Introduction to R

Guodong Li

Department of Statistics & Actuarial Science, HKU

Introduction to R

- S: an interactive environment for data analysis developed at Bell Laboratories since 1976
  - 1988 - S2: RA Becker, JM Chambers, A Wilks
  - 1992 - S3: JM Chambers, TJ Hastie
  - 1998 - S4: JM Chambers
- Exclusively licensed by AT&T/Lucent to Insightful Corporation, Seattle WA. Product name: "S-plus".
- R: initially written by Ross Ihaka and Robert Gentleman at Dep. of Statistics of U of Auckland, New Zealand during 1990s.
  - " We have named our language R –in part to acknowledge the influence of S and in part to celebrate our own efforts."
- Since 1997: international R-core team ~15 people & 1000s of code writers and statisticians happy to share their libraries!

Introduction to R

- Most people come to R because of
  - Its statistical analysis, and
  - Drawing a beautiful plot.
- Implementation languages: C, Fortran.
- R is an interpreted computer language.
  - Most user-visible functions are written in R itself, calling upon a smaller set of internal primitives.
  - It is possible to interface procedures written in C, C+, or FORTRAN languages for efficiency, and to write additional primitives.
  - System commands can be called from within R
- R is free, and anyone can contribute to it as packages.
- In R, a statistical analysis is normally done as a series of steps, with intermediate results being stored in objects.

Getting started

- Open the webpage: http://web.hku.hk/~gdli/Tutorial_R.html.
- Where to get R?
  - Go to http://www.r-project.org
  - Downloads: CRAN
  - Set your Mirror
  - Select operation system, e.g. Windows
  - Select base.
  - Download R.
- The textbook
  - An Introduction to R (English version,繁體版,简体版)

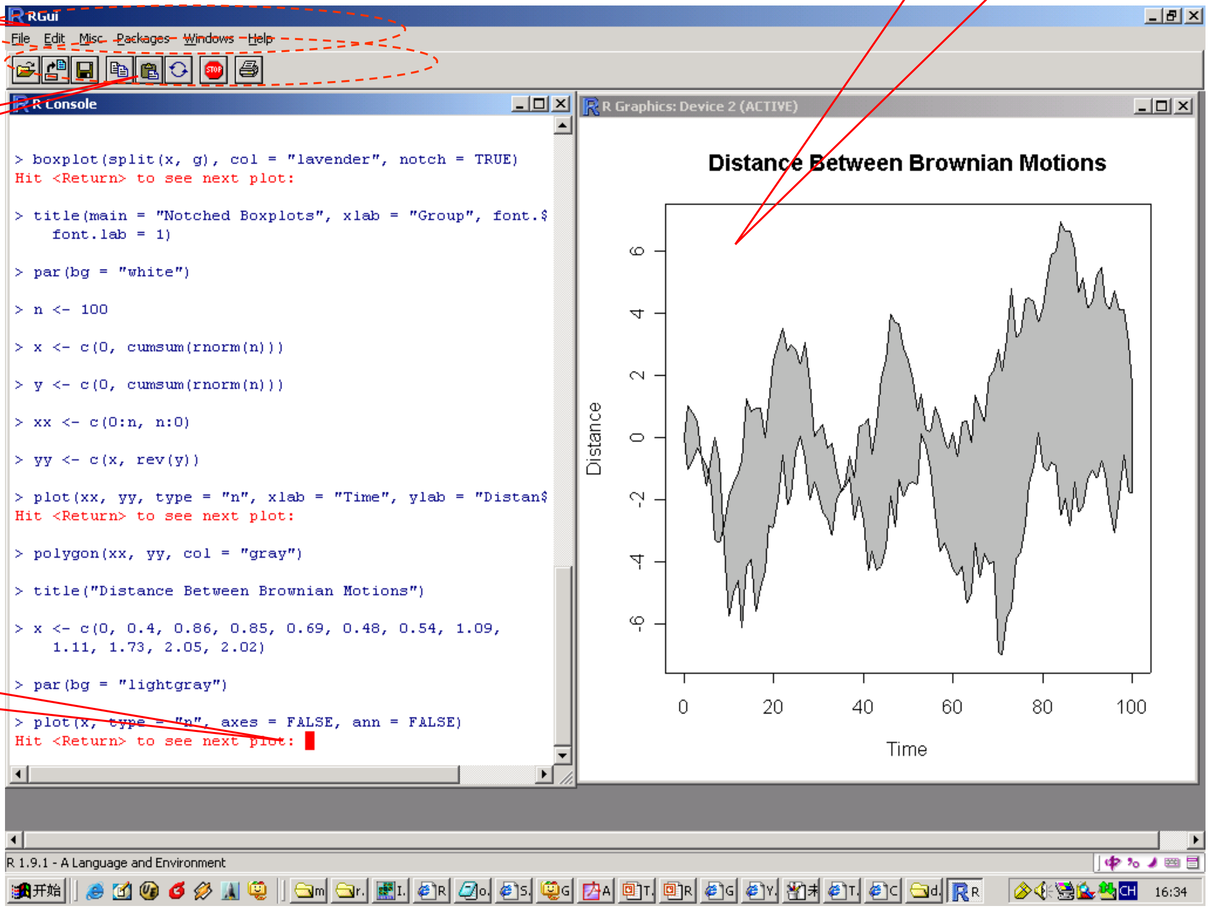
R Console/RGui in Windows(MS)

Menu

Icons

Command box

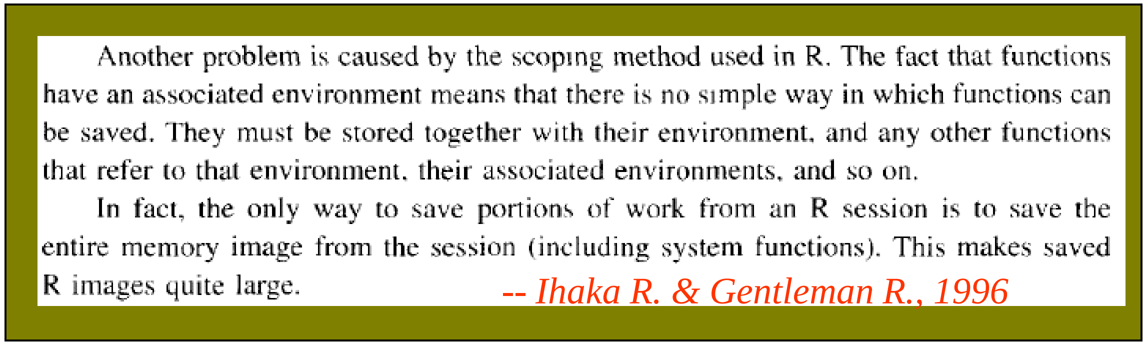
Graphics box


RGui
File Edit Misc Packages Windows Help
R Console
R Graphics: Device 2 (ACTIVE)
Distance Between Brownian Motions
Distance
Time
R 1.9.1 - A Language and Environment
开始
16:34

```
> boxplot(split(x, g), col = "lavender", notch = TRUE)
Hit <Return> to see next plot:

> title(main = "Notched Boxplots", xlab = "Group", font.$
    font.lab = 1)

> par(bg = "white")

> n <- 100

> x <- c(0, cumsum(rnorm(n)))

> y <- c(0, cumsum(rnorm(n)))

> xx <- c(0:n, n:0)

> yy <- c(x, rev(y))

> plot(xx, yy, type = "n", xlab = "Time", ylab = "Distan$
Hit <Return> to see next plot:

> polygon(xx, yy, col = "gray")

> title("Distance Between Brownian Motions")

> x <- c(0, 0.4, 0.86, 0.85, 0.69, 0.48, 0.54, 1.09,
    1.11, 1.73, 2.05, 2.02)

> par(bg = "lightgray")

> plot(x, type = "n", axes = FALSE, ann = FALSE)
Hit <Return> to see next plot:
```

Several concepts in Administrating R

- Workspace
  - xxx.RData
- History
  - xxx.Rhistory
- Package
- Object
- Session
- Console



Another problem is caused by the scoping method used in R. The fact that functions have an associated environment means that there is no simple way in which functions can be saved. They must be stored together with their environment, and any other functions that refer to that environment, their associated environments, and so on.

In fact, the only way to save portions of work from an R session is to save the entire memory image from the session (including system functions). This makes saved R images quite large.

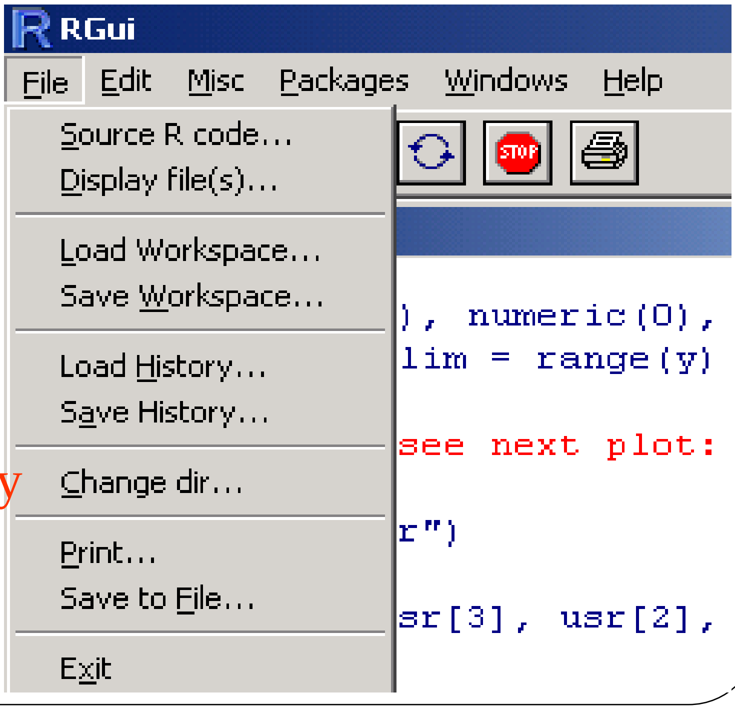
-- Ihaka R. & Gentleman R., 1996

Run your R codes

Load/save workspace

Load/save History

Change your working directory



```
), numeric(0),
lim = range(y)

see next plot:

r")

sr[3], usr[2],
```

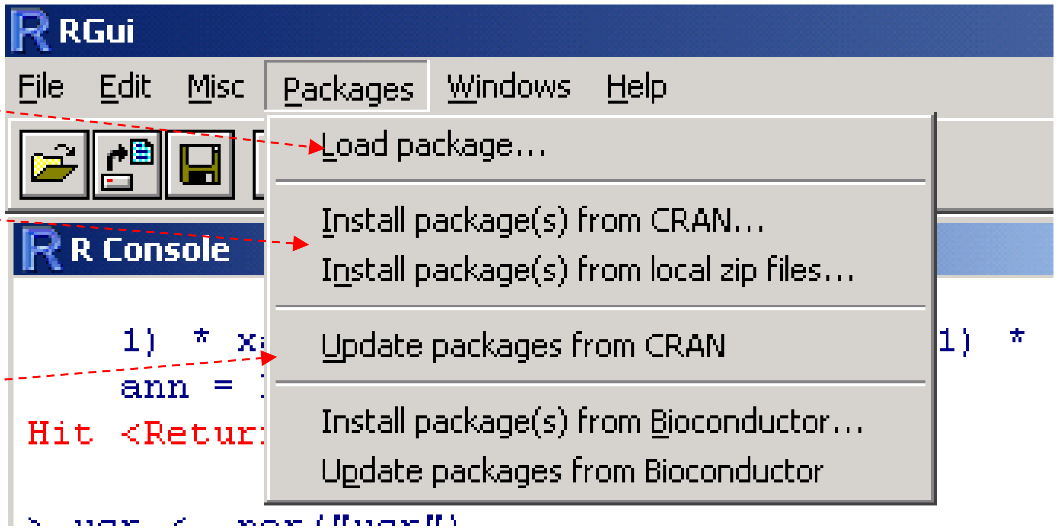
Add a new package

- Commands:
  - *library()* add a package in the library
  - *detach(package : xxx)* detach a package
- All can do in the GUI (*except detach()*)

