

Oulun yliopiston matemaattisten tieteiden laitos/tilastotiede

R-ohjelman käyttö tilastollisissa analyyseissä

Tilastotieteen peruskurssi II/perusmenetelmät II, kl 2003

Luento 1: Johdatus R-ohjelmointikieleen

Ke 12.3.2003 klo 14-16 KE1139

Esa Läärä

esa.laara@oulu.fi <http://stat.oulu.fi/laara>

What is R?

- Statistical computing environment
- Both a software ("package") and a programming language
- Developed by a community of volunteers
- Freely distributed and used
- Source code open
- Available for many platforms, eg. Windows, Linux, Unix

See <http://cran.r-project.org/> to download and for further information.

R contains versatile and flexible facilities for

- data handling and storage
- statistical analysis
- graphical presentation
- numerical mathematics & simulation
- scientific programming

R as a programming language

- dialect of the S language (S-Plus)
- superficially like C or C++
- vectorized arithmetics
- object-oriented
- extendable with user-defined functions

R as a statistical program

- large repertory of basic and advanced methods
- easy to expand and tailor to specific needs
- deals simultaneously with different objects or data structures — not just a single data matrix
- results of analysis saved as objects and readily available for further processing
- parsimonious output listing.

To learn more about R

Dalgaard, P. (2002). *Introductory Statistics with R*. Springer, New York.

Oksanen, J. (2002). *R: Opas ekologeille*. Biologian laitos, Oulun yliopisto. <http://cc.oulu.fi/jarioksa/opetus/>

Kim, H.-Y. (2002). *Introduction to R*. A course (2 cu) at the Department of Mathematical Sciences, University of Oulu. <http://cc.oulu.fi/hyon/r.html>


Running R

- Interactive but not mouse-driven!
- *Commands* typed from keyboard
- Execution of tasks
 - evaluation of *expressions* contained in commands
 - based on calls of *functions*

Difficult to learn & slow to use?

- Maybe, but flexibility is rewarding in the long run

Running R on Windows

- Start by double-clicking the R-icon.
- *Console window*:
 - command lines to be typed after prompt '>',
 - prompt '+' to continue an incomplete command line,
 - output follows a completed command requesting it,
 - arrow key  leads to previous command lines.
- Menu bar for a few useful pull-down menus.
- `ls()`: objects in the memory.
- On-line help, also in HTML form.

A couple of things before starting

- Names of *objects* (e.g. *vectors* for "variables")
 - always start with a letter A--Z or a--z;
lower case separate from upper case, e.g. 'x' vs 'X'
 - integers 0--9 and '.' (dot) allowed after 1st letter
but not '_' (underline, see below).
- *Assignment operator* '<-':
 - assigns a value to an object, e.g.

```
> A <- 7      # object A gets 7 as its value, or
```
 - also '=' and '_' allowed.

Vectors and their generation

Vector = ordered set of elements of same *mode*
(e.g. real numbers)

- basic data structure in R
- values of elements can be assigned by function `c` (*concatenate*), e.g. vector `x` with 4 elements 1, 2, 4, 7:

```
> x <- c(1,2,4,7)  
  
> x      # the elements of vector x are printed  
[1] 1 2 4 7
```
- function `seq()` generates regular sequences
- function `rep()` replicates same element(s)

Mathematics – vectorized arithmetics

- basic operations $+$, $-$, $*$, $/$, $^$ for vectors of same *length*
($\text{length}(x)$ = number of elements in x)

⇒ result: a new vector whose elements are outcomes of the operation on the corresponding elements in operand vectors

- different lengths: recycle elements of the shorter vector
- common functions: $\text{sqrt}(x)$, $\text{exp}(x)$, $\text{log}(x)$, $\text{sin}(x)$, ...
operate similarly on each element of the argument vector x

Probability distributions

For common distributions used in statistics (normal, Poisson, binomial, gamma, ...) tools available for

- probability (density) function, e.g. `dnorm()`
- cumulative distribution function, e.g. `pnorm()`
- quantile function, e.g. `qnorm()`
- random number generator, e.g. `rnorm()`

Graphics in R

- versatile, flexible, high quality
- various *high-level* graphic functions available
- adding items (points, lines, text, ...) to an existing graph possible by *low-level* plotting commands
- `par`: fine tuning by *graphic parameters* (67 of them!), e.g. `pch`, `lty`, `mfrow`, `xlim`, `ylim`, `col`
- interactive drawing: `locator()` function
- graphs can be saved in various formats, e.g. `.pdf`, `.ps`, `.png`, `.bmp`, `.jpg`

