Chapter 1

Introduction

1.1 R Download and Installation

- R web page (http://www.r-project.org/), select Download - CRAN → Select a mirror → Select in the first frame, named 'Download and Install R', your operating system.
- Windows: → select 'base' → click on 'R-?.?.?-win32.exe'.
- Under Linux Ubuntu:

  ```
  sudo apt-get install r-base
  ```

- Alternatively, go to http://cran.r-project.org/doc/manuals/R-admin.html and follow description there.
- A package is a collection of functions and programs that can be used within R. To install new packages type from R command line:

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

  See http://mirrors.dotsrc.org/cran/web/packages/index.html for an alphabetic list of packages and http://mirrors.dotsrc.org/cran/web/views/ for a thematic organization of the packages.

- Package `Rcmdr` provides a graphical interface to R.
- R documentation: see http://www.sbtc.ltd.uk/freenotes.html: 'Getting Started in R' by Saghir Bashir, and the references under the R link http://cran.r-project.org/manuals.html.
- Emacs users can find a mode for R at http://ess.r-project.org/.

1.2 Basic commands

See help page for details on all commands listed here.

1.2.1 System, help and documentation commands

- `R` starts R from command line.
- `library(Rcmdr)` loads the `Rcmdr` package and starts graphical interface.
- `q()` quits your R session.
- `ls()` provides a list of objects in the current R workspace.
- `options(width=120)` determines the position of the line break in R output.
- `?plot` a question mark followed by the name of the function opens the help page relative to that function.
- `help.start()` opens a browser with documentation.
- `example(plot)` calls one or more examples implemented for the function.
- `demo(package.name)` calls a demonstration for the functionality of the defined package.
- `vignette(package="packagename",topic="name")` opens a pdf document on the package, if provided by the maintainer.

1.2.2 Data and operations

- `read.table()` and `write.table()` read and write from text file.
- `load()` and `save()` load and save R objects.
- `source("myscript.r")` loads and executes an R script.
- `^`, `%%`, `%/%` operators for power, modulus and integer part of the division, respectively.
- `c()` function used to collect objects together into a vector, example: `x <- c(1,2,3)`

  ```
  > c(A1 = 1, list(A2 = 1))

  $A1
  [1] 1
  $A2
  [1] 1
  ```

- `vector()`, `matrix()`, `array()`, `data.frame()`, `list()` data structures. Tests or coercions can be done with `is.data.frame()`, `as.data.frame()`, respectively
- `integer()` `double()` data types. Can be queried or coerced with `is.integer()` and `as.integer()`
- `str()` `head()` `tail()` compactly displays the structure, head and tail of an arbitrary R object and a data.frame, respectively.

- `mean()`. `median()`, `sum()`, `var()`, `summary()`, `interquartile()` `range()` compute sample statistics.

- `factor()` offers an alternative way of storing character data. For example a *factor* can have four elements and two *levels*:

  ```
  > algorithms <- c("greedy", "grasp", "greedy", "grasp")
  > algorithms
  ```

  ```
  [1] "greedy" "grasp"  "greedy" "grasp"
  ```

  ```
  > algorithms <- factor(algorithms)
  > algorithms
  ```

  ```
  [1] greedy grasp  greedy grasp
  Levels: grasp greedy
  ```

- Generate sequences of integers by:

  ```
  > 1:12
  ```

  ```
   [1]  1  2  3  4  5  6  7  8  9 10 11 12
  ```

  ```
  > seq(1, 21, by = 2)
  ```

  ```
   [1]  1  3  5  7  9 11 13 15 17 19 21
  ```

  ```
  > rep(3, 12)
  ```

  ```
   [1] 3 3 3 3 3 3 3 3 3 3 3 3
  ```

- Generate factors by specifying the pattern of their levels

  ```
  > gl(2, 8, labels = c("Control", "Treat"))
  ```

  ```
   [1] Control Control Control Control Control Control Control Control Treat
  [10] Treat   Treat   Treat   Treat   Treat   Treat   Treat
  Levels: Control Treat
  ```

