A brief introduction to R

Ionuţ Florescu
Dept. of Mathematical Sciences
Stevens Institute of Technology

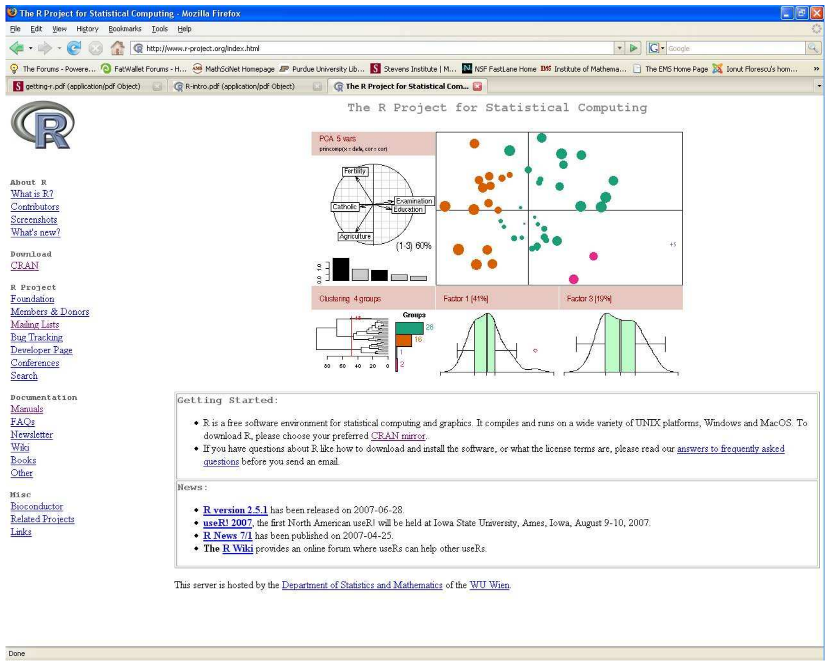
May 15, 2009

1 What is R? Where to obtain it?

R is a software package, one of the most versatile and useful in existence. In principal it is thought as a statistical software, however its versatility makes it useful for virtually any problem that requires use of software for its analysis.

1.1 Differences and similarities with other statistical software.

There is one important difference between R and any other professional software. **R IS FREE**. Normally, when one uses free versions one expects to use a stripped down, no good version of a commercial software. The reasoning in the commercial world we live in usually is: if it is valuable – it should cost money. *R is an exception.* It actually is more powerful that its commercial predecessor (S and Splus) and it is absolutely free. Even more, one can obtain support for it by subscribing to R-help mailing list, https://stat.ethz.ch/mailman/listinfo/r-help, which has over 200 posts every day. Amazingly, this free support was by far my best experience with support of any software I have ever used. If the question is interesting you should expect an answer very quickly. Careful what you post though, the people who maintain and answer questions are well known statistics professors and take very harshly at silly questions which could be easily answered by reading the introductory guides.


The R Project for Statistical Computing - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.r-project.org/index.html
The Forums - Powere... FatWallet Forums - H... MathSciNet Homepage Purdue University Lib... Stevens Institute | M... NSF FastLane Home Institute of Mathema... The EMS Home Page Ionut Florescu's hom...
getting-r.pdf (application/pdf Object)
R-intro.pdf (application/pdf Object)
The R Project for Statistical Com...
About R
What is R?
Contributors
Screenshots
What's new?
Download
CRAN
R Project
Foundation
Members & Donors
Mailing Lists
Bug Tracking
Developer Page
Conferences
Search
Documentation
Manuals
FAQs
Newsletter
Wiki
Books
Other
Misc
Bioconductor
Related Projects
Links
PCA 5 vars
princomp(x = data, cor = cor)
Fertility
Catholic
Examination
Education
Agriculture
(1-3) 60%
Clustering 4 groups
Factor 1 [41%]
Factor 3 [19%]
Groups
Done

The R Project for Statistical Computing

Getting Started:

- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To download R, please choose your preferred CRAN mirror.
- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News:

- **R version 2.5.1** has been released on 2007-06-28.
- **useR! 2007**, the first North American useR! will be held at Iowa State University, Ames, Iowa, August 9-10, 2007.
- **R News 7/1** has been published on 2007-04-25.
- **The R Wiki** provides an online forum where useRs can help other useRs.

