PROGRAMAÇÃO

2. Manipulação de Dados

Rita P. Ribeiro

Mestrado em Informática Médica
Curso de Especialização em Informática da Saúde

2016/2017

U.PORTO
FMUP FACULDADE DE MEDICINA UNIVERSIDADE DO PORTO
FC FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Programa

1. Conceitos Básicos do R
2. Manipulação de Dados
3. Sumarização de Dados
4. Visualização de Dados
5. Geração de Relatórios

Tabelas de Dados em Data Frames

- Tipicamente,as tarefas de análise de dados assentam sobre conjuntos de dados sob a forma de tabela onde:
  - as linhas representam observações;
  - as colunas representam atributos que caracterizam as observações.
- Em R usam-se *data frames* para guardar estas tabelas de dados.
- Como podemos importar dados de diferentes origens e/ou formatos para *data frames*?
- Como podemos facilmente manipular os dados contidos nestes *data frames*?

Importação de Dados

Dados Internos ao R

- O R inclui vários conjuntos de dados (*data sets*). Cada *package* pode também incluir novos conjuntos de dados.
- A lista de *data sets* pode ser obtida executando:

```
# datasets carregados
data()
# datasets da package MASS
data(package = "MASS")
# datasets das packages instaladas
data(package = .packages(all.available = TRUE))
```

- Para carregar um data set:

```
# carregar apenas o dataset
data(Pima.tr,package="MASS")
## carregar todo o package
library(MASS)
data(Pima.tr)
```

Importação de Dados

Ficheiros RData

- Qualquer objeto R e, portanto, qualquer *data frame* pode ser guardado num ficheiro `RData`.

```
dados <- data.frame(X=rnorm(3),Y=rnorm(3))
save(dados,file="exp.RData")
```

- O objeto guardado no ficheiro `RData` pode ser novamente carregado para o R.

```
rm(dados) # remove o objeto df da memória
dados

## Error in eval(expr, envir, enclos):  object 'dados' not found

load("exp.RData") # carrega o objeto contido no ficheiro
dados

##            X          Y
## 1 -0.6023181 -1.8427645
## 2  0.6380087 -1.6670012
## 3 -0.5526226  0.2369833
```

Importação de Dados

Ficheiros de Texto

- Muitas vezes, os *data sets* encontram-se guardados em ficheiros de texto onde:
  - cada observação do *data set* é representada por uma linha de texto;
  - o valor de cada atributo da observação é delimitada por um caracter separador.

Importação de Dados

Ficheiros de Texto (2)

- Ficheiros `CSV` (*comma-separated values*): os valores da linha são separados por vírgulas.

`pesoalt.csv`

```
Nome, Peso, Altura
Marco, 62.6, 1.65
Filipe, 55.4, 1.60
Valter, 70.2, 1.69
Nuno,, 1.72
João, 68.1, 1.73
```

```
dados <- read.csv("pesoalt.csv",
                  header=TRUE)
dados

##     Nome Peso Altura
## 1  Marco 62.6   1.65
## 2 Filipe 55.4   1.60
## 3 Valter 70.2   1.69
## 4   Nuno   NA   1.72
## 5   João 68.1   1.73
```

Importação de Dados

Ficheiros de Texto (3)

- Em alguns países, onde a vírgula é usada como separador decimal, é comum usar o ponto-e-vírgula como separador.

pesoaltPT.csv

```
Nome; Peso; Altura
Marco; 62,6; 1,65
Filipe; 55,4; 1,60
Valter; 70,2; 1,69
Nuno;; 1,72
João; 68,1; 1,73
```

```
dados <- read.csv2("pesoaltPT.csv",
                   header=TRUE)
dados

##     Nome Peso Altura
## 1  Marco 62.6   1.65
## 2 Filipe 55.4   1.60
## 3 Valter 70.2   1.69
## 4   Nuno   NA   1.72
## 5   João 68.1   1.73
```

Importação de Dados

Ficheiros de Texto (4)

- A função `read.table()` permite a importação de dados a partir de um qualquer formato de texto.