Some high-level graphic commands

- `plot(x,y)`: an all-purpose tool for e.g. index plot, bivariate scatterplot, line diagram, step function
- `stripchart()`, `boxplot()`, `stem()`, `hist()` for univariate distribution of a continuous variable
- `barplot()`, `dotchart()`, `pie()` for categorical data

Some useful low-level commands

- `points()`, `lines()`, `abline()`, `polygon()`
- `text()`, `legend()`, `title()`

Description of univariate data

Example: Random blood glucose levels (mmol/l) obtained from a group ($n = 40$) of 1st year medical students:

4.7, 3.6, 3.8, 2.2, 4.7, ..., 4.3, 6.0 .

(1) Data entry from keyboard into a vector Y

```
> Y <- scan() # scan() is a handy alternative to c(...)
```

```
1: 4.7 3.6 3.8 2.2 4.7 4.1 3.6 4.0 4.4 5.1 4.2 4.1 4.4 5.0
```

```
15: 3.7 3.6 2.9 3.7 4.7 3.4 3.9 4.8 3.3 3.3 3.6 4.6 3.4
```

```
29: 4.5 3.3 4.0 3.4 4.0 3.8 4.1 3.8 4.4 4.9 4.9 4.3 6.0
```

```
41:
```

```
Read 40 items
```

Description of univariate data (cont'd)

(2) Some basic operations on vector Y

```
> N <- length(Y) ; N # number of elements of Y
```

```
[1] 40
```

```
> sum(Y) # sum of the elements
```

```
[1] 162.2
```

```
> range(Y) # minimum and maximum values of Y
```

```
[1] 2.2 6.0
```

```
> which(Y==min(Y)) # which element contains the minimum?
```

```
[1] 4
```

```
> which(Y==max(Y)) # which element contains the maximum?
```

```
[1] 40
```

Description of univariate data (cont'd)

(3) Measures of location and dispersion

```
> mean(Y) # arithmetic mean or average
```

```
[1] 4.055
```

```
> var(Y) ; sd(Y) # variance & standard deviation
```

```
[1] 0.4876667
```

```
[1] 0.6983313
```

```
> summary(Y) # mean and some quantiles
```

```
Min.  1st Qu.  Median Mean 3rd Qu.  Max.
```

```
2.200 3.600 4.000 4.055 4.525 6.000
```

```
> quantile(Y,0.1) # 1st decile or 10% quantile
```

```
10%
```

Graphical displays of univariate data

```
> stem(Y) # stem-and-leaf diagram
```

```
The decimal point is at the |
```

```
2 | 2
2 | 9
3 | 333444
3 | 6666778889
4 | 00011123444
4 | 56777899
5 | 01
5 |
6 | 0
```

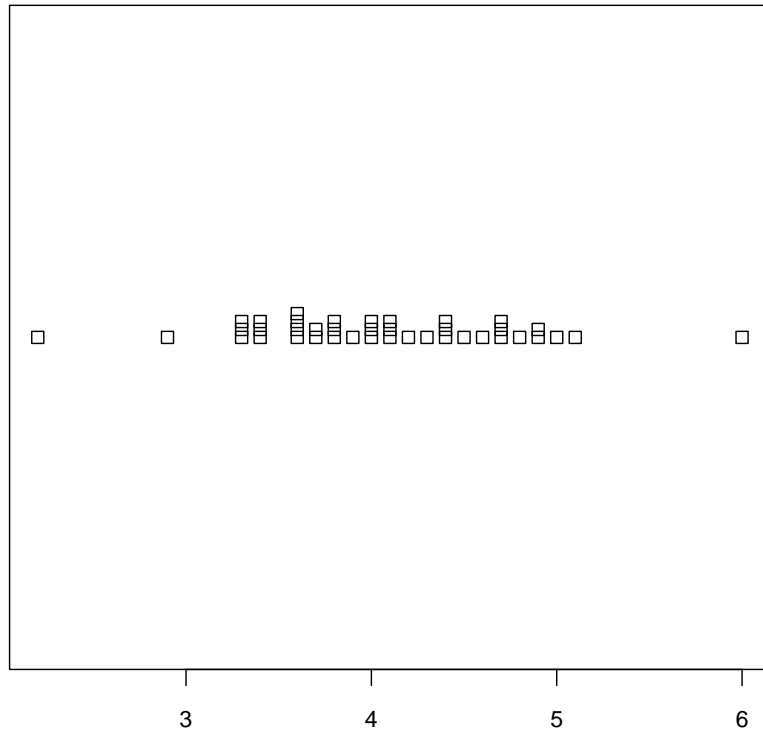
```
> stripchart(Y, method="stack", main="Stripchart of Y")
```

```
> boxplot(Y, main="Boxplot of Y") ; hist(Y)
```