Load a local package

Install packages from internet or local

Update the local package from internet


RGui
File
Edit
Misc
Packages
Windows
Help
Load package...
Install package(s) from CRAN...
Install package(s) from local zip files...
Update packages from CRAN
Install package(s) from Bioconductor...
Update packages from Bioconductor
R Console

```
    1)  * x
    ann = 
Hit <Retur
```

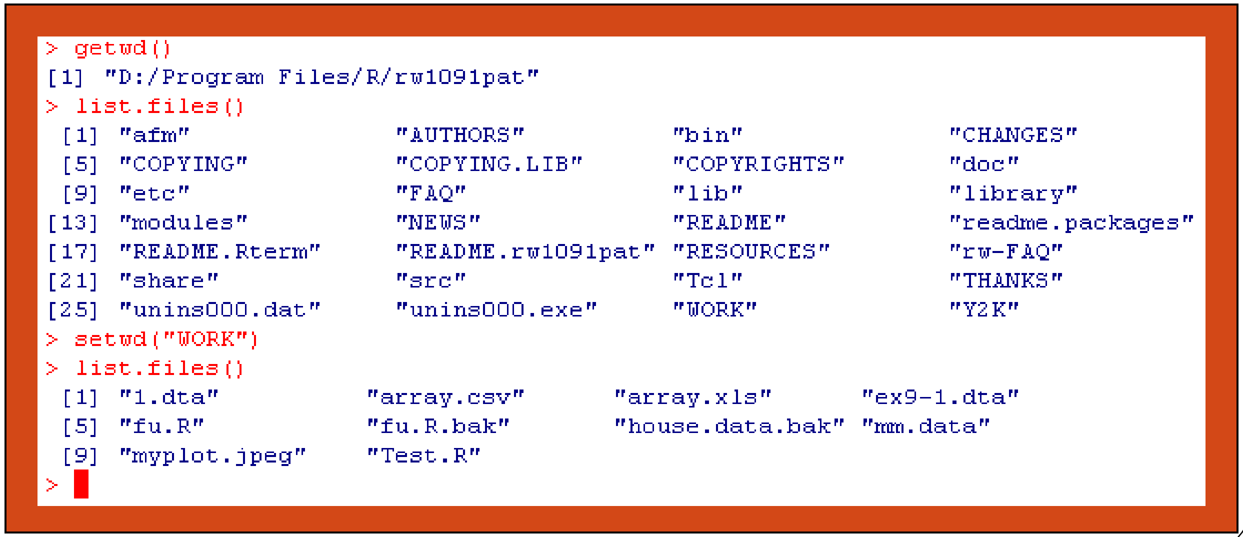
Packages in R Environment

- *Basic* packages
  - "package:methods" "package:stats" "package:graphics“ "package:utils"   "package:base"
- *Recommanded* packages
  - grid; lattice;e1071…
- *Contributed* packages (more than 366 packages nowadays)
  - ……

You can see what packages loaded now by the command *search().*

Don’t lose your way!

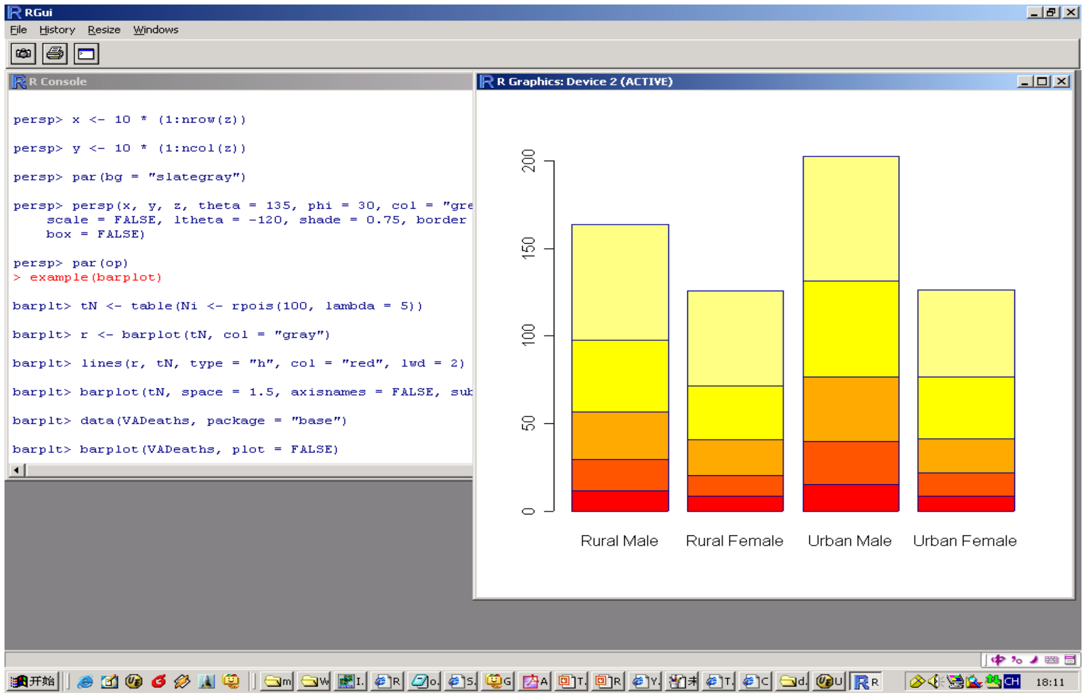
- Three useful system command
  - *getwd()* Get Working Directory
  - *setwd()* Set Working Directory
  - *list.files()* List the Files in a Directory/Folder



```
> getwd()
[1] "D:/Program Files/R/rw1091pat"
> list.files()
 [1] "afm"             "AUTHORS"           "bin"          "CHANGES"
 [5] "COPYING"         "COPYING.LIB"       "COPYRIGHTS"   "doc"
 [9] "etc"             "FAQ"               "lib"          "library"
[13] "modules"         "NEWS"              "README"       "readme.packages"
[17] "README.Rterm"    "README.rw1091pat"  "RESOURCES"    "rw-FAQ"
[21] "share"           "src"               "Tcl"          "THANKS"
[25] "unins000.dat"    "unins000.exe"      "WORK"         "Y2K"
> setwd("WORK")
> list.files()
 [1] "1.dta"         "array.csv"     "array.xls"       "ex9-1.dta"
 [5] "fu.R"          "fu.R.bak"      "house.data.bak"  "mm.data"
 [9] "myplot.jpeg"   "Test.R"
> 
```

Show the Demonstrations of the Packages/Functions

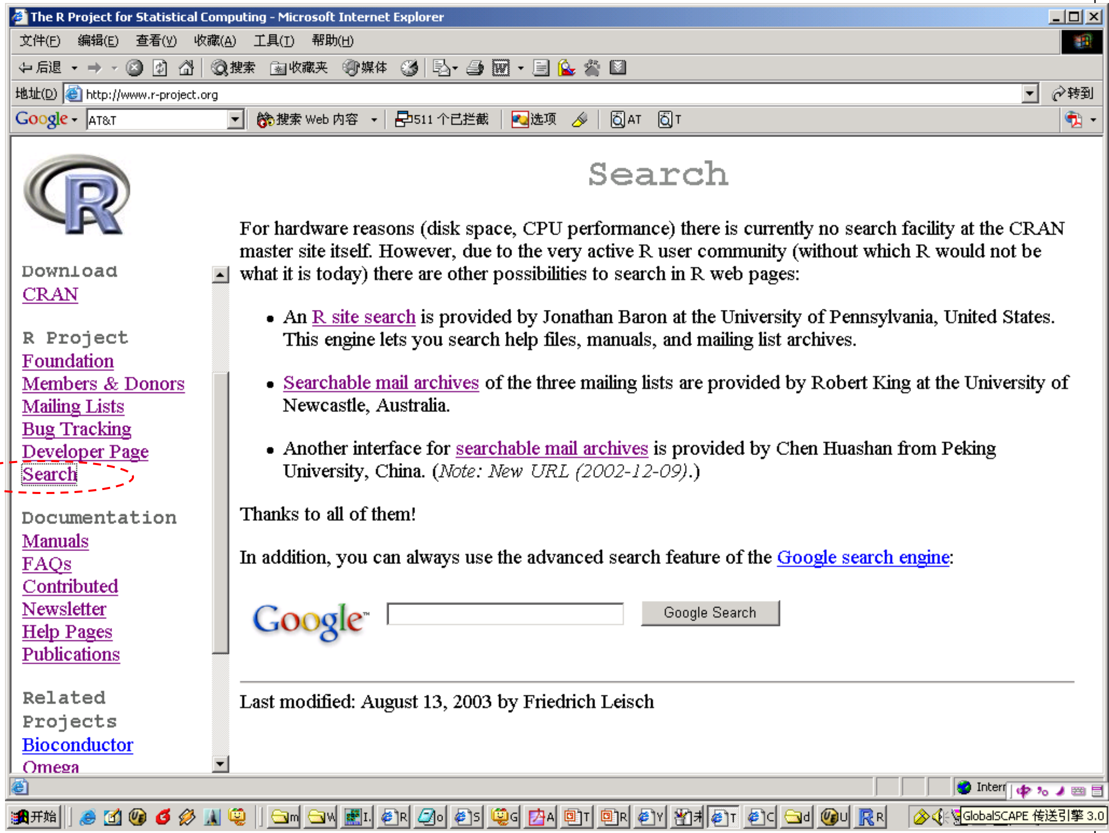
- Commands
  - *demo()* Demonstrations of R Functionality
  - *example()* Run an Examples Section from the Online Help



```
persp> x <- 10 * (1:nrow(z))

persp> y <- 10 * (1:ncol(z))

persp> par(bg = "slategray")

persp> persp(x, y, z, theta = 135, phi = 30, col = "gre
    scale = FALSE, ltheta = -120, shade = 0.75, border
    box = FALSE)

persp> par(op)
> example(barplot)

barplt> tN <- table(Ni <- rpois(100, lambda = 5))

barplt> r <- barplot(tN, col = "gray")

barplt> lines(r, tN, type = "h", col = "red", lwd = 2)

barplt> barplot(tN, space = 1.5, axisnames = FALSE, sub

barplt> data(VADeaths, package = "base")

barplt> barplot(VADeaths, plot = FALSE)
```

Getting Helps

- Several commands
  - *help.start()*
  - *help() or ?()*
  - *help.search()*
- Internet searching
  - I like it very much. It seems omnipotence.



