Programming for Data Science
I/O in R language

Marco Beccuti
Università degli Studi di Torino

Dipartimento di Informatica

Writing a file in R

- R provides a set of high level functions to write data into files:

  - write.table() is used to write data frames into formatted text files.
    A variable separator can be specified.

  - write.csv() is used to write data frames into comma separated variable files.

  - write.csv2() is used to write data frames into semicolon separated variable files.

  - save() is used to save datasets into a binary file.
    Data are stored in binary format (more compact!!).

Writing a file in R

- write.table() is used to write data frames into formatted text files ,

  write.table(x,file,col.names=TRUE,row.names=TRUE, sep=" ", dec=".", ...)

  - x : the object to be written;
  - file : the name of the file in which the data are stored;
  - col.names : if TRUE column names are stored;
  - row.names : if TRUE row names are stored;;
  - sep : the field separator character;
  - dec : the character used for decimal points;
  - ... : optional arguments.

  $>$ *write.table*(b, "./*example.txt*", *col.names* = *TRUE*, *row.names* = *TRUE*, *sep* = "!")

  $>$ *write.table*(b, "./*example.txt*", *col.names* = *FALSE*, *row.names* = *FALSE*, *sep* = ",")

Writing a file in R

- write.csv() is used to write data frames into formatted text files ,

  write.csv(x,file,col.names=TRUE,row.names=TRUE, sep=",", dec=".", ...)

  - x : the object to be written;
  - file : the name of the file in which the data are stored;
  - col.names : if TRUE column names are stored;
  - row.names : if TRUE row names are stored;;
  - sep : the field separator character;
  - dec : the character used for decimal points;
  - ... : optional arguments.

  $> write.csv(b, "./example.txt", col.names = TRUE, row.names = TRUE)$

  $> write.csv(b, "./example.txt", col.names = FALSE, row.names = FALSE)$

Writing a file in R

- write.csv2() is used to write data frames into formatted text files ,

  write.csv2(x,file,col.names=TRUE,row.names=TRUE, sep=";",dec=",", ...)

x :	the object to be written;
file :	the name of the file in which the data are stored;
col.names :	if TRUE column names are stored;
row.names :	if TRUE row names are stored;;
sep :	the field separator character;
dec :	the character used for decimal points;
... :	optional arguments.

  $> write.csv2(b, \text{``./example.txt''}, col.names = TRUE, row.names = TRUE)$

  $> write.csv2(b, \text{``./example.txt''}, col.names = FALSE, row.names = FALSE)$

Writing a file in R

- save() writes an external representation of R objects to the specified file,

  save(...,file, ...)

          ... : a list of objects to be saved;
        file : the name of the file in which the data are stored;
          ... : optional arguments.

  $> save(b, c, file = "./example.data")$

Writing a file in R

- R provides a function to write text lines into a file

  writeLines(text, con = stdout(), sep = "\n", ...)

        text : a character vector;

        con : a connection object or a character string;

        sep : a string to be written to the connection after each line of text.

  $>$ tex $=$ c("line1", "line2", "line3")

  $>$ writeLines(tex, "./example.txt")

Writing a file in R

- How can we append a new text line into a file?
  We have to open the file in append mode using:
  file(description = "", open = "", ...)
    description : a character vector;
         open : a description of how to open the connection:

                       "rt" : open for reading in text mode;
                       "wt" : open for writing in text mode;
                       "at" : open for appending in text mode;
                       "rb" : open for reading in binary mode;
                       "wb" : open for writing in binary mode;
                       "ab" : open for appending in binary mode;

> tex = c("line1","line2","line3")

> con = file("./example.txt","at")

> writeLines(tex, con)

> close(con)   **file must be always closed**

Input from a file in R

- R provides a set of high level functions to read data from files:

  - read.table() is used to read data frames from formatted text files.
    A variable separator can be specified.

  - read.csv() is used to read data frames from comma separated variable files.

  - read.csv2() is used to read data frames from semicolon separated variable files.

  - load() is used to reload datasets written with the function save().
    Data are stored in binary format (more compact!!).