This server is hosted by the Department of Statistics and Mathematics of the WU Wien.

Figure 1: R main page

1.1.1 Matlab

In spirit R is similar with Matlab through the uses of objects such as vectors and matrices rather than one dimensional variables. It is superior to Matlab when dealing with specific statistical analysis such as we see in the current course. However, linear algebra (matrix decomposition and numerical analysis of PDE is debatably faster with Matlab).

1.1.2 Splus

Commercial software similar with R. Very expensive. Predecessor to R. Going into point and click direction to try and differentiate itself from R. Not advised.

1.1.3 SAS

SAS is a commercial version used for statistical analysis. It is a strong software that essentially is different in approach and analysis from R. It is complementing R. Better and much faster at dealing with large datasets (millions of observations). Harder to program nested algorithms than R.

1.1.4 Spreadsheet software EXCEL, SPSS, Minitab etc.

These are all point and click programs. Not recommended for serious analysis. They can work with small data sets but crash quickly. They were designed to understand statistical concepts in undergraduate courses as well as manipulate data in a simple way. However, I would say their level is appropriate for high school students.

They are unbeatable however at quick and simple data manipulation. I use EXCEL to keep grades for this course and to calculate the final percentage. I also use it to quickly sort up to moderately sized data (up to 100000 observations).

1.1.5 Symbolic software: Mathematica, Maple etc.

These programs serve a different purpose than R. Mathematicians use these programs to essentially deal with non-random problems. Since statistics is by definition studying randomness they are not appropriate.

1.1.6 C, C++, JAVA

These are low level languages. Any program you write in any of the software mentioned thus far you should be able to program using low level language. The development of the program will take much longer, thus they are not suitable for use in a class such as the one presented. They, however have no equal when the speed of execution is an issue.

Conclusion: R pretty much has no equal. A good statistician should know how to use R, SAS and a low level language preferably C.

1.2 Obtaining R. Useful tools.

For a detailed description please see the file "R Installation and Administration" on the course web site or on the R web site directly (see Figure 1 on page 2).

You can find a wealth of information about R on the Manuals page and on the Wiki page for R.

The installation is straightforward and I expect students at Stevens to have no trouble with it. The last version at the time of this writing is 2.9 but the R versiuons evolve all the time adding new and better functionality. You should always install the last version.

In addition a very useful tool is an editor. For the Windows environment you could use Tinn-R, or if you use Notepad++ there is an extension called *npptor* that allows it to edit R code. http://sourceforge.net/projects/npptor/. For Linux you can use R commander (Rcmdr). If you use Eclipse for developing purposes you can set it up to work with R here is a document that may help: Setting up Eclipse with R.pdf. For MAC you have to find your own editor since I do not like that OS.

I personally use *Tinn-R*. I find it the easiest to use and it does its job very well in my opinion. I will explain its basic installation and configuration.

1.2.1 Installing Tinn-R under Windows

.

You can obtain Tinn-R from Sourceforge. The authors' web address is http://www.sciviews.org/Tinn-R/. You can access the download webpage from there.

The story about installing new versions is even more important with the editor since it depends on the main R program. At the time of this writing the Tinn-R version is 2.2.0.2. The installation and configuration may change from version to version so make sure that you read the documentation.