Search

For hardware reasons (disk space, CPU performance) there is currently no search facility at the CRAN master site itself. However, due to the very active R user community (without which R would not be what it is today) there are other possibilities to search in R web pages:

- An R site search is provided by Jonathan Baron at the University of Pennsylvania, United States. This engine lets you search help files, manuals, and mailing list archives.
- Searchable mail archives of the three mailing lists are provided by Robert King at the University of Newcastle, Australia.
- Another interface for searchable mail archives is provided by Chen Huashan from Peking University, China. (*Note: New URL (2002-12-09).*)

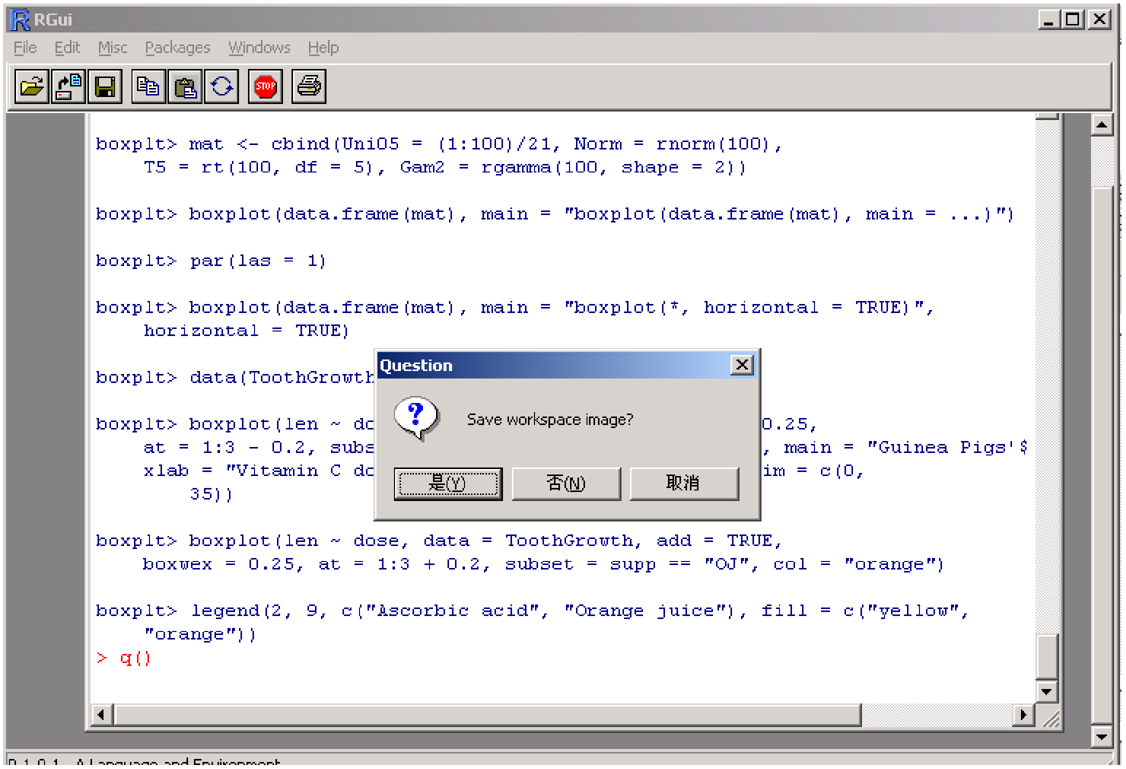
Thanks to all of them!

In addition, you can always use the advanced search feature of the Google search engine:

Last modified: August 13, 2003 by Friedrich Leisch

Quit R

- Command
  - *q() or quit()* Terminate an R Session

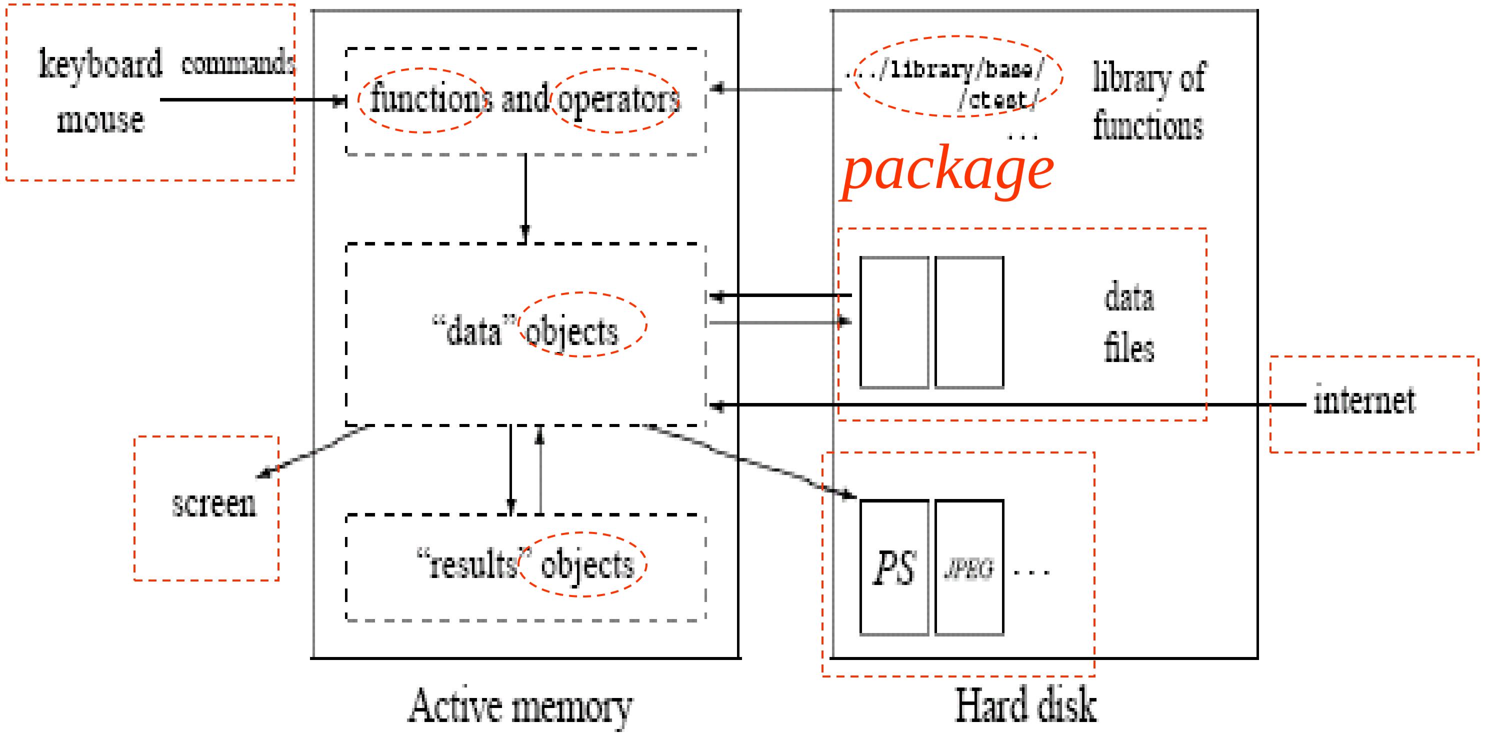


```
boxplt> mat <- cbind(Uni05 = (1:100)/21, Norm = rnorm(100),
    T5 = rt(100, df = 5), Gam2 = rgamma(100, shape = 2))

boxplt> boxplot(data.frame(mat), main = "boxplot(data.frame(mat), main = ...)")

boxplt> par(las = 1)

boxplt> boxplot(data.frame(mat), main = "boxplot(*, horizontal = TRUE)",
    horizontal = TRUE)

boxplt> data(ToothGrowth

boxplt> boxplot(len ~ do                                   0.25,
    at = 1:3 - 0.2, subs                                  , main = "Guinea Pigs'$
    xlab = "Vitamin C do                                  im = c(0,
        35))

boxplt> boxplot(len ~ dose, data = ToothGrowth, add = TRUE,
    boxwex = 0.25, at = 1:3 + 0.2, subset = supp == "OJ", col = "orange")

boxplt> legend(2, 9, c("Ascorbic acid", "Orange juice"), fill = c("yellow",
    "orange"))
> q()
```

Question

Save workspace image?

Basic R working flow(Object orientation)



-- *R for Beginners.* Emmanuel Paradis

How to use R?
Part I

Chapters 1-6 in An Introduction to R

R commands

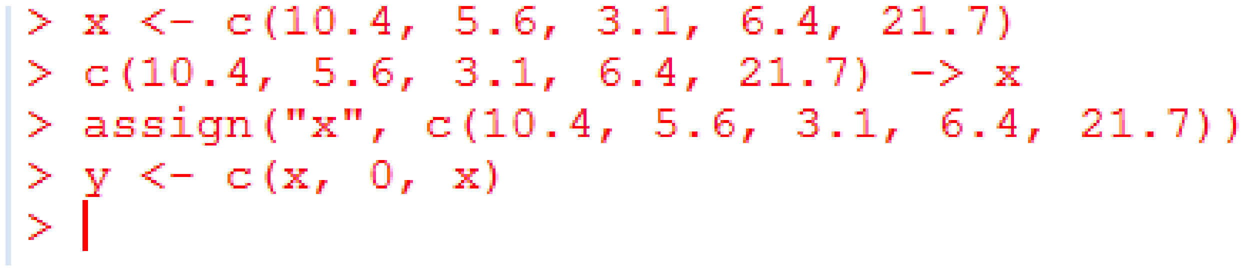
- It is case sensitive.
- The R name
  - consists of letter, number, "." and "_" (*a1*._);
  - must start with letter or ".";
  - "*.2XX*" is illegal.
- Elementary commands consist of either expressions or assignments.
- Commands are separated either by a semi-colon (';'), or by a newline.
- Elementary commands can be grouped by braces ('{' and '}').
- Comments start with a hashmark ('#').
- If a plus mark ('+') appears, then the command is not complete.
- The vertical arrow keys on the keyboard can be used to scroll forward and backward through a command history.

R commands

- 1.10 Executing commands from or diverting output to a file
  - *source("commands.R")* Executing commands in *commands.R*
  - *sink("record.lis")* Output all subsequent to file, *record.lis*
  - *sink()* Restores it to the console.
- 1.11 Checking and removing objects
  - *Objects() or ls()* Display current objects;
  - *rm(x,y)* Romove objects *x* and *y*.