- `expand.grid()` creates a data frame from all combinations of factors

  ```
  > (table <- expand.grid(algorithm = algorithms, instance = c("A",
       "B")))
  ```

  ```
    algorithm instance
  1    greedy        A
  2     grasp        A
  3    greedy        A
  4     grasp        A
  5    greedy        B
  6     grasp        B
  7    greedy        B
  8     grasp        B
  ```

- paste(), substr() **strsplit()** work with strings. The first concatenates, the second returns substrings within two positions, the third splits strings. Example:

```
> colors <- c("red", "yellow", "green")
> paste(colors, "flowers")

[1] "red flowers"    "yellow flowers" "green flowers"

> paste("several ", colors, "s", sep = "")

[1] "several reds"    "several yellows" "several greens"

> paste("I like", colors, collapse = ", ")

[1] "I like red, I like yellow, I like green"

> substr(colors, 1, 2)

[1] "re" "ye" "gr"

> unlist(strsplit("a.b.c", "\\."))

[1] "a" "b" "c"


> sub("(.*)-(.*)-(.*)", "\\1", "a-b-c")

[1] "a"
```

1.2.3 Graphics

- plot(), lines(), points(), curve(), hist(), barplot(), boxplot(), graphics functions from the base installation

- par() lists and changes graphic setting

- colors() the colors available in R

- Graphics are device independent. Type ?device to see on which device they can be printed and how.

- dev.copy(dev=pdf,file='Rplot.pdf') copies the graphic in a pdf file. Remember to close the pipeline with dev.off()

- lattice and ggplot two packages for multivariate conditional plots. Try demo() on them for a demonstration of their facilities. The package lattice is thoroughly explained in reference [1].

1.2.4 Data Manipulation and Reshaping Data

- stack, unstack, reshape and merge are useful functions for rearragging data.

```
> table$res <- runif(8, 0, 1)
> reshape(table, timevar = "algorithm", idvar = "instance", direction = "wide")

  instance res.greedy res.grasp
1        A     0.0649     0.866
5        B     0.8833     0.385
```

```
> tab <- data.frame(instance = c("A", "B"), opt = c(1, 2))
> merge(table, tab, by.x = "instance", by.y = "instance")

  instance algorithm    res opt
1        A    greedy 0.0649   1
2        A     grasp 0.8659   1
3        A    greedy 0.1856   1
4        A     grasp 0.5507   1
5        B    greedy 0.8833   2
6        B     grasp 0.3849   2
7        B    greedy 0.5419   2
8        B     grasp 0.0698   2
```

- Function melt from package reshape:

```
> L <- c(t1 = list(table), t2 = list(table))
> str(L, max.level = 2)

List of 2
 $ t1:'data.frame':        8 obs. of  3 variables:
  ..$ algorithm: Factor w/ 2 levels "grasp","greedy": 2 1 2 1 2 1 2 1
  ..$ instance : Factor w/ 2 levels "A","B": 1 1 1 1 2 2 2 2
  ..$ res      : num [1:8] 0.0649 0.8659 0.1856 0.5507 0.8833 ...
  ..- attr(*, "out.attrs")=List of 2
 $ t2:'data.frame':        8 obs. of  3 variables:
  ..$ algorithm: Factor w/ 2 levels "grasp","greedy": 2 1 2 1 2 1 2 1
  ..$ instance : Factor w/ 2 levels "A","B": 1 1 1 1 2 2 2 2
  ..$ res      : num [1:8] 0.0649 0.8659 0.1856 0.5507 0.8833 ...
  ..- attr(*, "out.attrs")=List of 2

> require(reshape)
> melt(L)

   algorithm instance variable  value L1
1     greedy        A      res 0.0649 t1
2      grasp        A      res 0.8659 t1
3     greedy        A      res 0.1856 t1
4      grasp        A      res 0.5507 t1
5     greedy        B      res 0.8833 t1
6      grasp        B      res 0.3849 t1
7     greedy        B      res 0.5419 t1
8      grasp        B      res 0.0698 t1
```

```
9     greedy        A      res 0.0649 t2
10     grasp        A      res 0.8659 t2
11    greedy        A      res 0.1856 t2
12     grasp        A      res 0.5507 t2
13    greedy        B      res 0.8833 t2
14     grasp        B      res 0.3849 t2
15    greedy        B      res 0.5419 t2
16     grasp        B      res 0.0698 t2
```

- `which()` returns the index of an element in a vector.