pesoalt.txt

```
Nome   Peso   Altura
Marco  62.6   1.65
Filipe   55.4  1.60
Valter  70.2   1.69
Nuno  ???  1.72
João  68.1  1.73
```

```
dados <- read.table("pesoalt.txt",
                    header=TRUE,na.strings="???")
dados

##     Nome Peso Altura
## 1  Marco 62.6   1.65
## 2 Filipe 55.4   1.60
## 3 Valter 70.2   1.69
## 4   Nuno   NA   1.72
## 5   João 68.1   1.73
```

Importação de Dados

Ficheiros de Texto (5)

- Família de funções para importação de dados a partir de ficheiros: `read.table()`, `read.csv()`, `read.csv2()`, `read.delim()`. etc.
- Alguns parâmetros:
  - `sep`: indica o caracter separador das colunas;
  - `dec`: indica o separador decimal;
  - `header`: indica se a primeira linha contém o nome das colunas;
  - `na.strings`: indica o conjunto de valores de texto que deve ser interpretado como valores desconhecidos, i.e. `NA`.

Importação de Dados

Folhas de Cálculo

- Existem vários métodos para importar dados a partir de uma folha de cálculo:
  - através de um ficheiro `csv`;
  - através do *clipboard* ou área de transferência;
  - através de funções de *packages* específicos.

Importação de Dados

Folhas de Cálculo (2)

- Podemos importar dados de uma folha de cálculo guardando o seu conteúdo num ficheiro de texto (p. ex. `CSV`)

`pesoalt.xls`

	A	B	C	D
1	Nome	Peso	Altura	
2	Marco	62.6	1.65	
3	Filipe	55.4	1.60	
4	Valter	70.2	1.69	
5	Nuno		1.72	
6	João	68.1	1.73	
7				
8				
9				

$\rightarrow$

`pesoalt.csv`

```
Nome, Peso, Altura
Marco, 62.6, 1.65
Filipe, 55.4, 1.60
Valter, 70.2, 1.69
Nuno,, 1.72
João, 68.1, 1.73
```

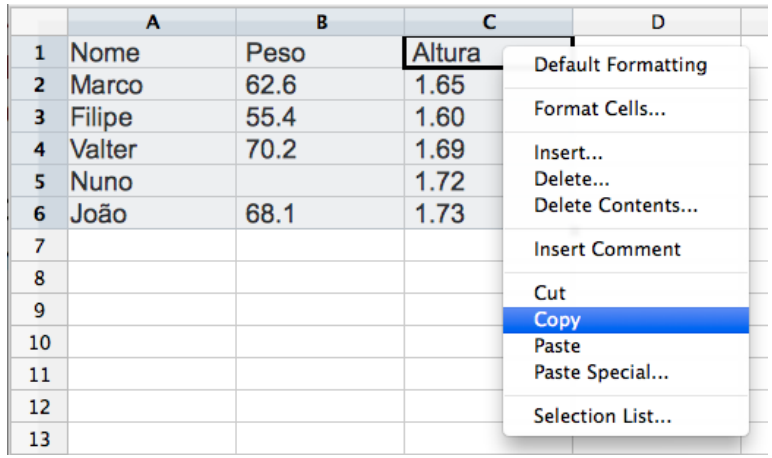
- e depois carregando o conteúdo do ficheiro para um *data frame* no R, através de funções como `read.table`, `read.csv`, etc

```
dados <- read.csv("pesoalt.csv",header=TRUE)
```

Importação de Dados

Folhas de Cálculo (3)

- Podemos importar dados de uma folha de cálculo, selecionando e copiando a área que pretendemos na folha de cálculo



- e depois usando a função `read.table()` sobre o "ficheiro temporário" `clipboard`.

```
dados <- read.table("clipboard",header=TRUE)
```

- Não é muito aconselhável porque o acesso ao *clipboard* pode variar com o sistema operativo.