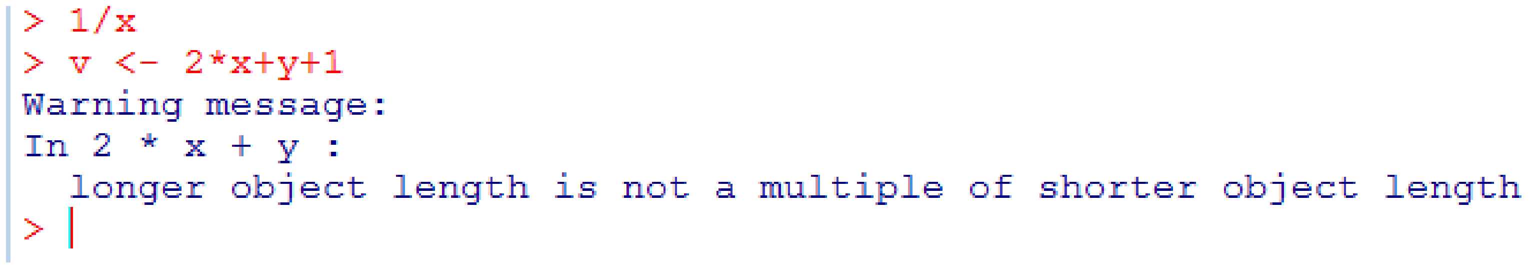
Chapter 2 Simple manipulations; numbers and vectors

- 2.1 Vectors and assignment



```
> x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
> c(10.4, 5.6, 3.1, 6.4, 21.7) -> x
> assign("x", c(10.4, 5.6, 3.1, 6.4, 21.7))
> y <- c(x, 0, x)
> |
```

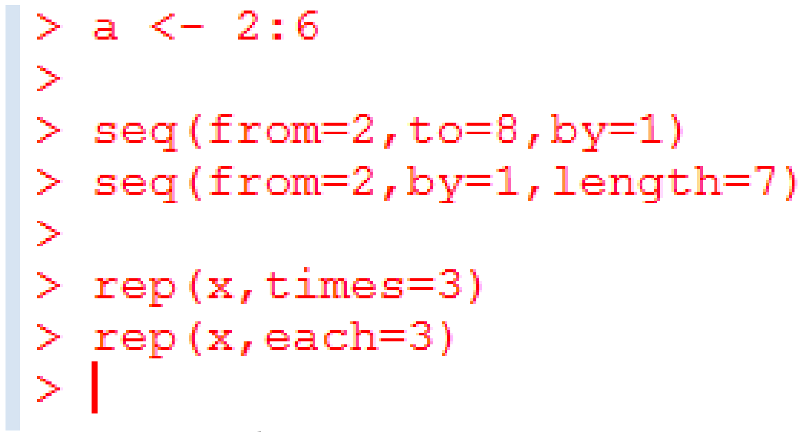
- 2.2 Vector arithmetic
  - Elementary operators: +, -, *, / and ^;
  - Functions: log, exp, sin, cos, tan, sqrt, max, min, range, length, sum, prod, sort, order,etc.;
  - *sqrt(-17)* and *sqrt(-17+0i)*



```
> 1/x
> v <- 2*x+y+1
Warning message:
In 2 * x + y :
  longer object length is not a multiple of shorter object length
> |
```

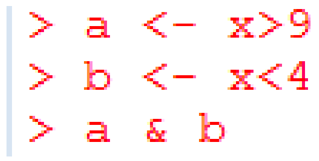
Chapter 2 Simple manipulations; numbers and vectors

- 2.3 Generating regular sequences



```
> a <- 2:6
>
> seq(from=2,to=8,by=1)
> seq(from=2,by=1,length=7)
>
> rep(x,times=3)
> rep(x,each=3)
> |
```

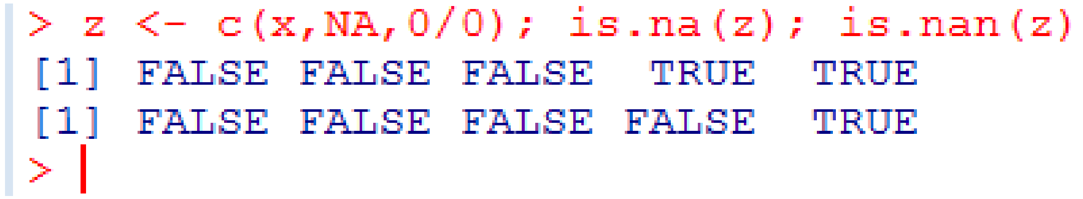
- 2.4 Logical vectors
  - Values: TRUE, FALSE, and NA;
  - Logical operators: <, <=, >, >=, ==, !=;
  - Logical expressions: c1 & c2 (“and”), c1 | c2 (“or”), !c1 (“not”);
  - Sometimes, FALSE becoming 0 and TRUE becoming 1.



```
> a <- x>9
> b <- x<4
> a & b
```

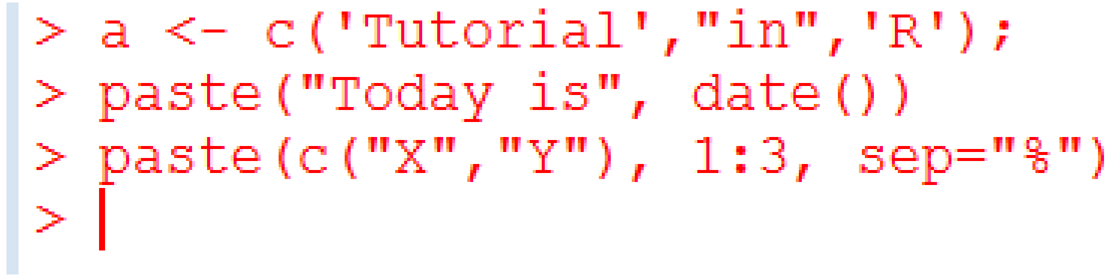
Chapter 2 Simple manipulations; numbers and vectors

- 2.5 Missing values



```
> z <- c(x,NA,0/0); is.na(z); is.nan(z)
[1] FALSE FALSE FALSE  TRUE  TRUE
[1] FALSE FALSE FALSE FALSE  TRUE
> 
```

- 2.6 Character vectors

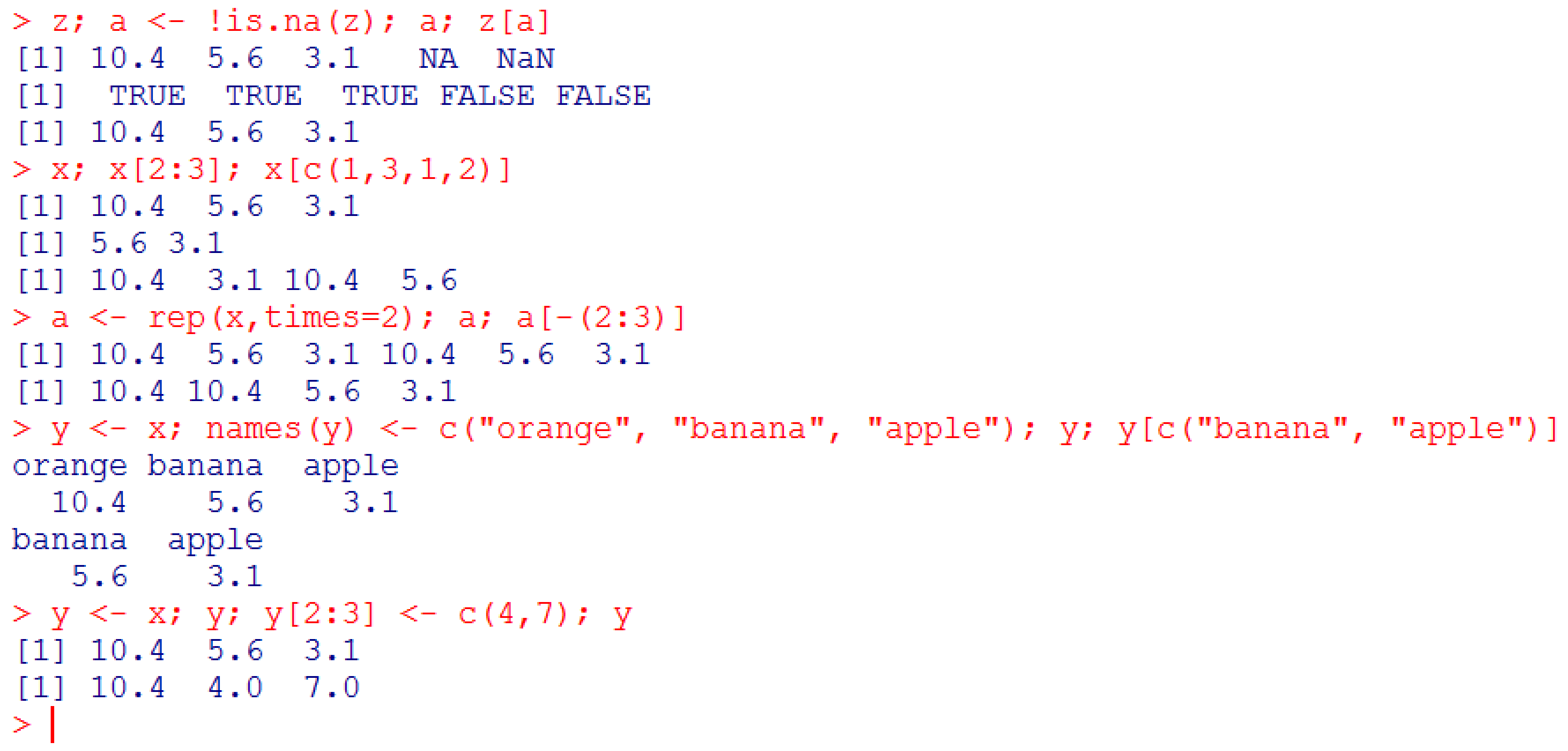


```
> a <- c('Tutorial',"in",'R');
> paste("Today is", date())
> paste(c("X","Y"), 1:3, sep="%")
> 
```

- 2.7 Index vectors; selecting and modifying subsets of a data set
  1. A logical vector.
  2. A vector of positive integral quantities.
  3. A vector of negative integral quantities.
  4. A vector of character strings.

Chapter 2 Simple manipulations; numbers and vectors

- 2.7 Index vectors; selecting and modifying subsets of a data set (cont.')



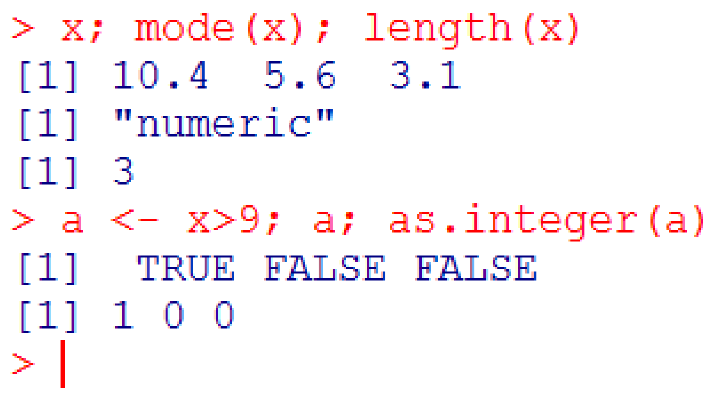
```
> z; a <- !is.na(z); a; z[a]
[1] 10.4  5.6  3.1   NA  NaN
[1]  TRUE  TRUE  TRUE FALSE FALSE
[1] 10.4  5.6  3.1
> x; x[2:3]; x[c(1,3,1,2)]
[1] 10.4  5.6  3.1
[1] 5.6 3.1
[1] 10.4  3.1 10.4  5.6
> a <- rep(x,times=2); a; a[-(2:3)]
[1] 10.4  5.6  3.1 10.4  5.6  3.1
[1] 10.4 10.4  5.6  3.1
> y <- x; names(y) <- c("orange", "banana", "apple"); y; y[c("banana", "apple")]
orange banana  apple
  10.4    5.6    3.1
banana  apple
   5.6    3.1
> y <- x; y; y[2:3] <- c(4,7); y
[1] 10.4  5.6  3.1
[1] 10.4  4.0  7.0
> |
```

Chapter 2 Simple manipulations; numbers and vectors

- 2.8 Other types of objects
  - matrices or more generally arrays are multi-dimensional generalizations of vectors. In fact, they are vectors that can be indexed by two or more indices and will be printed in special ways.
  - factors provide compact ways to handle categorical data.
  - lists are a general form of vector in which the various elements need not be of the same type, and are often themselves vectors or lists. Lists provide a convenient way to return the results of a statistical computation.
  - data frames are matrix-like structures, in which the columns can be of different types. Think of data frames as ‘data matrices’ with one row per observational unit but with (possibly) both numerical and categorical variables. Many experiments are best described by data frames: the treatments are categorical but the response is numeric.
  - functions are themselves objects in R which can be stored in the project’s workspace. This provides a simple and convenient way to extend R.

