

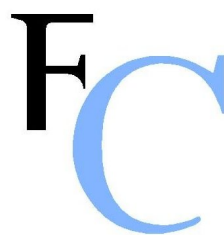
Importing Data Into R

L. Torgo

ltorgo@fc.up.pt

Departamento de Ciência de Computadores / Faculdade de Ciências
Universidade do Porto

Feb, 2021



FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Introduction

Data Sets

- The majority of data analysis tasks uses as source data sets stored in a tabular format
- A data set is a bi-dimensional structure (a table)
 - Rows represent observations of a phenomenon
 - Columns contain information obtained for each observation
- The R data structure used to store these tables is the *data frame*
- In the following slides we will see illustrations on how to import data stored in different formats/infra-structures into a R data frame

Internal R Data Sets

- Any R installation includes many data sets. The list can be obtained doing:

```
data()
```

- Each new package we install may also add new data sets for illustration purposes

```
data(package="DMwR") # data sets from a specific package
data(package = .packages(all.available = TRUE)) # all packages
```

- This type of data sets can be used/loaded using the function `data`

```
data(iris)
head(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
## 4           4.6         3.1         1.5         0.2   setosa
## 5           5.0         3.6         1.4         0.2   setosa
## 6           5.4         3.9         1.7         0.4   setosa
```

The RData file format

- Any data frame (in reality any R object) may be saved in a RData file

```
dat <- data.frame(x=rnorm(10), y=rnorm(10))
save(dat, file="exp.RData")
```

- These data may be loaded back into R later

```
rm(dat) # remove dat from the computer memory
dat     # confirming that it was deleted

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

load("exp.RData") # load it back from the file
head(dat)         # here it is again!

##           x           y
## 1 0.07611641 -0.5965060
## 2 -1.56715114 -0.9988497
## 3 -0.11878837  1.0428524
## 4 -0.46173060 -0.6761860
## 5 -1.41758109 -0.1412697
## 6  0.26774836  0.7264043
```

Importing data from text files

- Text files are a frequent way of storing and sharing data sets
 - Rows of the file usually correspond two rows of the data set
 - Values in the columns stored in a single row separated by some special character
- Text files are frequently the easiest way of importing data into R from other tools
- There are many ways to read text files in R. We will use the infra-structure provided by (extra) package `readr`

Data Sets in Text Files

CSV files

The values on each line are separated by commas

File “x.csv”

ID, Name, Age
23424, Ana, 45
11234, Charles, 23
77654, Susanne, 76

To read the data into a data frame:

```
library(readr)
dat <- read_csv("x.csv", col_types = "cci")
dat

## # A tibble: 3 x 3
##   ID      Name    Age
##   <chr> <chr>   <int>
## 1 23424 Ana       45
## 2 11234 Charles   23
## 3 77654 Susanne   76
```

Variations of the CSV format

On countries where the comma is used as a decimal separator it is common to use the semi-colon to separate values

Reading this into a data frame:

File “y.csv”

ID; Name; Grade
23424; Ana; 4,6
11234; Charles; 12,3
77654; Susanne; 15,9

```
dat <- read_csv2("y.csv", col_types = "ccd")

## Using ',' as decimal and '.' as grouping
## mark. Use read_delim() for more control.

dat

## # A tibble: 3 x 3
##   ID      Name      Grade
##   <chr> <chr>    <dbl>
## 1 23424 Ana        4.6
## 2 11234 Charles    12.3
## 3 77654 Susanne    15.9
```

Other text formats

File “z.txt”

ID	Name	Age	Phone
23424	Ana	40	???
11234	Charles	12	34567678
77654	Susanne	45	23435567

To read this data into a data frame we do:

```
dat <- read_table2("z.txt", na="???", col_types = "ccic")
dat

## # A tibble: 3 x 4
##   ID      Name      Age Phone
##   <chr> <chr>    <int> <chr>
## 1 23424 Ana        40 <NA>
## 2 11234 Charles    12 34567678
## 3 77654 Susanne    45 23435567
```

Summary on text formats

- Family of functions with similar syntax and parameters

- `read_table`, `read_table2`, `read_csv`, `read_csv2`, `read_delim`, etc.