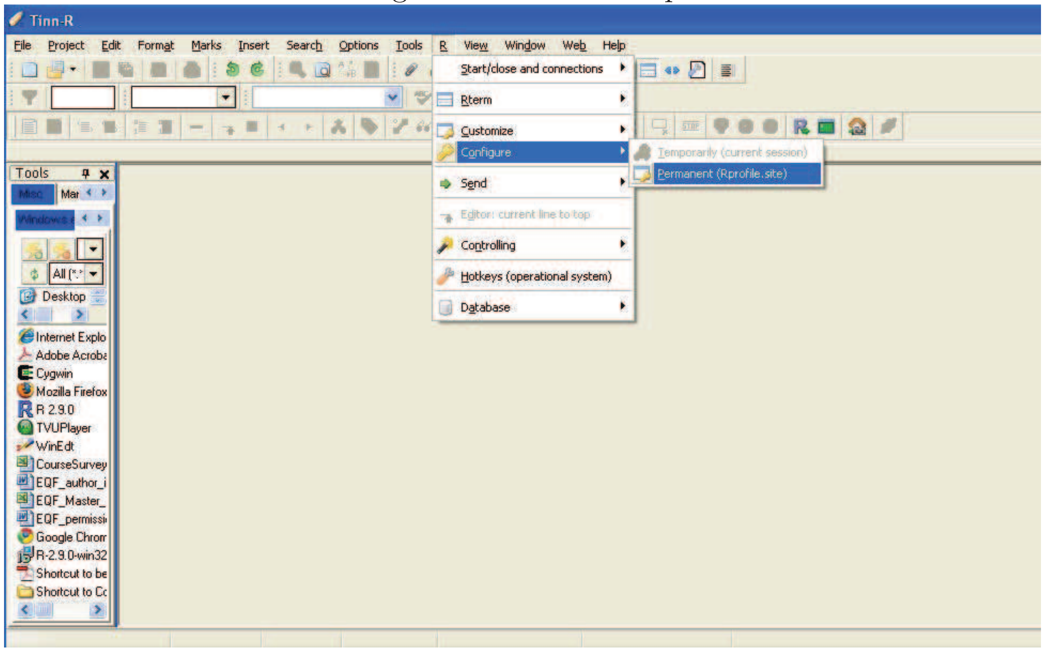
First install R. You may configure it however you want, I recommend installing the html version of help as it provides a familiar interface.

Then install Tinn-R. Use the default and do not forget to chose to associate it with .r files.

After this, start Tinn-R and configure it to work with R. To this end, choose R/Configure/Permanent(Rprofile.site) from the top menu and click on it. See Figure 2 bellow. You only need to do this once (first time you run the program). This will automatically configure everything. Chose to save the file (*Rprofile.site*) when prompted and close it. Then start R again when prompted or chose the button from the menu (see Figure3). This first time R will automatically connect to internet and download needed packages. It will ask you to chose a downloading site for this purpose. This is done only

the first time you start R.
That is all.

Figure 2: Tinn-R setup


Tinn-R
File Project Edit Format Marks Insert Search Options Tools R View Window Web Help
Start/close and connections
Rterm
Customize
Configure
Temporarily (current session)
Permanent (Rprofile.site)
Send
Editor: current line to top
Controlling
Hotkeys (operational system)
Database
Tools
Desktop

After each session R asks if you wish to save the data file. If you choose "Yes", R will save a R.data and a R.history files which contain the objects that you created during the session in the current working directory. Thus, next time when you work from the same directory the functions and variables that you created will be available to you. If you start R from within the editor the current working directory is set up to the directory of the current editor file.

I usually create directories (project directories) where I save the editor file. Starting R from inside the editor insures that the R.data and R.hist files for R are also saved in the project directory and they are loaded every time for the particular project. This insures that each time you work with functions and data specific to the project and that the data file does not grow extremely large.

2 Basic facts about R

In this section I want to make a brief introduction to the philosophy of R. For more detail read the manuals and the textbook.

First of all how do you get help from inside the program.

Figure 3 on page 7 details the steps in searching the description of R functions and objects using the included html help system. I found it very useful since the description of functions usually includes examples of its use.