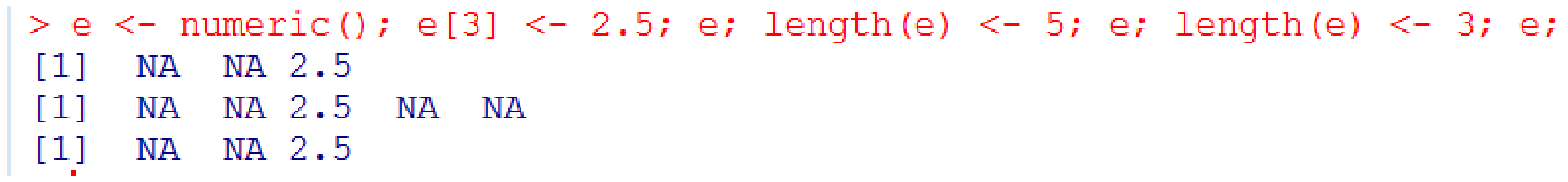
Chapter 3 Objects, their modes and attributes

- 3.1 Intrinsic attributes: mode and length
  - *vector* has a "atomic" structure, i.e. their components are all of the same type, or mode (numeric, complex, logical, character and raw).
  - *list* has a "recursive" structure.



```
> x; mode(x); length(x)
[1] 10.4  5.6  3.1
[1] "numeric"
[1] 3
> a <- x>9; a; as.integer(a)
[1]  TRUE FALSE FALSE
[1] 1 0 0
> |
```

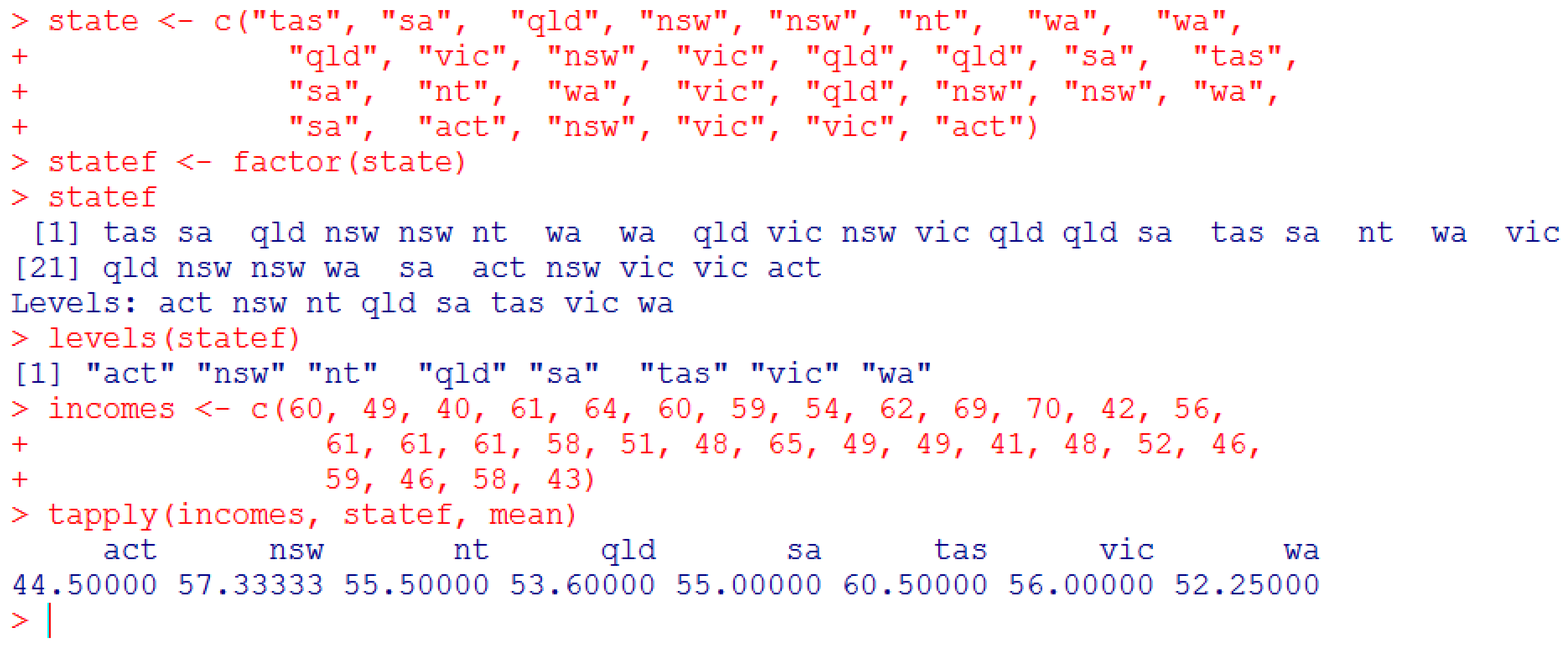
- 3.2 Changing the length of an object



```
> e <- numeric(); e[3] <- 2.5; e; length(e) <- 5; e; length(e) <- 3; e;
[1]  NA  NA 2.5
[1]  NA  NA 2.5  NA  NA
[1]  NA  NA 2.5
```

Chapter 4 Ordered and unordered factors

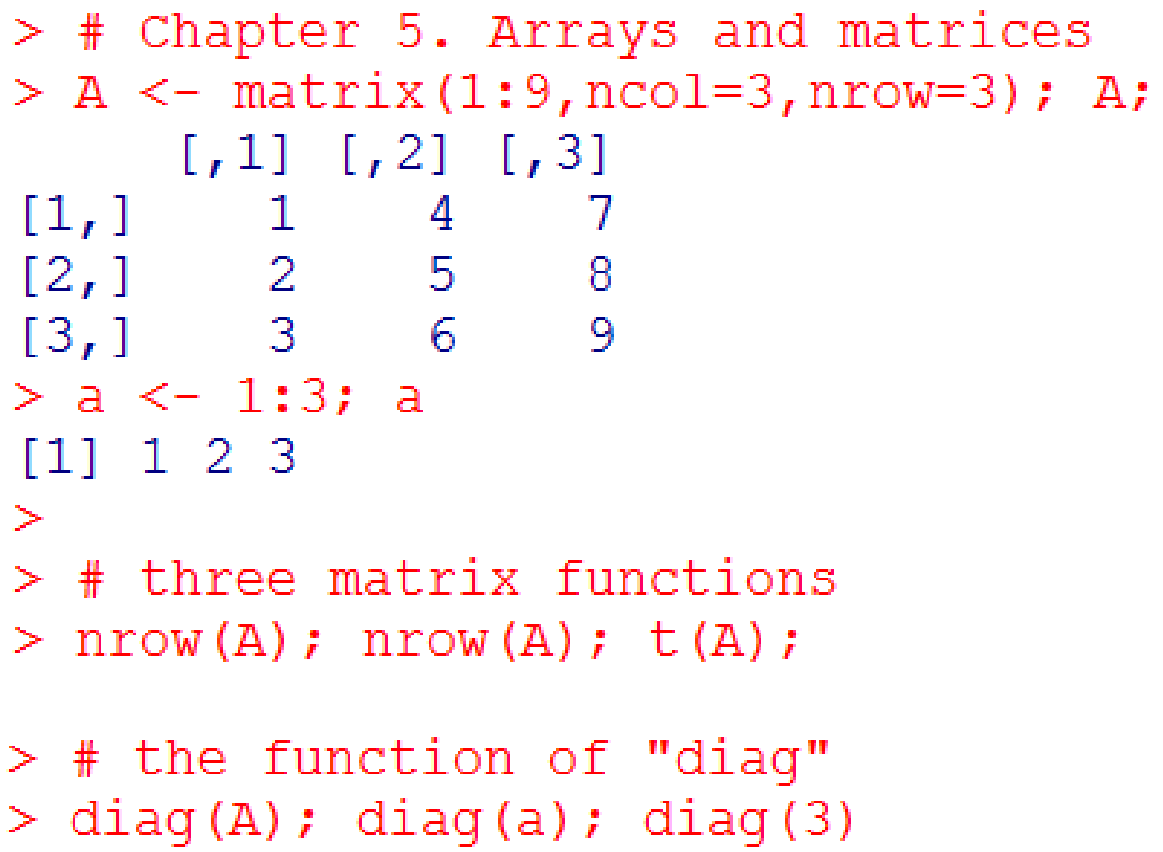
- Ordered and unordered factors.
- Functions: factor, levels, ordered, tapply.



```
> state <- c("tas", "sa",  "qld", "nsw", "nsw", "nt",  "wa",  "wa",
+              "qld", "vic", "nsw", "vic", "qld", "qld", "sa",  "tas",
+              "sa",  "nt",  "wa",  "vic", "qld", "nsw", "nsw", "wa",
+              "sa",  "act", "nsw", "vic", "vic", "act")
> statef <- factor(state)
> statef
 [1] tas sa  qld nsw nsw nt  wa  wa  qld vic nsw vic qld qld sa  tas sa  nt  wa  vic
[21] qld nsw nsw wa  sa  act nsw vic vic act
Levels: act nsw nt qld sa tas vic wa
> levels(statef)
[1] "act" "nsw" "nt"  "qld" "sa"  "tas" "vic" "wa"
> incomes <- c(60, 49, 40, 61, 64, 60, 59, 54, 62, 69, 70, 42, 56,
+                61, 61, 61, 58, 51, 48, 65, 49, 49, 41, 48, 52, 46,
+                59, 46, 58, 43)
> tapply(incomes, statef, mean)
     act      nsw       nt      qld       sa      tas      vic       wa
44.50000 57.33333 55.50000 53.60000 55.00000 60.50000 56.00000 52.25000
> |
```