Importação de Dados

Folhas de Cálculo (4)

- Podemos importar dados de uma folha de cálculo, usando um *package* com uma função específica para tal.
- Exemplos:
  - *package* `gdata`, função `read.xls()`

```
library(gdata)
dados <- read.xls("/Users/rpribeiro/Desktop/pesoalt.xls",
                  sheet=1)
```

  - *package* `readxl`, função `read_excel()`

```
library(readxl)
dados <- read_excel("/Users/rpribeiro/Desktop/pesoalt.xls",
                    sheet=1)
```

Importação de Dados

Outras Fontes de Dados

- Bases de Dados
  - O R faz *interface* com os SGBD principais através de *packages* como `DBI`, `RMySQL`, `ROracle`, etc.
- Software Estatístico
  - O R importa dados a partir do SPSS, Stata, S-Plus, SAS, etc. através dos *packages* `foreign`, `Hmisc`.
- Vários outros formatos e software
  - Mais detalhes no manual R Data Import/Export'.

Manipulação de Dados

Informações Básicas

- Existem funções que permitem inspecionar o conteúdo de data frames, sem que para tal seja necessário mostrar todo data frame. Estas funções são especialmente úteis perante grandes conjuntos de dados.

```
dim(Pima.tr) # número de linhas e colunas

## [1] 200   8

str(Pima.tr) # obter descrição da estrutura

## 'data.frame': 200 obs. of  8 variables:
##  $ npreg: int  5 7 5 0 0 5 3 1 3 2 ...
##  $ glu  : int  86 195 77 165 107 97 83 193 142 128 ...
##  $ bp   : int  68 70 82 76 60 76 58 50 80 78 ...
##  $ skin : int  28 33 41 43 25 27 31 16 15 37 ...
##  $ bmi  : num  30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3 ...
##  $ ped  : num  0.364 0.163 0.156 0.259 0.133 ...
##  $ age  : int  24 55 35 26 23 52 25 24 63 31 ...
##  $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 2 ...
```

Manipulação de Dados

Informações Básicas (2)

```
head(Pima.tr) # primeiras linhas

##   npreg glu bp skin  bmi   ped age type
## 1     5  86 68   28 30.2 0.364  24   No
## 2     7 195 70   33 25.1 0.163  55  Yes
## 3     5  77 82   41 35.8 0.156  35   No
## 4     0 165 76   43 47.9 0.259  26   No
## 5     0 107 60   25 26.4 0.133  23   No
## 6     5  97 76   27 35.6 0.378  52  Yes

tail(Pima.tr) # últimas linhas

##     npreg glu bp skin  bmi   ped age type
## 195     1 140 74   26 24.1 0.828  23   No
## 196     2 141 58   34 25.4 0.699  24   No
## 197     7 129 68   49 38.5 0.439  43  Yes
## 198     0 106 70   37 39.4 0.605  22   No
## 199     1 118 58   36 33.3 0.261  23   No
## 200     8 155 62   26 34.0 0.543  46  Yes
```

Manipulação de Dados com `dplyr`

- O package `dplyr` oferece funcionalidades que permitem manipular dados em R, de forma fácil e também eficiente.
- Permite lidar com várias fontes de dados (data frames, bases de dados, etc.)
- Tem implementadas as operações mais básicas de manipulação de dados.

Manipulação de Dados com `dplyr`

Formato dos Dados

- Os dados são guardados em `data frame tables`, uma espécie de data frames.
- A principal vantagem é na impressão

```
library(dplyr)
p <- tbl_df(Pima.tr)
p
```

```
## # A tibble: 200 × 8
##    npreg   glu    bp  skin   bmi   ped   age   type
## *  <int> <int> <int> <int> <dbl> <dbl> <int> <fctr>
## 1      5    86    68    28  30.2 0.364    24     No
## 2      7   195    70    33  25.1 0.163    55    Yes
## 3      5    77    82    41  35.8 0.156    35     No
## 4      0   165    76    43  47.9 0.259    26     No
## 5      0   107    60    25  26.4 0.133    23     No
## 6      5    97    76    27  35.6 0.378    52    Yes
## 7      3    83    58    31  34.3 0.336    25     No
## 8      1   193    50    16  25.9 0.655    24     No
## 9      3   142    80    15  32.4 0.200    63     No
## 10     2   128    78    37  43.3 1.224    31    Yes
## # ... with 190 more rows
```