Input from a file in R

- read.table() reads a file in table format and creates a data frame from it,

  read.table(file,header=FALSE, sep=" ", dec=".", stringAsFactors=FALSE ...)

file :	the name of the file in which the data are stored;
header :	a logical value indicating whether the file contains the names of the variables as its first line;
sep :	the field separator character;
dec :	the character used for decimal points;
stringAsFactors :	logical: should character vectors be converted to factors?;
row.names :	it can be a vector giving the actual row names, or a single number giving the column of the table which contains the row name;
... :	optional arguments;

  $> d = read.table("./example.txt", header = TRUE, sep = "!")$

  $> b = read.table("./example1.txt", header = FALSE, sep = " ")$

Input from a file in R

- read.csv() reads a file in table format and creates a data frame from it,

  read.csv(file,header=FALSE, sep=",", dec=".",...)

file :	the name of the file in which the data are stored;
header :	a logical value indicating whether the file contains the names of the variables as its first line;
sep :	the field separator character;
dec :	the character used for decimal points;
stringAsFactors :	logical: should character vectors be converted to factors?;
row.names :	it can be a vector giving the actual row names, or a single number giving the column of the table which contains the row name
... :	optional arguments;

  $> d = read.csv(\text{``}./example.txt\text{''}, header = TRUE)$

  $> b = read.csv(\text{``}./example1.txt\text{''}, header = FALSE)$

Input from a file in R

- read.csv2() reads a file in table format and creates a data frame from it,

  read.csv2(file,header=FALSE, sep=";", dec=".", ...)

    file : the name of the file in which the data are stored;

    header : a logical value indicating whether the file contains the names of the variables as its first line;

    sep : the field separator character;

    dec : the character used for decimal points;

  stringAsFactors : logical: should character vectors be converted to factors?;

    row.names : it can be a vector giving the actual row names, or a single number giving the column of the table which contains the row name

    ... : optional arguments;

  $> d = read.csv2(\text{``}./example.txt\text{''}, header = T)$

  $> b = read.csv2(\text{``}./example1.txt\text{''}, header = F)$

Input from a file in R

- load() reload datasets written with the function save().

  load(file, ...)

          File : the name of the file in which the data are stored;
  verbose = FALSE : if TRUE item names are printed;
        ... : optional arguments.

  $> load("./example.data")$

  $> load("./example.data", verbose = T)$
  Loading objects :
  m

Input from a file in R

- R provides a function to read some or all text lines from a file

  readLines(con = stdin(), n = -1L, ok = TRUE, warn = TRUE, ...)

  con : a connection object or a character string;

  n : the number of lines to read. Negative values mean all the lines until the end of connection;

  ok : if TRUE returns an error when less then n lines are read;

  warn : if TRUE returns a warning when final EOL is missing.

  > line = readLines("./example.txt", n = 1, warn = FALSE)   **read one line**

  > lines = readLines("./example.txt", n = −1)   **read all lines storing in a vector**

Download and install a package in R

- In R, a package can be downloaded and installed from CRAN-like repositories or from local files;

install.packages(pkgs,rep=getOption("repos"))

pkgs : character vector of the names of packages to be downloaded;
rep : base URL(s) of the repositories to use.
Default CRAN repository.
... : optional arguments;

> install.packages("KDE")

> install.packages(path_to_file, repos = NULL, type = "source")

Load a package in R

- In R a package must be loaded before being used;

library(package,....)
      package : name of the package to be loaded;
        ... : optional arguments;

> *library*(*MASS*)

> *library*()   **see all packages installed**

Save and Load the R workspace

- In R the workspace can be saved and loaded using:

  save.image(file =".RData")
  load(file =".RData")

  > *save.image*(*file* = *"OutputWorkspace"*)

  > *load*(*file* = *"OutputWorkspace"*)

How to invoke a system command

- In OS command can be executed using:

  system2(command, args = character(), wait = TRUE, ...)
  - command : the system command to be invoked, as a character string;
  - args : a character vector of arguments to command.
  - wait : a logical indicating whether the R interpreter should wait for the command to finish, or run it asynchronously.

  $> system2(command = ``dir", args = `` * .txt")$

Exercises on input/ouput

- Save in the textual file "example.txt" the data frame trees;

- Load the data frame stored in the textual file "example.txt";

- Save in the textual file "example.csv" the data frame trees using ";" as variable separator;

- Load the data frame stored in the textual file "example.csv";

- Create a matrix with 1,000,000 elements and save it using "write.table" and "save".

Exercises on input/ouput

- Save in the textual file "example.txt" the data frame trees;

  $>$ *write.table(trees, file = "./example.txt")*

- Load the data frame stored in the textual file "example.txt";

  $>$ *D = read.table(file = "./example.txt")*

- Save in the textual file "example.csv" the data frame trees using ";" as variable separator;

  $>$ *write.table(trees, file = "./example.csv", sep = ";")*

- Load the data frame stored in the textual file "example.csv";

  $>$ *K = read.table(file = "./example.csv", sep = ";")*

Exercises on input/ouput

- Create a matrix with 1,000,000 elements and save it using "write.table" and "save".

  $>$ *m = matrix*(1 : 1000000, *ncol* = 100000)

  $>$ *write.table*(*m*, *file* = "./*example.csv*")

  $>$ *save*(*m*, *file* = "./*example.csv*")