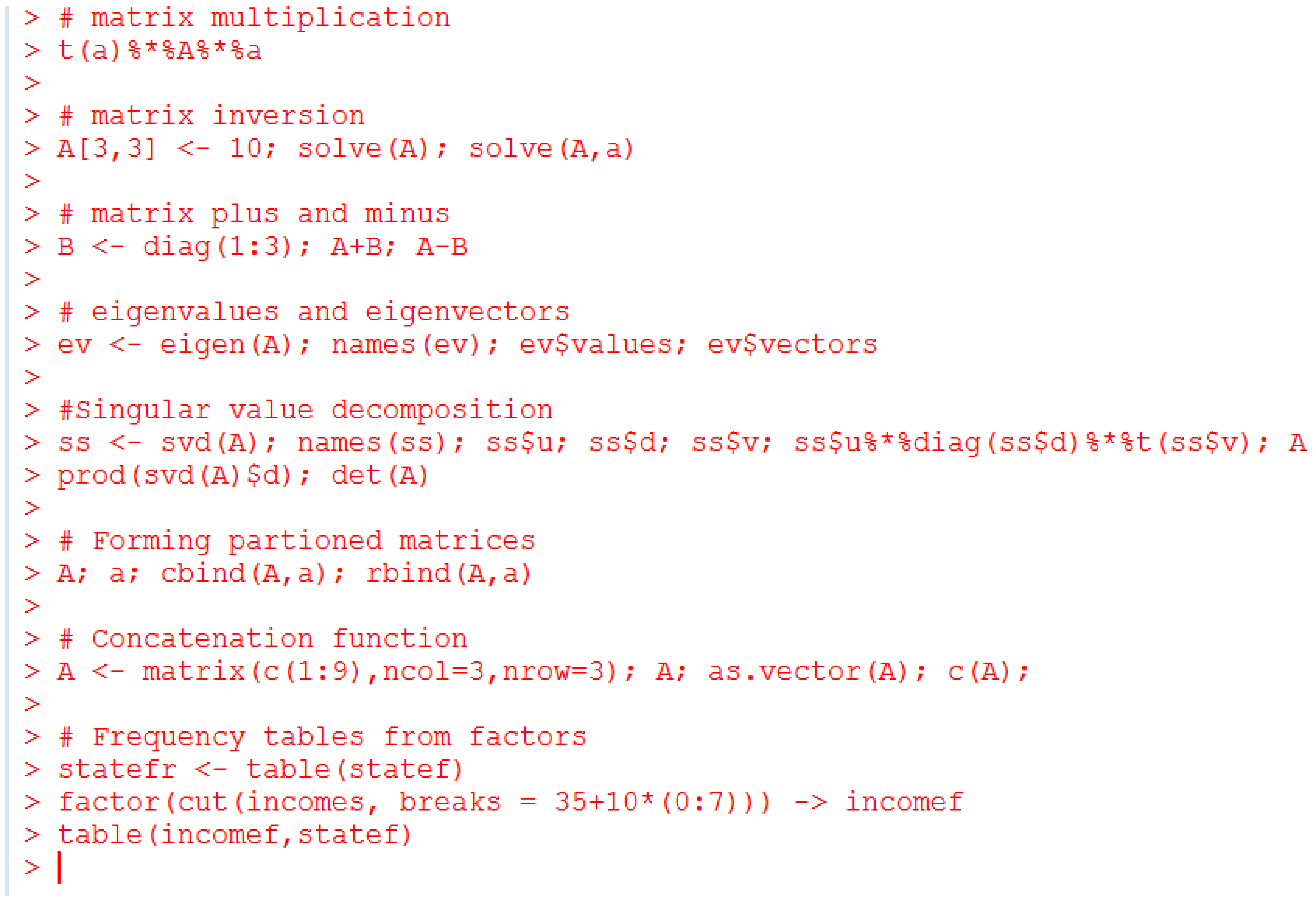
Chapter 5 Arrays and matrices

- An array can be considered as a multiply subscripted collection of data entries, for example numeric.
- A matrix is a special array with two dimensions.



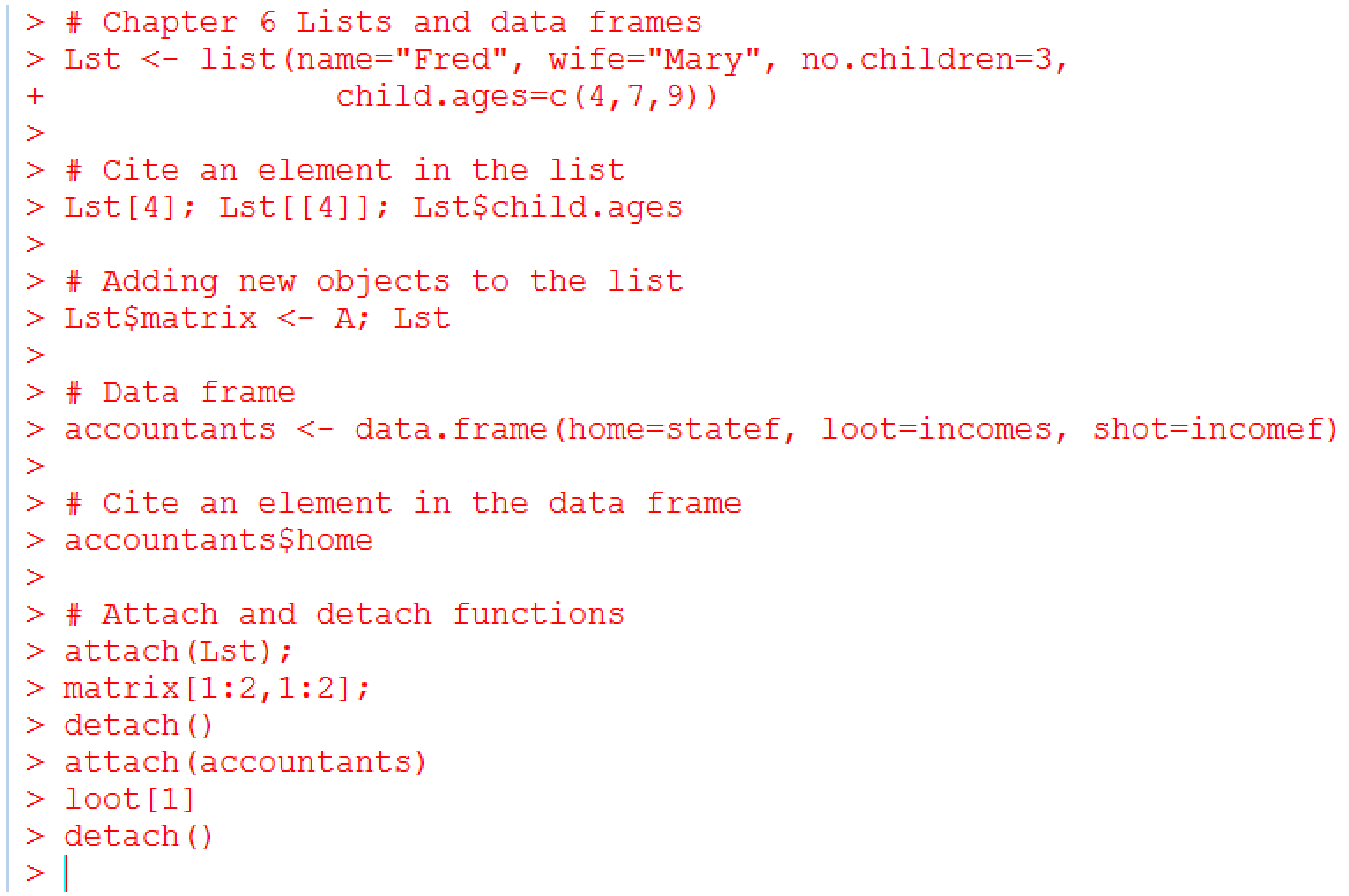
```
> # Chapter 5. Arrays and matrices
> A <- matrix(1:9,ncol=3,nrow=3); A;
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> a <- 1:3; a
[1] 1 2 3
>
> # three matrix functions
> nrow(A); nrow(A); t(A);

> # the function of "diag"
> diag(A); diag(a); diag(3)
```

Chapter 5 Arrays and matrices



```
> # matrix multiplication
> t(a)%*%A%*%a
>
> # matrix inversion
> A[3,3] <- 10; solve(A); solve(A,a)
>
> # matrix plus and minus
> B <- diag(1:3); A+B; A-B
>
> # eigenvalues and eigenvectors
> ev <- eigen(A); names(ev); ev$values; ev$vectors
>
> #Singular value decomposition
> ss <- svd(A); names(ss); ss$u; ss$d; ss$v; ss$u%*%diag(ss$d)%*%t(ss$v); A
> prod(svd(A)$d); det(A)
>
> # Forming partioned matrices
> A; a; cbind(A,a); rbind(A,a)
>
> # Concatenation function
> A <- matrix(c(1:9),ncol=3,nrow=3); A; as.vector(A); c(A);
>
> # Frequency tables from factors
> statefr <- table(statef)
> factor(cut(incomes, breaks = 35+10*(0:7))) -> incomef
> table(incomef,statef)
> |
```

Chapter 6 Lists and data frames



```
> # Chapter 6 Lists and data frames
> Lst <- list(name="Fred", wife="Mary", no.children=3,
+              child.ages=c(4,7,9))
>
> # Cite an element in the list
> Lst[4]; Lst[[4]]; Lst$child.ages
>
> # Adding new objects to the list
> Lst$matrix <- A; Lst
>
> # Data frame
> accountants <- data.frame(home=statef, loot=incomes, shot=incomef)
>
> # Cite an element in the data frame
> accountants$home
>
> # Attach and detach functions
> attach(Lst);
> matrix[1:2,1:2];
> detach()
> attach(accountants)
> loot[1]
> detach()
> |
```

How to use R?
Part II

Chapters 7-12 in An Introduction to R

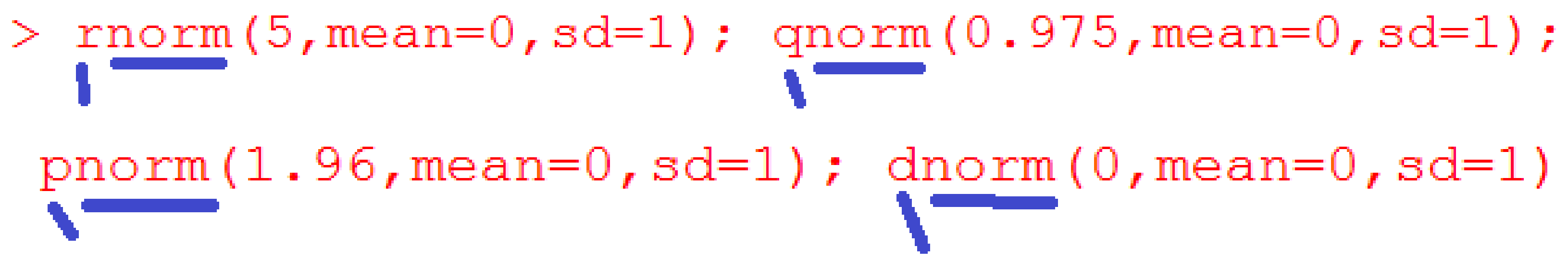
Chapter 7. Reading data from files

- *read.table()*
    - copy two data in section 7.1 into ‘file1.txt’ and ‘file2.txt’.
    - save them into ‘D:/temp’.
    - creat ‘file3.txt’ without header line and the last column.
- *scan* function
- Accessing builtin datasets
- Loading data from other R packages
- Editing data

```
> # Chapter 7. Reading data from files
> getwd()
> setwd('D:/temp')
> list.files()
>
> # read.table function
> read.table('file1.txt') -> house
> mode(house); names(house)
>
> read.table('file2.txt')
> read.table('file2.txt',header=TRUE) -> house
> mode(house); names(house)
>
> # scan function
> scan('file1.txt'); scan('file2.txt');
> scan('file3.txt'); scan('file3.txt',list(0,0,0,0,""));
> # Accessing builtin datasets
> data(); data(co2); co2
>
> # Loading data from other R packages
> data(package="rpart")
> data(solder,package="rpart")
>
> # Editing data
> xnew <- edit(house)
> xnew <- edit(data.frame())
> |
```

Chapter 8. Probability distributions

- 8.1 Probability distributions
  - Prefix the name
    - 'd' for the density,
    - 'p' for the CDF,
    - 'q' for the quantile function and
    - 'r' for simulation (random deviates).
  - The first argument is x for dxxx, q for pxxx, p for qxxx and n for rxxx (except for rhyper, rsignrank and rwilcox, for which it is nn).
  - Please refer to section 8.1 for other distributions.



