8.1.2 Come analizzare i dati: R Leggere e scrivere dati

Insegnamento di Informatica

Elisabetta Ronchieri

Corso di Laurea di Economia, Universitá di Ferrara

I semestre, anno 2014-2015

Argomenti

Argomenti

Reading Data

R functions for reading data are:

- read.table(), read.csv() for reading tabular data;
- readLines() for reading lines of a text file;
- source() for reading in R code files (inverse of dump);
- dget() for reading in R code files (inverse of dput);
- load() for reading in saved workspaces;
- unserialize() for reading single R objects in binary form.

read.table()

The read.table() function has a few important arguments:

- file, the name of a file, or a connection;
- header, logical indicating if the file has a header line;
- sep, a string indicating how the columns are separated;
- colClasses, a character vector indicating the class of each column in the dataset;
- nrows, the number of rows in the dataset;
- comment.char, a character string indicating the comment character;
- skip, the number of lines to skip from the beginning;
- stringsAsFactors, coding character variables as factors.

Telling R all these things directly makes R run faster and more efficiently.

read.table()

For small to moderately sized datasets, you can usually call read.table without specifying any other arguments

```
> data <- read.table("data.txt")
```

R will automatically:

- skip lines that begin with a #;
- figure out how many rows there are (and how much memory needs to be allocated);
- figure what type of variable is in each column of the table.

However, set comment.char = "" if there are no commented lines in your file.

read.csv()

Read table built by using for example Excel.

```
> data <- read.csv("data.csv", header=TRUE, sep=",")
```

What to do in case of a very large dataset

It is useful to know a few things about your system.

- How much memory is available?
- What other applications are in use?
- Are there other users logged into the same system?
- What operating system?
- Is the OS 32 or 64 bit?

Considering a data.frame with 1,500,000 rows and 120 columns, all of which are numeric data, memory requirements is calculated as follows:

$$1,500,000 \times 120 \times 8 bytes/numeric = 1440000000 bytes$$

Argomenti

Writing Data

R functions for writing data are:

- write.table() for writing tabular data.
- writeLines() for writing lines in a text file.
- dump() for writing text representations of the R objects.
- dput() for writing an ASCII text representation of R objects.
- save() for writing R objects in file.
- serialize() for writing single R objects.

write.table()

Write and read data built by using data.frame type.

```
> m <- matrix(1:6, nrow=2, ncol=3)
> v <- c("A", "B")
> data1 <- data.frame(m,v)
> data1
  X1 X2 X3 v
1  1  3  5  A
2  2  4  6  B
> write.table(data1, file="data1.txt")
> data2 <- read.table("data1.txt")
> data2
  X1 X2 X3 v
1  1  3  5  A
2  2  4  6  B
```

Remember that *v* and *m* must have the same number of rows.

Content of the data1.txt file.

```
"X1" "X2" "X3" "v"
"1" 1 3 5 "A"
"2" 2 4 6 "B"
```

dump()

Multiple objects can be deparsed using the dump() function.

```
> x <- "dato"
> y <- data.frame(a = 1, b = "a")
> dump(c("x", "y"), file = "data.R")
```

Content of the data.R file.

```
x <-
"dato"
y <-
structure(list(a = 1, b = structure(1L, .Label = "a",
class = "factor")), .Names = c("a", "b"), row.names
= c(NA, -1L), class = "data.frame")
```

dump()

Multiple objects can be deparsed using the dump() function and read back in using source().

```
> x <- "dato"
> y <- data.frame(a = 1, b = "a")
> dump(c("x", "y"), file = "data.R")
> rm(x, y)
> source("data.R")
> y
  a b
1 1 a
> x
[1] "dato"
```

The rm() function removes the x and y objects.

Argomenti

Examples

Install the R "HSAUR" packages to get data about Forbes 2000 world leading companies.

```
> install.packages("HSAUR")
```

Dowload the vincentarelbundock Rdatasets repository
https://github.com/vincentarelbundock/Rdatasets
by using the following command on your local workspace.

```
$ git clone <repository name>.git
```

Read it by using read.table() [Everitt]:

```
> data <- read.table("Rdatasets/csv/HSAUR/Forbes2000.csv", ...
+ header=TRUE, sep=",")
```

Examples

Determine the variable names included in the data frame:

```
> names(data)
[1] "rank"      "name"      "country"   "category"
[5]  "sales"    "profits"   "assets"    "marketvalue"
```

Look at the structure of data:

```
> str(data)
'data.frame':   2000 obs. of  8 variables:
 $ rank       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ name       : Factor w/ 2000 levels "Aareal Bank",..: 438 747 100 659 311 219
 $ country    : Factor w/ 61 levels "Africa","Australia",..: 60 60 60 60 56 60
 $ category   : Factor w/ 27 levels "Aerospace & defense",..: 2 6 16 19 19 2 2
 $ sales      : num  94.7 134.2 76.7 222.9 232.6 ...
 $ profits    : num  17.85 15.59 6.46 20.96 10.27 ...
 $ assets     : num  1264 627 648 167 178 ...
 $ marketvalue: num  255 329 195 277 174 ...
```

Per ulteriori letture

- W. N. Venables, D. M. Smith and the R Core Team, *An Introduction to R*, July 2014, `http://cran.r-project.org/doc/manuals/R-intro.pdf`
- Vito M. R. Muggeo, Giancarlo Ferrara, *Il Linguaggio R: concetti introduttivi ed esempi*, 2005, `http://cran.r-project.org/doc/contrib/nozioniR.pdf`
- Josef Eschgfaller, *Programmare in R*, 2005, `http://cran.r-project.org/doc/contrib/Fondamenti-0405.pdf`
- Brian S. Everitt, Torsten Hothorn, *A Handbook of Statistical Analyses Using R*, `http://cran.r-project.org/web/packages/HSAUR/vignettes/Ch_introduction_to_R.pdf`