2.1 Basic objects

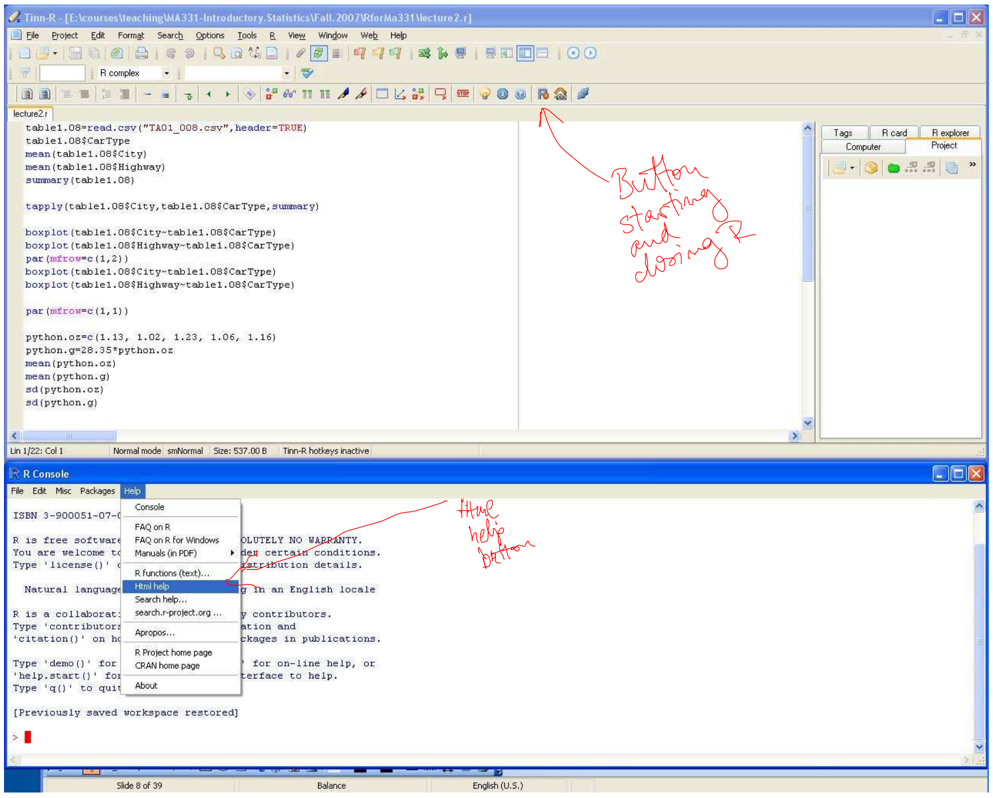
Basic object in R are vectors. The simplest way to cerate a vector is to specify its elements or *concatenate* them. This is done with the command **c**(). Look at the following code:

```
> x=c(1,2,3,4)
> x
[1] 1 2 3 4
> y<-c(1,2,3,4)
> y
[1] 1 2 3 4
> y->c(1,2,3,4)
Error in c(1, 2, 3, 4) <- y : target of assignment expands to
non-language object
```

Note that assignment is done using either = or object 1<−object 2; object 1 −> object 2. In the second case care has to be taken at what is assigned where. Historically, the second method was the only way to assign variables. Using = is recent and simpler in some cases.

Notice that by default R does not echo the value stored in x, that is why we had to call it again. The [1] in front of the output means that next to it is the value of the first element in the vector x. This makes it easy to see the indices of certain values. For example:

```
> x=rnorm(100)
> x
  [1]  0.237642561  0.876295252  1.974330233 -1.481039969
  [5]  1.351891376 -0.568888051  0.173439260 -2.484203697
  [9] -0.926123597 -0.372810408 -0.538202536 -0.060924253
 [13] -0.152219008  0.499729046 -0.223129476  0.668758491
 [17] -0.158439432  0.318846848 -1.525156072  0.180975917
```

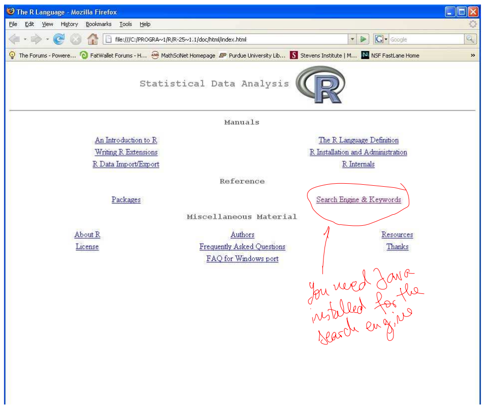

Tinn-R - [E:\courses\teaching\MA331-Introductory.Statistics\Fall.2007\RforMa331\lecture2.r]
File Project Edit Format Search Options Tools R View Window Web Help
R complex
lecture2.r
Tags R card R explorer
Computer Project
Button starting and closing R
R Console
File Edit Misc Packages Help
Console
FAQ on R
FAQ on R for Windows
Manuals (in PDF)
R functions (text)...
Html help
Search help...
search.r-project.org ...
Apropos...
R Project home page
CRAN home page
About
Html help button
Slide 8 of 39 Balance English (U.S.)

```
table1.08=read.csv("TA01_008.csv",header=TRUE)
table1.08$CarType
mean(table1.08$City)
mean(table1.08$Highway)
summary(table1.08)

tapply(table1.08$City,table1.08$CarType,summary)

boxplot(table1.08$City~table1.08$CarType)
boxplot(table1.08$Highway~table1.08$CarType)
par(mfrow=c(1,2))
boxplot(table1.08$City~table1.08$CarType)
boxplot(table1.08$Highway~table1.08$CarType)

par(mfrow=c(1,1))

python.oz=c(1.13, 1.02, 1.23, 1.06, 1.16)
python.g=28.35*python.oz
mean(python.oz)
mean(python.g)
sd(python.oz)
sd(python.g)
```

```
Lin 1/22: Col 1   Normal mode   smNormal   Size: 537.00 B   Tinn-R hotkeys inactive
```

```
ISBN 3-900051-07-0

R is free software                    OLUTELY NO WARRANTY.
You are welcome to                    der certain conditions.
Type 'license()' o                    stribution details.

  Natural language                    g in an English locale

R is a collaborati                    y contributors.
Type 'contributors                    ation and
'citation()' on ho                    ckages in publications.

Type 'demo()' for                     ' for on-line help, or
'help.start()' for                    terface to help.
Type 'q()' to quit

[Previously saved workspace restored]

>
```

