	produz uma lista
`lst$nome1` ou `lst[[1]]`	1ª componente da lista `lst`
`data.frame(nome1=..,nome2=...)`	produz um data frame
`df$nome1`	1ª colona do data frame `df`
`df[ , ]`	seleciona as componentes do data frame `df` correspondentes aos índices ou às condições indicado(a)s
`attach(df)`	coloca o data frame `df` no caminho de pesquisa (`search()`) permitindo chamar uma coluna de `df` pelo seu nome
`detach(df)`	operação inversa de `attach(df)`
`str(df)`	dá a estrutura do data frame `df`
`read.table("ficheiro.txt",header=T)`	importa os dados do ficheiro num data frame
`read.csv("ficheiro.csv")`	importa os dados do ficheiro num data frame

Outras funções usadas no âmbito do estudo das matrizes e data frames:

`diag, which(...,arr.ind=T), is.na.`