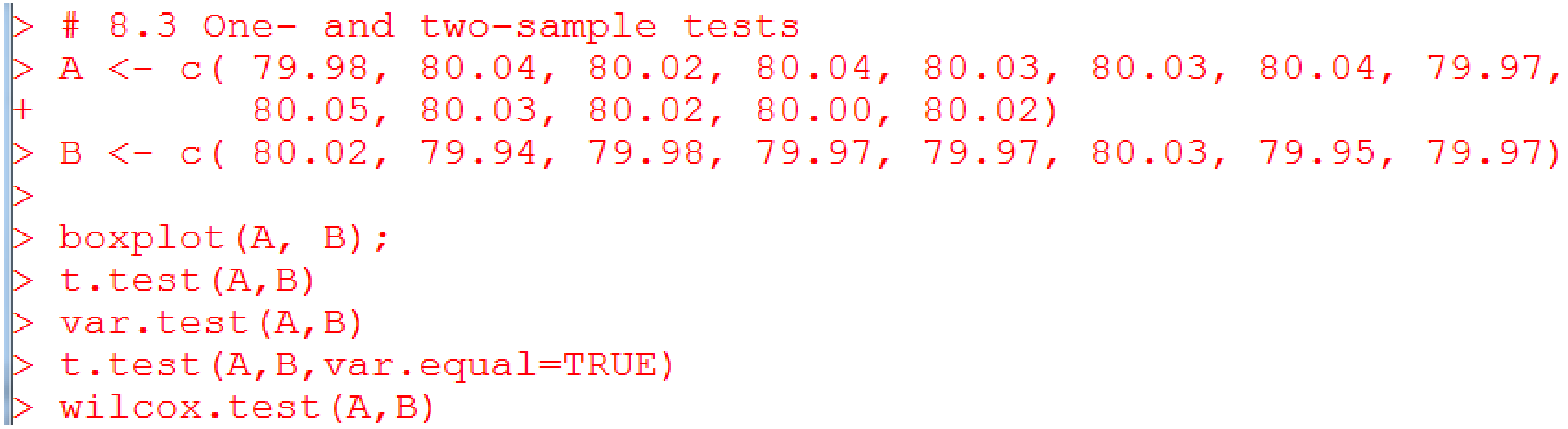
```
> rnorm(5,mean=0,sd=1); qnorm(0.975,mean=0,sd=1);
 pnorm(1.96,mean=0,sd=1); dnorm(0,mean=0,sd=1)
```

Chapter 8. Probability distributions

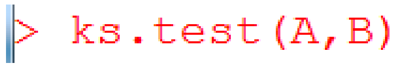
- 8.2 Examining the distribution of a set of data
  - Functions: *summary*, *fivenum*, *stem*, *hist*, *ecdf*, *qqnorm*, *shapiro.test*, *ks.test*, etc.
  - example: the *eruption* times of dataset *faithful*.
  - Questions: Suppose that we focus on the values greater than 3. Please do the same checking
- 8.3 One- and two-sample tests
  - Functions: *boxplot*, *t.test*, *var.test*, *wilcox.test*, *ks.test*

```
> # 8.1 Probability distributions
> rnorm(5,mean=0,sd=1); qnorm(0.975,mean=0,sd=1);
> pnorm(1.96,mean=0,sd=1); dnorm(0,mean=0,sd=1)
>
> # 8.2 Examining the distribution of a set of data
> attach(faithful); eruptions;
> summary(eruptions);
>
> # Tukey's five number summary
> # (minimum, lower-hinge, median, upper-hinge, maximum)
> fivenum(eruptions)
>
> # Drawing histogram
> hist(eruptions)
> # make the bins smaller, make a plot of density
> hist(eruptions, seq(1.6, 5.2, 0.2), prob=TRUE)
> lines(density(eruptions, bw=0.1))
> rug(eruptions) # show the actual data points
>
> # Drawing empirical cumulative distribution function
> plot(ecdf(eruptions), do.points=FALSE, verticals=TRUE)
>
> # Drawing Q-Q plot
> par(pty="s")        # arrange for a square figure region
> qqnorm(eruptions); qqline(eruptions)
>
> # Shapiro-Wilk test and Kolmogorov-Smirnov test
> shapiro.test(eruptions)
> ks.test(eruptions, "pnorm", mean = mean(eruptions), sd = sqrt(var(eruptions)))
```

Chapter 8. Probability distributions



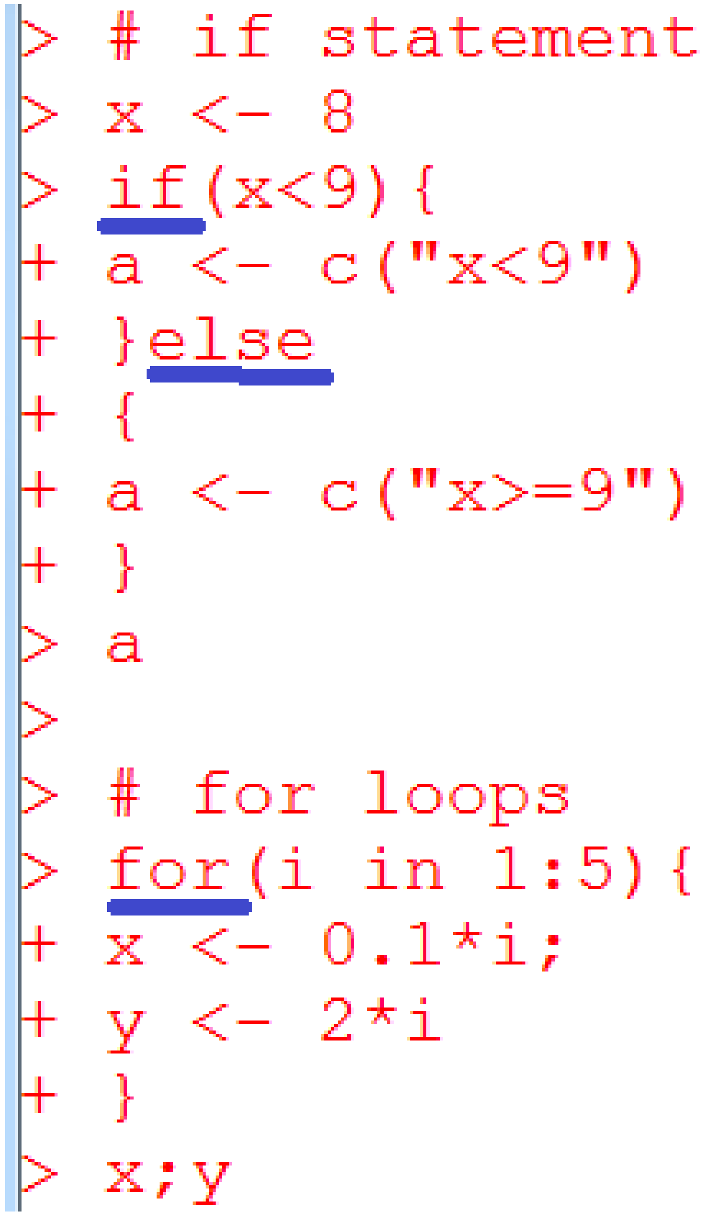
```
> # 8.3 One- and two-sample tests
> A <- c( 79.98, 80.04, 80.02, 80.04, 80.03, 80.03, 80.04, 79.97,
+         80.05, 80.03, 80.02, 80.00, 80.02)
> B <- c( 80.02, 79.94, 79.98, 79.97, 79.97, 80.03, 79.95, 79.97)
>
> boxplot(A, B);
> t.test(A,B)
> var.test(A,B)
> t.test(A,B,var.equal=TRUE)
> wilcox.test(A,B)
```



```
> ks.test(A,B)
```

Chapter 9. Grouping, loops and conditional execution

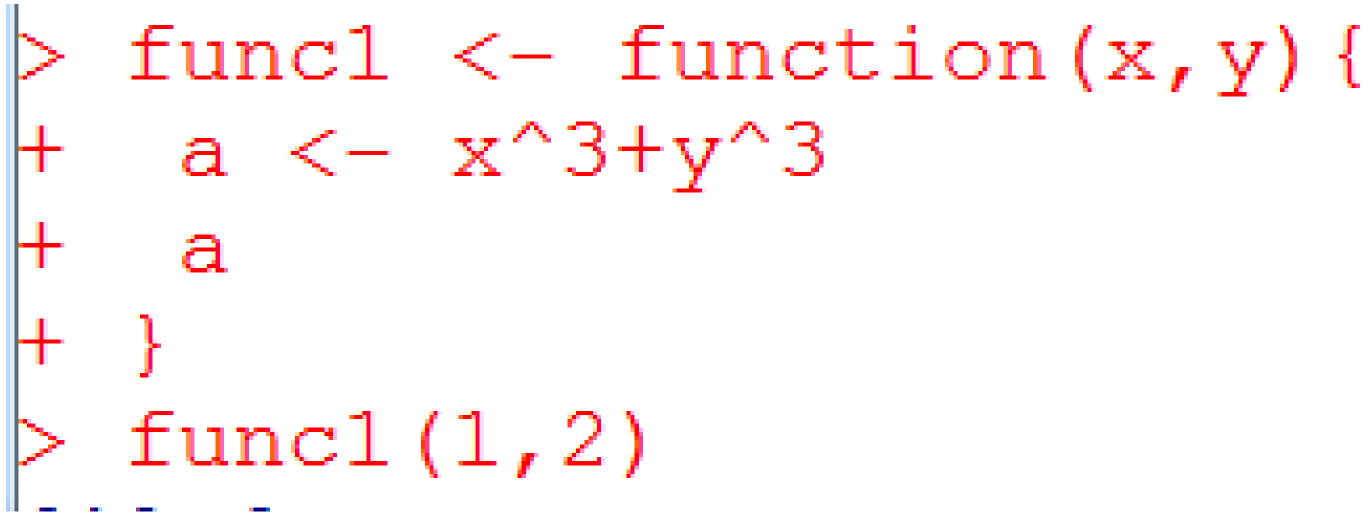
- There are also *repeat* and *while*, however, they are not commonly used.
- Braces ("{}") can be used to group more than one lines of commands.



```
> # if statement
> x <- 8
> if(x<9){
+ a <- c("x<9")
+ }else
+ {
+ a <- c("x>=9")
+ }
> a
>
> # for loops
> for(i in 1:5){
+ x <- 0.1*i;
+ y <- 2*i
+ }
> x;y
```

Chapter 10. Writing your own functions

- A simple example below.



```
> func1 <- function(x,y){
+   a <- x^3+y^3
+   a
+ }
> func1(1,2)
```

Chapter 11. Statistical models in R

- An example of the linear regression model

```
> data();
> anscombe
>
> lm(y1 ~ y2+y3,data=anscombe) -> fm1
> fm1
Call:
lm(formula = y1 ~ y2 + y3, data = anscombe)

Coefficients:
(Intercept)           y2            y3
    1.74409      0.72502       0.04247
```