(a) Starting R from Tinn-R and starting the html help


The R Language - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/PROGRA~1/R/R-25~1.1/doc/html/index.html
The Forums - Powere... FatWallet Forums - H... MathSciNet Homepage Purdue University Lib... Stevens Institute | M... NSF FastLane Home
You need Java installed for the search engine

Statistical Data Analysis

Manuals

An Introduction to R
Writing R Extensions
R Data Import/Export
The R Language Definition
R Installation and Administration
R Internals

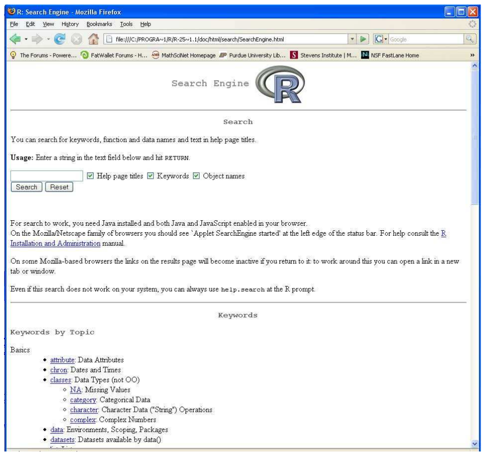
Reference

Packages
Search Engine & Keywords

Miscellaneous Material

About R
License
Authors
Frequently Asked Questions
FAQ for Windows port
Resources
Thanks

(b) The main html help page


R: Search Engine - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/PROGRA~1/R/R-25~1.1/doc/html/search/SearchEngine.html
The Forums - Powere... FatWallet Forums - H... MathSciNet Homepage Purdue University Lib... Stevens Institute | M... NSF FastLane Home

Search Engine

Search

You can search for keywords, function and data names and text in help page titles.

Usage: Enter a string in the text field below and hit RETURN.

[✓] Help page titles [✓] Keywords [✓] Object names

Search Reset

For search to work, you need Java installed and both Java and JavaScript enabled in your browser.
On the Mozilla/Netscape family of browsers you should see 'Applet SearchEngine started' at the left edge of the status bar. For help consult the R Installation and Administration manual.

On some Mozilla-based browsers the links on the results page will become inactive if you return to it: to work around this you can open a link in a new tab or window.

Even if this search does not work on your system, you can always use `help.search` at the R prompt.

Keywords

Keywords by Topic

Basics

- attribute: Data Attributes
- chron: Dates and Times
- classes: Data Types (not OO)
  - NA: Missing Values
  - category: Categorical Data
  - character: Character Data ("String") Operations
  - complex: Complex Numbers
- data: Environments, Scoping, Packages
- datasets: Datasets available by data()

(c) The search page

Figure 3: Getting help in R

```
 [21]  0.748301774  1.787367985  0.095750077  0.173754937
 [25]  0.786151309  1.764053131  0.704160617 −0.840505632
 [29]  0.285558128 −0.032446896 −1.228041496 −0.608433995
 [33] −1.356020869 −0.315292458 −1.157034611 −0.666455515
 [37] −0.952164495  0.002923223 −1.042511768 −2.106197568
 [41] −0.991882428 −1.137833460  0.642215596  0.021005544
 [45]  0.704502306  0.408772645  0.578284895  0.613572300
 [49]  0.330036063 −1.273529338  0.870827551  1.168062548
 [53]  0.332890027  2.574568298 −0.160492916  0.726620457
 [57]  0.088086003  1.433450143  0.319529910 −1.436210775
 [61] −0.922078075 −0.459300901  0.069265586  0.877002607
 [65]  0.584109383 −0.585970333 −0.563005373  2.336329173
 [69]  0.149144611 −0.028811727 −1.281079465 −0.208302014
 [73]  1.159875117 −1.637882479 −0.070133186  0.165062552
 [77] −0.928366237  0.218066972 −0.422775329 −0.903656363
 [81]  0.875540989 −0.638739741 −0.105328419 −0.601101522
 [85]  1.203358057  0.510160841  1.293336278  1.100393583
 [89] −0.356183702 −1.717985456  1.499592407  0.763135155
 [93]  0.156543343 −3.740034195 −0.565474655  0.152592785
 [97]  2.817728145  0.886599846 −1.977246574  0.758044979
```