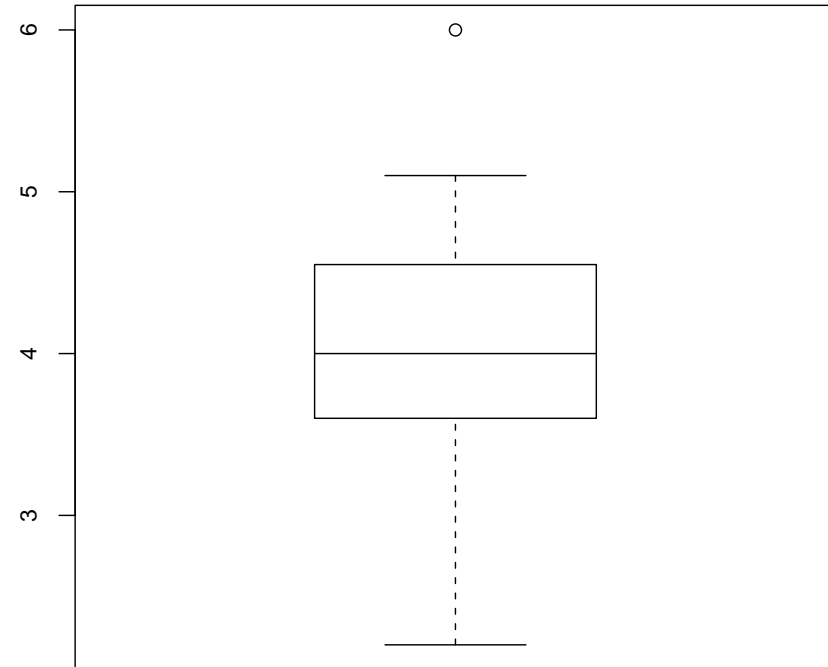
```
> plot(sort(Y), (1:N)/N, type="s", main="Cumulative frequency  
+ function of Y")
```

Graphical displays of univariate data (cont'd)