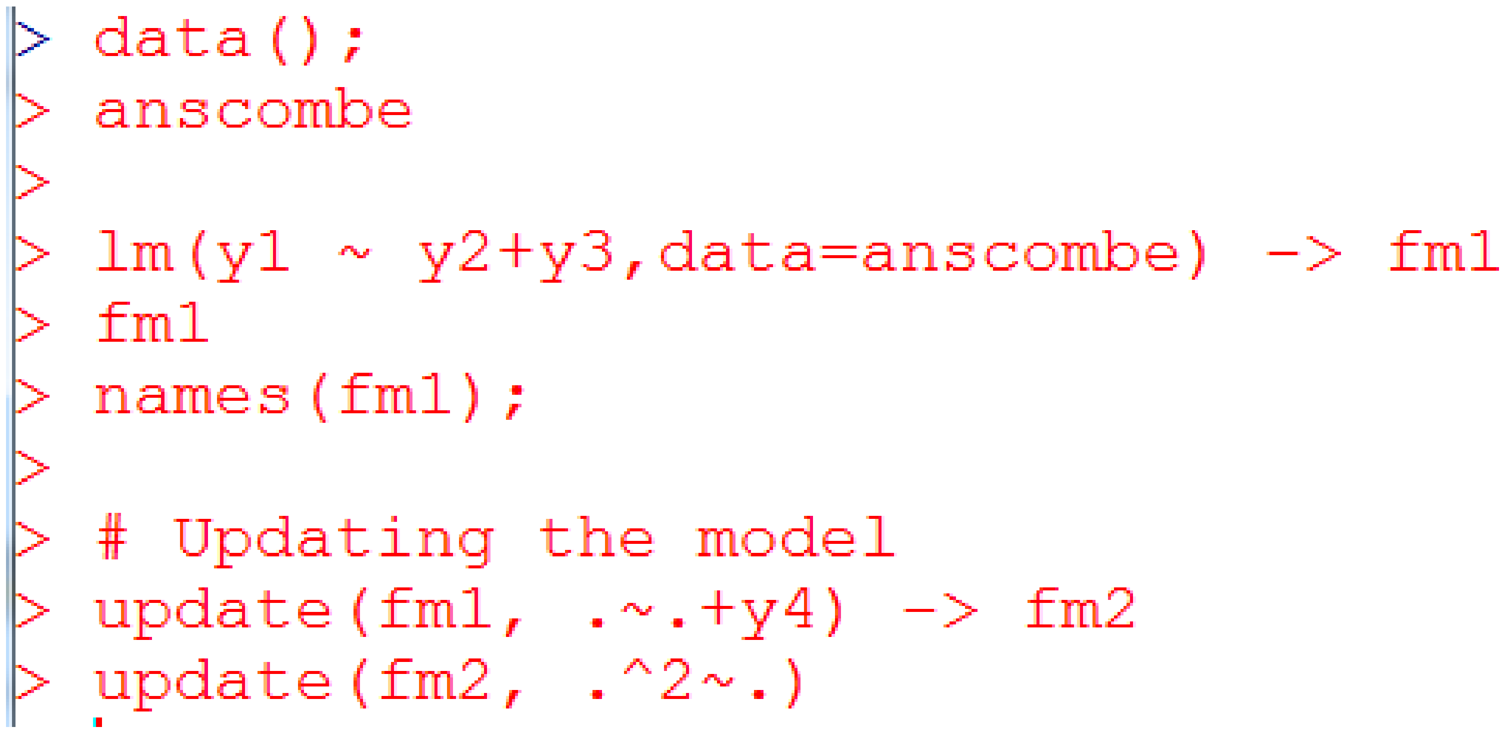
- How to specify “formula”?
  - Please refer to section 11.1
- “y1”, “y2” and “y3” are all vectors of the dataframe “anscombe”

Chapter 11. Statistical models in R

- How to perform the analysis of variance (ANOVA)?
  - Please refer to section 11.4
- How to perform the generalized linear model?
  - Please refer to section 11.6 for families, *glm()* function, the gaussian family, the binomial family, poisson models.
- How to perform optimizations?
  - Functions: *optim()*, *nlm()* *and* *nlminb()*.
- For some non-standard models,
  - please refer to section 11.8

Chapter 11. Statistical models in R

- Linear regression models.
  - Interpreting outputs.
  - Updating the model.



```
> data();
> anscombe
>
> lm(y1 ~ y2+y3,data=anscombe) -> fm1
> fm1
> names(fm1);
>
> # Updating the model
> update(fm1, .~.+y4) -> fm2
> update(fm2, .^2~.)
```

Chapter 11. Statistical models in R

Generating the linear regression model,

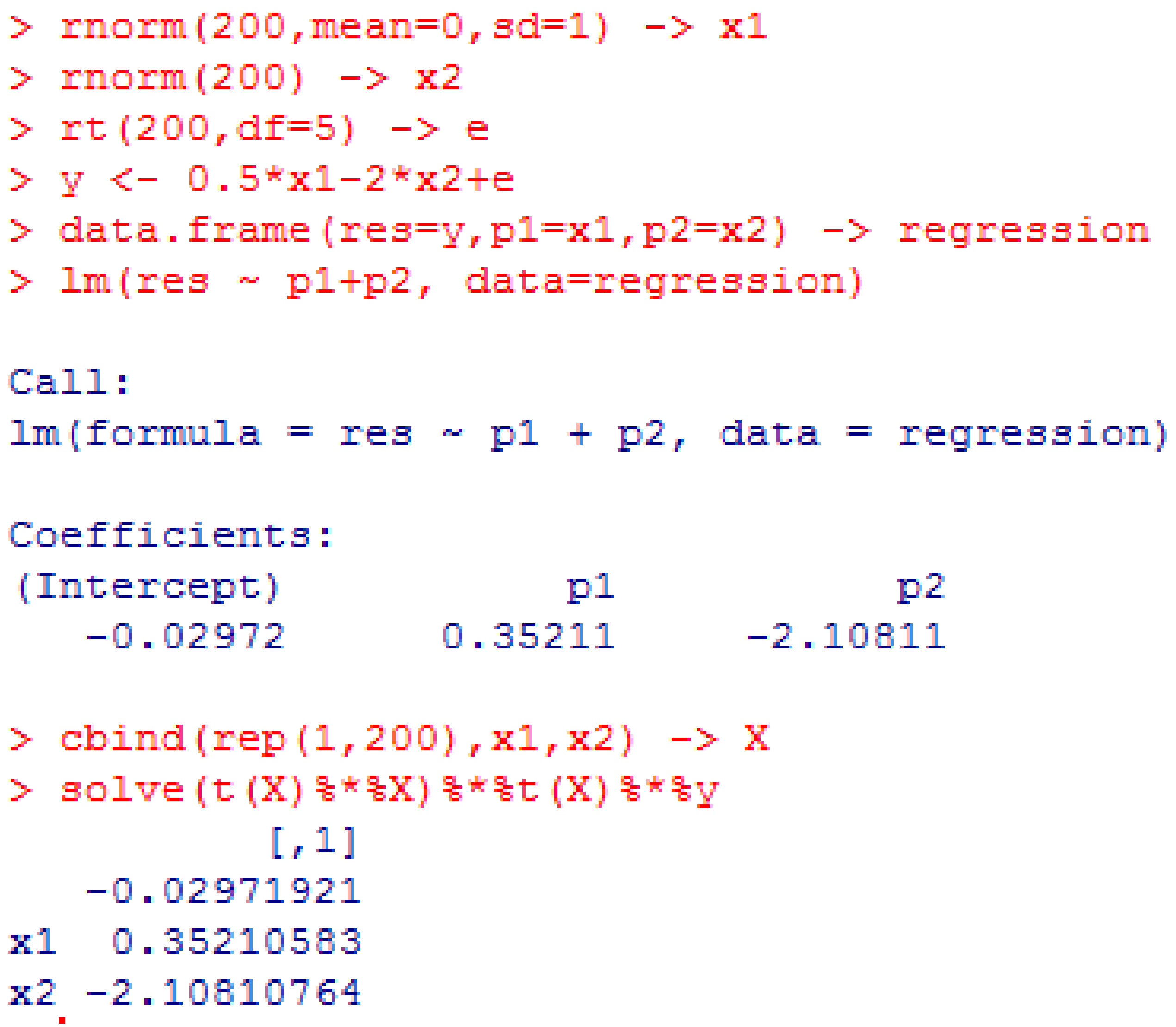
y=0.5*x1-2*x2+e,

where x1 and x2 are standard normal, e is Student's t(5), and sample size is 200.

- Calculate the regression coefficients
  - by fitting the model, and
  - by direct calculation.

$$\hat{\beta} = (X'X)^{-1}X'Y$$

Solutions



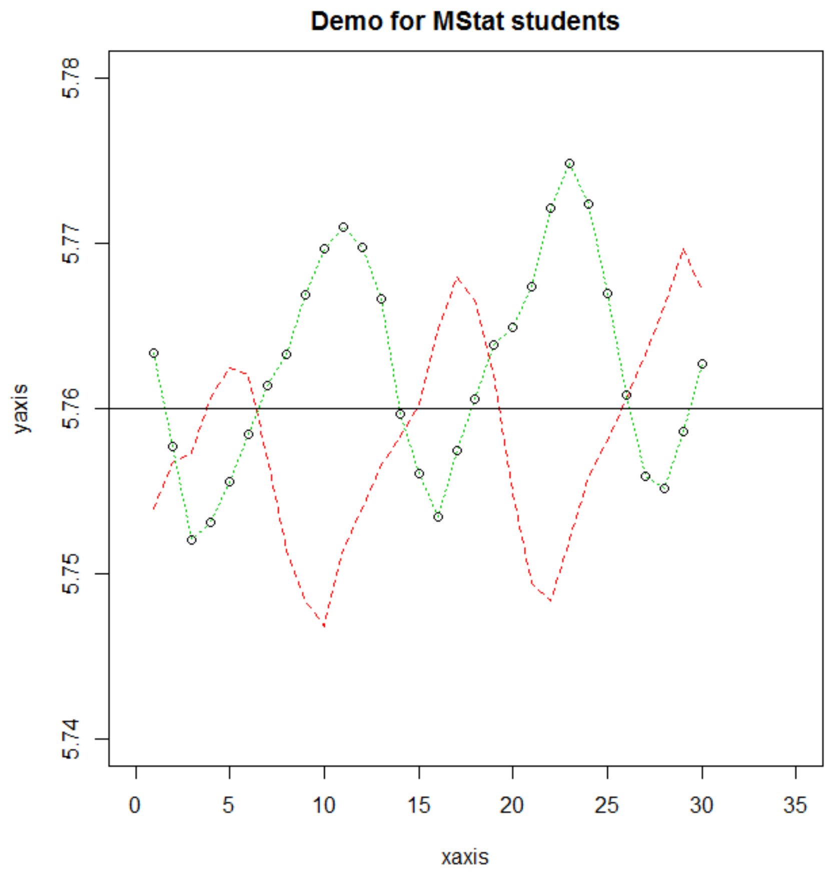
```
> rnorm(200,mean=0,sd=1) -> x1
> rnorm(200) -> x2
> rt(200,df=5) -> e
> y <- 0.5*x1-2*x2+e
> data.frame(res=y,p1=x1,p2=x2) -> regression
> lm(res ~ p1+p2, data=regression)

Call:
lm(formula = res ~ p1 + p2, data = regression)

Coefficients:
(Intercept)           p1           p2
   -0.02972      0.35211     -2.10811

> cbind(rep(1,200),x1,x2) -> X
> solve(t(X)%*%X)%*%t(X)%*%y
          [,1]
   -0.02971921
x1  0.35210583
x2 -2.10810764
```

Chapter 12. Graphical procedures

- Plotting commands are divided into three basic groups:
  - High-level plotting functions create a new plot on the graphics device, possibly with axes, labels, titles and so on.
  - Low-level plotting functions add more information to an existing plot, such as extra points, lines and labels.
  - Interactive graphics functions allow you interactively add information to, or extract information from, an existing plot, using a pointing device such as a mouse.
  - In addition, R maintains a list of graphical parameters which can be manipulated to customize your plots.
- Sections 12.1-12.3 are respectively for high-level, low-level and interactive functions. Section 12.4 & 12.5 are for graphical parameters.

Chapter 12. Graphical procedures

```
> par(mai=c(1,0.5,0.5,0))
> yy <- log(co2[1:30]); xx <- 1:30
> plot(y=yy,x=xx,type='l',col=2,lty=2,
+      xlim=c(0,35),ylim=c(5.74,5.78),
+      xlab='xaxis',ylab='yaxis',main='Demo for MStat students')
> y2 <- log(co2[31:60])
> lines(y=y2,x=xx,lty=3,col=3)
> points(y=y2,x=xx)
> abline(h=5.76)
```



Chapter 12. Graphical procedures

- Draw densities of the standard normal, t(3), t(10), t(20) in one plot.
- I require to use different colors, different line types, different line width.
- Draw vertical lines at 95% quantiles.