I can see that the 50th element of the vector above is −1.273529338. By the way the function **rnorm**(100) generates 100 random numbers all independent distributed like a normal with mean 0 and variance 1.

An integer sequence such as one presented above can be created simpler with:

```
> 1:4
[1] 1 2 3 4
> seq(1,4,by=1)
[1] 1 2 3 4
```

The second way is more general. Read the manual.

Note that a vector in R has no dimension. Its values can be accessed as in the following:

```
> x=seq(5,7,by=0.1)
> x
 [1] 5.0 5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 6.0 6.1 6.2 6.3 6.4 6.5 6.6 6.7
> dim(x)
NULL
> length(x)
```

```
[1] 21
> x[4]
[1] 5.3
```

Once you bind two vectors together they become a matrix:

```
> cbind(x,1:length(x))
        x
 [1,] 5.0  1
 [2,] 5.1  2
 [3,] 5.2  3
 [4,] 5.3  4
 [5,] 5.4  5
 [6,] 5.5  6
 [7,] 5.6  7
 [8,] 5.7  8
 [9,] 5.8  9
[10,] 5.9 10
[11,] 6.0 11
[12,] 6.1 12
[13,] 6.2 13
[14,] 6.3 14
[15,] 6.4 15
[16,] 6.5 16
[17,] 6.6 17
[18,] 6.7 18
[19,] 6.8 19
[20,] 6.9 20
[21,] 7.0 21
```

Elements in a matrix are accessed using :

```
> z=cbind(x,1:length(x))
> z[1,2]
[1] 1
> z[1,]
x
5 1
> z[,2]
[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
```

The second and third call displays the first row and the first column respectively.

Notice that I didn't have to create a z variable. The following also displays the first row:

```
> cbind(x,1:length(x))[1,]
x
5 1
```

So what happens when the two vectors have different lengths? Observe:

```
> cbind(1:6,1:4)
     [,1] [,2]
[1,]    1    1
[2,]    2    2
[3,]    3    3
[4,]    4    4
[5,]    5    1
[6,]    6    2
Warning message:
number of rows of result is not a multiple of
vector length (arg 2) in: cbind(1, 1:6, 1:4)
```

Important. R works with objects. It does not care what dimension they are. If the operation can be done it is done. If not a warning or an error are generated. Look at the following example:

```
> 1+11
[1] 12
> 1:4+11:14
[1] 12 14 16 18
> 1:4+10:14
[1] 11 13 15 17 15 Warning message: longer object length
        is not a multiple of shorter object length in: 1:4 + 10:14
```

In all cases the addition operation was performed. Also note that the following does not produce a warning:

```
> 1:4+1
[1] 2 3 4 5
```

There are many other things I could try and describe. However, the best way to learn a language (any language) is to practice it.

One last issue I want to present is the creation of functions. For example

```
> test.fct=function(a,b=1){a+b}
> test.fct(1,3)
```

```
[1] 4
> test.fct(1)
[1] 2
> test.fct(,3)
Error in test.fct(, 3) : argument "a" is missing, with no default
> test.fct=function(a=2,b){a+b}
> test.fct(,3)
[1] 5
```

The code create a function that adds 2 objects. Note the default argument. One can call the function to perform addition on vectors:

```
> test.fct(1:4,11:14)
[1] 12 14 16 18
> test.fct(,11:14)
[1] 13 14 15 16
```

Also note that the second time you run the function the old function is rewritten. An easy way to check any function is to call its name without arguments:

```
> test.fct
function(a=2,b){a+b}
```

I will stop here and advice you to practice. As it is true about anything it is also true about R: *practice makes it perfect*.