Manipulação de Dados com `dplyr`

Operações Básicas

- `filter` - mostrar um subconjunto de linhas
- `select` - mostrar um subconjunto de colunas
- `arrange` - reordenar linhas
- `mutate` - adicionar novas colunas
- `summarise` - sumariar valores de uma coluna

Manipulação de Dados com `dplyr`

Operações Básicas: Algumas Regras

```
dados2 <- <operacao>(dados1,...)
```

- O primeiro argumento (`dados1`) é um `data frame table`.
- Os restantes argumentos (`...`) descrevem o que fazer com os dados.
- O objeto retornado (`dados2`) é do mesmo tipo do primeiro argumento (com exceção da operação `summarise`).
- O objeto no primeiro argumento nunca é alterado.

Manipulação de Dados com dplyr

Operações Básicas: filter

```
dados2 <- filter(dados1, cond1, cond2, ...)
```

- Retorna as linhas que satisfazem TODAS as condições indicadas.

```
filter(p,npreg > 4,age < 30)
```

```
## # A tibble: 10 × 8
##    npreg   glu    bp  skin   bmi   ped   age   type
##    <int> <int> <int> <int> <dbl> <dbl> <int> <fctr>
## 1      5    86    68    28  30.2 0.364    24     No
## 2      5   123    74    40  34.1 0.269    28     No
## 3      6   109    60    27  25.0 0.206    27     No
## 4      5   139    80    35  31.6 0.361    25    Yes
## 5      6   134    70    23  35.4 0.542    29    Yes
## 6      5   158    84    41  39.4 0.395    29    Yes
## 7      6   111    64    39  34.2 0.260    24     No
## 8      6   154    78    41  46.1 0.571    27     No
## 9      5   139    64    35  28.6 0.411    26     No
## 10     5   105    72    29  36.9 0.159    28     No
```

```
filter(p,npreg > 6 | age > 50)
```

```
## # A tibble: 49 × 8
##    npreg   glu    bp  skin   bmi   ped   age   type
##    <int> <int> <int> <int> <dbl> <dbl> <int> <fctr>
## 1      7   195    70    33  25.1 0.163    55    Yes
## 2      5    97    76    27  35.6 0.378    52    Yes
## 3      3   142    80    15  32.4 0.200    63     No
## 4      9   154    78    30  30.9 0.164    45     No
## 5      1   189    60    23  30.1 0.398    59    Yes
## 6     12    92    62     7  27.6 0.926    44    Yes
## 7     11   143    94    33  36.6 0.254    51    Yes
## 8      9   164    84    21  30.8 0.831    32    Yes
## 9     13   145    82    19  22.2 0.245    57     No
## 10     8   176    90    34  33.7 0.467    58    Yes
## # ... with 39 more rows
```