Stripchart of Y

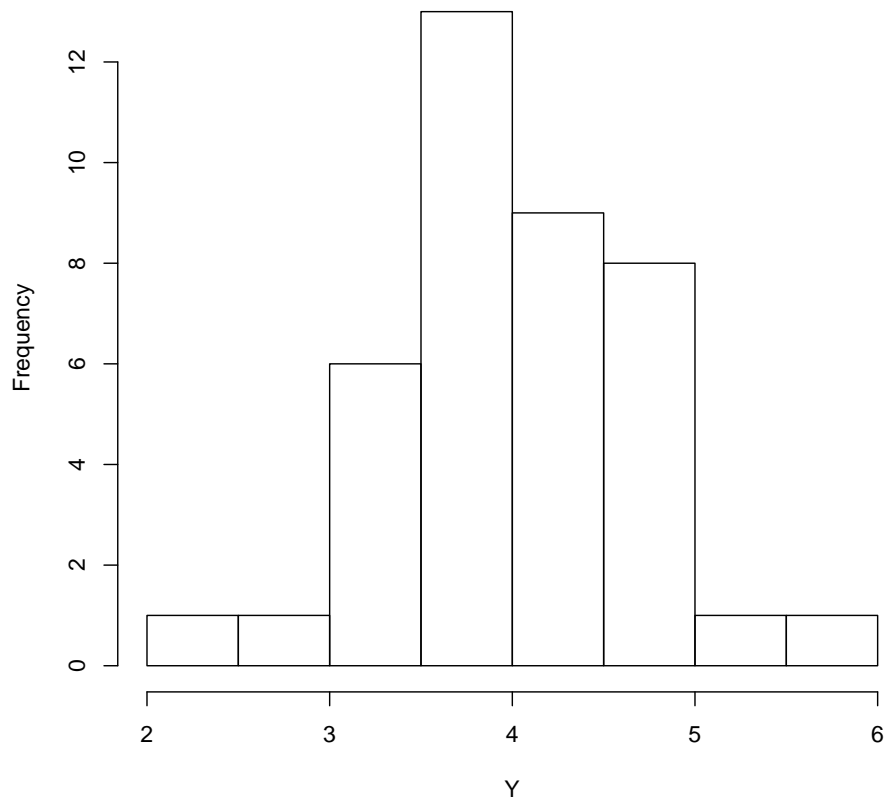


Boxplot of Y

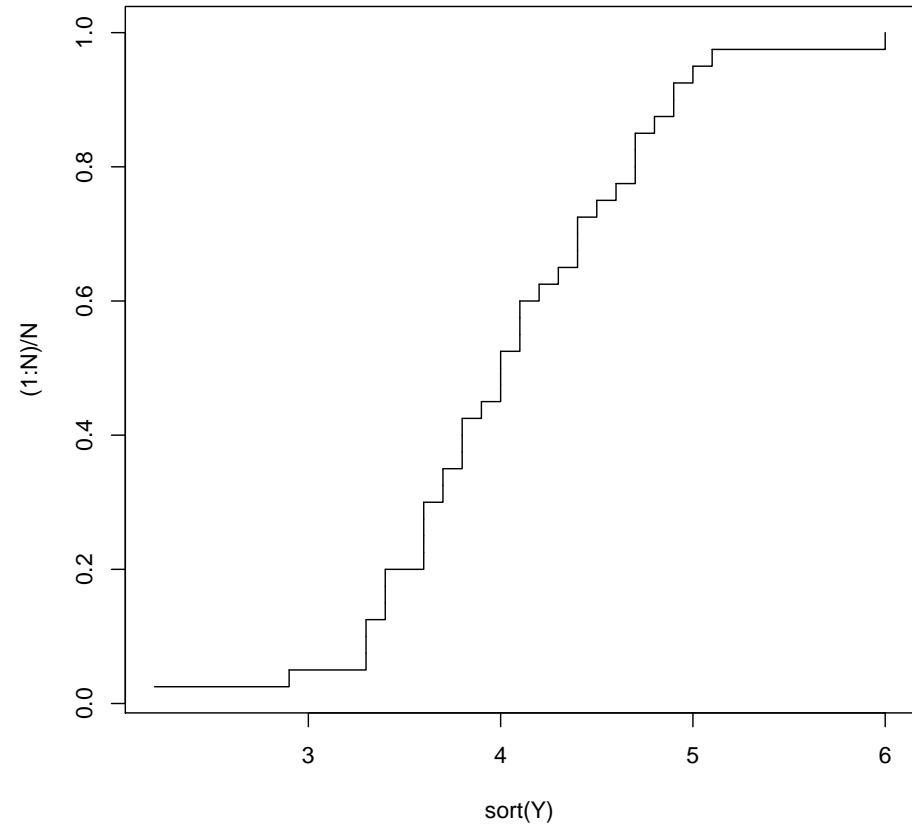


Graphical displays of univariate data (cont'd)

Histogram of Y



Cumulative frequency function of Y



Tabulation

Some functions to form and manipulate tables

- `table(c1, c2)` produces simple contingency tables
- `ftable(c1, c2, c3)` "flat" contingency tables
- `xtabs()` provides more elaborate tabulation features
- `apply()` for e.g. calculating margins in a cont. table
- `sweep()` for e.g. calculating percentages in table cells
- `tapply(v,c,f)` tabulates values of function `f` applied to values of variable (vector) `v` in categories of factor `c`

Statistical analyses in R

- Package `ctest`:
classical tests and confidence intervals for simple settings
- Function `lm`:
linear regression model for continuous outcome
- Function `glm`: for generalized linear models
- Several packages for special purposes,
e.g. `multiv`, `survival`, `dna`, ...

Different data structures and other objects

- `vector` (ordered set of similar elements)
- `factor` (categorical variable with levels)
- `array` (table)
- `matrix` (2-dimensional numeric table)
- `data.frame` ("data matrix")
- `ts` (time series)
- `list` (sequence of different objects)
- `function`, `expression`, ...