- Some of the more relevant parameters

`col_types` indicates the column types

`col_names` indicates whether the first line contains the column names

`na` indicates the character of vector of characters that should be interpreted as unknown values

- The result of these functions is a tibble

- *character encodings* - potential source of problems (check parameter `locale` of the functions)

- Other functions : `read_lines`, `scan` on base R

Importing from Spreadsheets

Importing data from SpreadSheets

Method 1 - using CSV format

- One way of sending some data into R consists in saving the wanted spreadsheet data on a text file, for instance in format CSV
- And then, inside R, use some of the previously explained methods to import the data into a data frame from the saved text file

Importing data from SpreadSheets

Method 2 - through the *clipboard*

- When we want to import small data tables in a spreadsheet this is the easiest process

	A	B	C	D	E
1					
2					
3		ID	Nome	Nota	
4		45676	Ana	12.5	
5		65677	Carlos	3.75	
6		53455	Joana	17.3	
7					
8					
9					

Select the table and do
Edit+Copy

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

```
dat

##      ID  Nome  Nota
## 1 45676   Ana 12.50
## 2 65677 Carlos  3.75
## 3 53455  Joana 17.30
```

Importing data from SpreadSheets

Method 3 - through package *readxl*

- Package *readxl* includes the function `read_excel`

Example

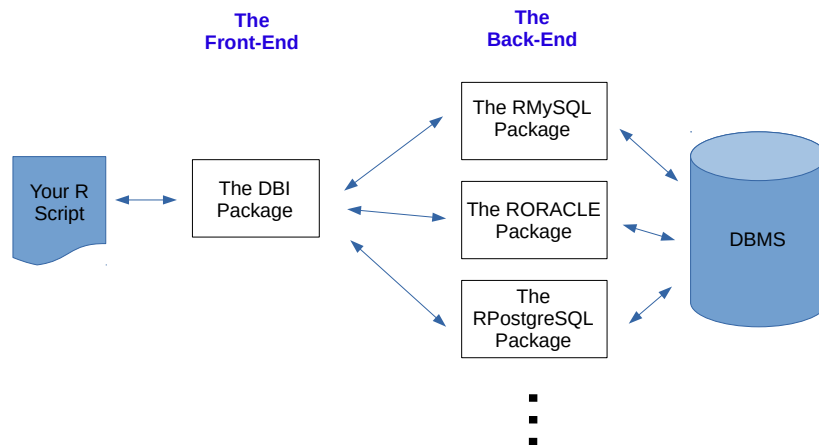
```
library(readxl)
fc <- "c:\\Documents and Settings\\xpto\\My Documents\\calc.xls"
dat <- read_excel(fc, sheet="MayValues")
dat2 <- read_excel(fc, sheet=2)
dat <- read_excel(fc, sheet="MayValues", range="C2:F24")
```

- More info and examples at
<https://readxl.tidyverse.org/>

Connecting with Data Bases

The DBI package

- DBI provides a front end interface to DBMS-specific drivers



Connecting with Data Bases

An Example with MySQL

```

library(DBI)
library(RMySQL)
drv <- dbDriver("MySQL") # Loading the MySQL driver
con <- dbConnect(drv, dbname="transDB", # connecting to the DBMS
                 username="myuser", password="mypasswd",
                 host="localhost")

# getting the results of a query as a data frame
data <- dbGetQuery(con, "SELECT * FROM clients")

dbDisconnect(con) # closing up stuff
dbUnloadDriver(drv)
  
```

Another Example with Results in Chunks

```
library(DBI)
library(RMySQL)
drv <- dbDriver("MySQL") # Loading the MySQL driver
con <- dbConnect(drv, dbname="transDB", # connecting to the DBMS
                 username="myuser", password="mypasswd",
                 host="localhost")

res <- dbSendQuery(con, "SELECT * FROM transactions")
while (!dbHasCompleted(res)) {
  # get the next 50 records on a data frame
  someData <- fetch(res, n = 50)
  # call some function that handles the current chunk
  process(someData)
}
dbClearResult(res) # clear the results set

dbDisconnect(con) # closing up stuff
dbUnloadDriver(drv)
```

Other Software

Other forms of data importation

- Data bases
 - R has interfaces to all major DBMS (packages DBI, RMySQL, ROracle, etc.)
- Other statistical software
 - Importing from Minitab, S-Plus, SPSS, Stata, SAS, etc.
 - Packages `foreign`, `Hmisc`
- Several other formats / software.
- More details / information in the manual *R Data Import/Export* that comes with R