Manipulação de Dados com dplyr

Operações Básicas: arrange

```
dados2 <- arrange(dados1, col1, col2, ...)
```

- Reordena as linhas pela `col1`, depois pela `col2`, etc.

```
arrange(p,age)
```

```
## # A tibble: 200 × 8
##    npreg   glu    bp  skin   bmi   ped   age   type
##    <int> <int> <int> <int> <dbl> <dbl> <int> <fctr>
## 1      4    99    76    15  23.2 0.223    21     No
## 2      1   109    60     8  25.4 0.947    21     No
## 3      0   139    62    17  22.1 0.207    21     No
## 4      0   101    64    17  21.0 0.252    21     No
## 5      1    99    58    10  25.4 0.551    21     No
## 6      1    97    64    19  18.2 0.299    21     No
## 7      1   114    66    36  38.1 0.289    21     No
## 8      1    96    64    27  33.2 0.289    21     No
## 9      2   100    64    23  29.7 0.368    21     No
## 10     2   115    64    22  30.8 0.421    21     No
## # ... with 190 more rows
```

```
arrange(p,desc(npreg),age)
```

```
## # A tibble: 200 × 8
##    npreg   glu    bp  skin   bmi   ped   age   type
##    <int> <int> <int> <int> <dbl> <dbl> <int> <fctr>
## 1     14   175    62    30  33.6 0.212    38    Yes
## 2     14   100    78    25  36.6 0.412    46    Yes
## 3     13   145    82    19  22.2 0.245    57     No
## 4     12   151    70    40  41.8 0.742    38    Yes
## 5     12   140    85    33  37.4 0.244    41     No
## 6     12    92    62     7  27.6 0.926    44    Yes
## 7     12    88    74    40  35.3 0.378    48     No
## 8     12   140    82    43  39.2 0.528    58    Yes
## 9     12   121    78    17  26.5 0.259    62     No
## 10    11   143    94    33  36.6 0.254    51    Yes
## # ... with 190 more rows
```

Manipulação de Dados com `dplyr`

Operações Básicas: `select`

```
dados2 <- select(dados1, col1, col2, ...)
```

- Mostra os valores das colunas `col1`, `col2`, etc.

```
select(p,npreg,age)
```

```
## # A tibble: 200 × 2
##    npreg   age
## *  <int> <int>
## 1      5    24
## 2      7    55
## 3      5    35
## 4      0    26
## 5      0    23
## 6      5    52
## 7      3    25
## 8      1    24
## 9      3    63
## 10     2    31
## # ... with 190 more rows
```

```
select(p,bmi:type)
```

```
## # A tibble: 200 × 4
##      bmi   ped   age   type
## *  <dbl> <dbl> <int> <fctr>
## 1   30.2 0.364    24     No
## 2   25.1 0.163    55    Yes
## 3   35.8 0.156    35     No
## 4   47.9 0.259    26     No
## 5   26.4 0.133    23     No
## 6   35.6 0.378    52    Yes
## 7   34.3 0.336    25     No
## 8   25.9 0.655    24     No
## 9   32.4 0.200    63     No
## 10  43.3 1.224    31    Yes
## # ... with 190 more rows
```

```
select(p,starts_with("b"))
```

```
## # A tibble: 200 × 2
##       bp   bmi
## *  <int> <dbl>
## 1     68  30.2
## 2     70  25.1
## 3     82  35.8
## 4     76  47.9
## 5     60  26.4
## 6     76  35.6
## 7     58  34.3
## 8     50  25.9
## 9     80  32.4
## 10    78  43.3
## # ... with 190 more rows
```

Manipulação de Dados com `dplyr`

Operações Básicas: `mutate`

```
dados2 <- mutate(dados1, novacol1, novacol2, ...)
```

- Adiciona as colunas `novacol1`, `novacol2`, etc.. Não altera o conjunto de dados original `dados1`

```
mutate(p,xpto=npreg*bmi/age)


## Source: local data frame [200 x 9]
##
##    npreg glu bp skin  bmi   ped age type     xpto
## 1      5  86 68   28 30.2 0.364  24   No 6.291667
## 2      7 195 70   33 25.1 0.163  55  Yes 3.194545
## 3      5  77 82   41 35.8 0.156  35   No 5.114286
## 4      0 165 76   43 47.9 0.259  26   No 0.000000
## 5      0 107 60   25 26.4 0.133  23   No 0.000000
## 6      5  97 76   27 35.6 0.378  52  Yes 3.423077
## 7      3  83 58   31 34.3 0.336  25   No 4.116000
## 8      1 193 50   16 25.9 0.655  24   No 1.079167
## 9      3 142 80   15 32.4 0.200  63   No 1.542857
## 10     2 128 78   37 43.3 1.224  31  Yes 2.793548
## ..   ... ... ..  ...  ...   ... ...  ...      ...
```