  Example: `which(!(colnames(table) %in% c("algorithm","instace")))`

- `which.min()` returns the index of the minnmal element

- drop unused levels in factors

  ```
  CHEUR.01<-subset(CHEUR, variability!="no")
  CHEUR.01$variability <- CHEUR.01$variability[drop=TRUE]
  ```

- For rank transformations within instances:

  ```
  D<-CHEUR.01
  D$rank <- D$res
  split(D$rank, D$inst) <- lapply(split(D$res, D$inst), rank)
  tapply(D$rank, D$alg, median)
  ```

- package `xtable` provides a function to convert an R object into a LaTeX or an HTML table.

  ```
  > library(xtable)
  > xtable(table)
  ```

  ```
  % latex table generated in R 2.9.1 by xtable 1.5-5 package
  % Mon Aug 31 19:41:09 2009
  \begin{table}[ht]
  \begin{center}
  \begin{tabular}{rllr}
    \hline
   & algorithm & instance & res \\
    \hline
  1 & greedy & A & 0.64 \\
    2 & grasp & A & 0.91 \\
    3 & greedy & A & 0.83 \\
    4 & grasp & A & 0.03 \\
    5 & greedy & B & 0.98 \\
    6 & grasp & B & 0.97 \\
    7 & greedy & B & 0.63 \\
    8 & grasp & B & 0.31 \\
     \hline
  \end{tabular}
  \end{center}
  \end{table}
  ```

- `Sweave` is a tool that allows to embed the R code in latex documents[2, 3]. It implements the literate programming concept.

1.3 Development Commands

- `.libPaths()` to check which paths are considered for loading libraries
- `missing()` tests whether a value was specified as an argument to a function.
- `stopifnot()` stops if one of a list of conditions is not true reporting an error with the first violated condition.
- `debug()` debugs a function. `Q` to exit debug mode.
- `sessionInfo()` prints version information about R and attached or loaded packages (see also `Sys.getlocale()`)
-
  ```
  > Sys.getpid()

  [1] 24633

  > getAnywhere("friedman.test")

  A single object matching 'friedman.test' was found
  It was found in the following places
    package:stats
    namespace:stats
  with value
  function (y, ...)
  UseMethod("friedman.test")
  <environment: namespace:stats>

  > methods("friedman.test")

  [1] friedman.test.default* friedman.test.formula*
     Non-visible functions are asterisked
  ```

- `prompt, promtData, package.skeleton` create documentation for functions, data sets and packages, respectively.
- to interrupt where a problem arise, `options(error=stop)` `options(error=recover)`, or to turn wanrings into erros `options(warn=2)`.
- `debug` to proceed step by step
- `browser()` to stop exectution at that point and open debug mode
- `if ( interactive() ) { ANSWER <- readline("Continue?") }` stops and asks whether to continue
- `R CMD build package`, `R CMD check package.tgz`, `R CMD INSTALL package.tgz`

1.4 Memory issues

- `.Machine` and `.Platform` variables holding information on the numerical characteristics of the machine and information on the platform on which R is running. Consult these for information such as the largest double or integer and the machine's precision. See also `?"Memory-limits"`

- `object.size()` provides an estimate of the memory that is being used to store an R object.

- `gc(verbose=TRUE)` causes a garbage collection to take place and prints memory usage statistics

Bibliography

[1] Deepayan, S.: Lattice Multivariate Data Visualization with R. Springer, New York (2007), iSBN 978-0-387-75968-5 (Cited on page 5.)

[2] Leisch, F.: Sweave: Dynamic generation of statistical reports using literate data analysis pp. 575–580 (2002), http://www.stat.uni-muenchen.de/~leisch/Sweave, iSBN 3-7908-1517-9 (Cited on page 7.)

[3] Leisch, F.: Sweave user manual (2008), http://www.statistik.lmu.de/~leisch/Sweave/ (Cited on page 7.)