Mode of object – basic type of its elements

- `numeric` (real numbers)
- `complex` (complex numbers)
- `logical` (Boolean: `TRUE` or `FALSE`)
- `character` (ASCII Strings)

Each `vector`, `factor`, `array` and `matrix` can be of one mode only, but `data.frame`, `list`, etc. can have elements of several modes.

Logical vectors

- Used particularly in selecting subsets of observations and in programming.
- Elements are either TRUE or FALSE (or NA = value missing).
- *Comparison operators* for numeric vectors:
< (less than), > (greater than), <= (less than or equal to), >= (greater than or equal to), == (equal to), != (not equal).
- *Relational expression* involves a comparison operator

```
> x      # recall the elements of vector x
[1] 1 2 4 7
> x < 7   # this is TRUE for elements < 7
[1] TRUE TRUE TRUE FALSE
```

Logical vectors (cont'd)

- Logical vector assigned by an expression:

```
> x7 <- x < 7 ; x7
```

```
[1] TRUE TRUE TRUE FALSE
```

- *Logical operators* & (and), | (or) and ! (not) can combine several expressions, e.g.

```
> x7x2 <- x7 & x >= 2 ; x7x2
```

```
[1] FALSE TRUE TRUE FALSE
```

- Can be used in numeric calculations, in which the value of TRUE is 1 and that of FALSE is 0, e.g.

```
> x * x7x2
```

```
[1] 0 2 4 0
```

Data frame

Data frame = *Data matrix* in common statistical jargon

- Rows correspond to observational units, and columns (same length) refer to variables
- Column vectors can be numeric, character or logical
- Columns are subobjects of the data frame. Their names are not directly accessible. Therefore
 - use "surname\$firstname", e.g. `mydata$var1`, or
 - `attach(mydata)` then use only "firstname" `var1`

Data frame input from existing files

- `read.table()` for common ASCII files
- `read.spss()`, `read.ssd()`, `read.dta()`, etc.
in package `foreign` for SPSS, SAS and Stata files etc.
- Excel-files:
 - (1) save the Excel-file in the CSV format,
 - (2) in R use function `read.csv2()`.
- Files from an Internet-address: function `url()`.

Also, a spreadsheet available for data input and editing:

```
dafa <- data.frame( ) ; fix(dafa); ... edit(dafa).
```

Writing, saving and running commands

- Open your favourite text editor
(Notepad, ConTEXT, PFE, etc.)
- Write the command lines without prompts '>' and '+'.
Save the file, e.g. `c:/.../mycmds.R`
- Copy the lines to be executed and paste them on the R console window
- Edit the file if necessary, save and continue
- Running the whole command file:

```
> source("c:/.../mycmds.R", echo=TRUE)
```

Functions

Example: Function `conf.limits` to calculate approximate confidence limits from point estimate (`estim`) and standard error (`SE`) defined:

```
> conf.limits <- function(estim, SE, level = 0.95) {  
+   z <- qnorm(1- (1-level)/2 ) # setting the quantile  
+   lower <- estim - z*SE ; upper <- estim + z*SE  
+   conf.limits <- c(lower, upper)  
+   conf.limits }
```

- *Formal arguments*, here `estim`, `SE`, `level`

Functions (cont'd)

- *Actual arguments*, are used when calling the function

```
> conf.limits(3,1,0.9)
```

```
[1] 1.355146 4.644854
```

NB! *Positional matching*: order of actual arguments.

- *Keyword matching*: the order of arguments in the call is irrelevant if the names of formal arguments are given

```
> conf.limits(SE=1.0, level=0.90, estim=3)
```

- If a *default value* for an argument is given in the definition and is OK, it can be omitted in calling

```
> conf.limits(3,1) # 95% conf. limits
```

```
[1] 1.040036 4.959964
```

Functions (cont'd)

- Defining code can be viewed by typing the function name.
- Can be saved into a separate source file, e.g. `myfuncs.R`, which may contain several functions.
- Accessible in an R run after

```
> source("C:/.../myfuncs.R")
```
- Alternatively from menu bar:
File – Source R code ... etc.
- Loading from Internet:

```
> source(url("http://.../myfuncs.R"))
```

Package

- Collection of functions pertaining to some specialized application area, e.g. `survival`, `mva`, `foreign`
- Available after loading command:
> `library(survival)`
- Alternatively load from the menu bar: *Packages ...*
- New version can be updated via Internet