Manipulação de Dados com `dplyr`

Composição de Operações Básicas

- Podemos usar a composição de funções para encadear um conjunto de operações.

```
select(filter(p,npreg > 4,age < 30),npreg,age,type)
```

```
## Source: local data frame [10 x 3]
##
##    npreg age type
## 1      5  24   No
## 2      5  28   No
## 3      6  27   No
## 4      5  25  Yes
## 5      6  29  Yes
## 6      5  29  Yes
## 7      6  24   No
## 8      6  27   No
## 9      5  26   No
## 10     5  28   No
```

Manipulação de Dados com dplyr

Composição de Operações Básicas (2)

- Contudo, a composição de funções pode tornar-se complexa

```
arrange(
        select(
               filter(
                      mutate(p,xpto=npreg*bmi/age),
                      xpto > 1.5,npreg > 4,age < 30),
               npreg,age,type),
        desc(age)
        )
```

```
## Source: local data frame [10 x 3]
##
##    npreg age type
## 1      6  29  Yes
## 2      5  29  Yes
## 3      5  28   No
## 4      5  28   No
## 5      6  27   No
## 6      6  27   No
## 7      5  26   No
## 8      5  25  Yes
## 9      5  24   No
## 10     6  24   No
```

Manipulação de Dados com dplyr

Composição de Operações Básicas (3)

- Em alternativa, podemos usar o operador de encadeamento %>%
- O conjunto de dados que resulta de cada operação vai sendo passado, implicitamente, como 1° argumento às operações seguintes.

```
mutate(p,xpto=npreg*bmi/age) %>% filter(xpto > 1.5, npreg > 4, age < 30) %>%
  select(npreg,age,type) %>% arrange(desc(age))
```

```
## Source: local data frame [10 x 3]
##
##    npreg age type
## 1      6  29  Yes
## 2      5  29  Yes
## 3      5  28   No
## 4      5  28   No
## 5      6  27   No
## 6      6  27   No
## 7      5  26   No
## 8      5  25  Yes
## 9      5  24   No
## 10     6  24   No
```

Manipulação de Dados com `dplyr`

Operações Básicas: `summarise`

```
dados2 <- summarise(dados1, sumF1, sumF2, ...)
```

- Sumaria as linhas de `dados1` usando as funções `sumF1`, `sumF2`, ...

```
summarise(p,varBmi=var(bmi),avgAge = mean(age),nAge=n_distinct(age))


## Source: local data frame [1 x 3]
##
##    varBmi avgAge nAge
## 1 37.5795  32.11   39
```

Manipulação de Dados com dplyr

Operações Básicas: group_by

```
dados2 <- group_by(dados1, crit1, crit2, ...)
```

- Cria grupos de linhas de dados de acordo com os critérios.
- Sobre esse grupos podem depois ser aplicadas funções.

```
group_by(p,type)
```

```
## Source: local data frame [200 x 8]
## Groups: type
##
##    npreg glu bp skin  bmi   ped age type
## 1      5  86 68   28 30.2 0.364  24   No
## 2      7 195 70   33 25.1 0.163  55  Yes
## 3      5  77 82   41 35.8 0.156  35   No
## 4      0 165 76   43 47.9 0.259  26   No
## 5      0 107 60   25 26.4 0.133  23   No
## 6      5  97 76   27 35.6 0.378  52  Yes
## 7      3  83 58   31 34.3 0.336  25   No
## 8      1 193 50   16 25.9 0.655  24   No
## 9      3 142 80   15 32.4 0.200  63   No
## 10     2 128 78   37 43.3 1.224  31  Yes
## ..   ... ... ..  ...  ...   ... ...  ...
```

```
group_by(p,type) %>%
  summarise(count = n(),
            avg_age = mean(age),
            avg_npreg = mean(npreg))
```

```
## Source: local data frame [2 x 4]
##
##   type count  avg_age avg_npreg
## 1   No   132 29.23485  2.916667
## 2  Yes    68 37.69118  4.838235
```

Manipulação de Dados com `dplyr`

Operações Básicas

- Mais detalhes sobre estas outras operações de manipulação de dados podem ser consultados em:

  Data Wrangling with dplyr and tidyr.