R for Research

Abhay Singh

2020-06-29

Contents

Preface

This is a part-I of the compilation of handouts for R^2 $RforResearch$ on getting started with R.

This part will cover the following

- Getting Started
- R Data Types & Data Structures
- A Short Introduction to R Programming
- Data Exploration: Preprocessing, Transformation
- Graphics in R

Chapter 1

What is R?

According to the official webpage:

- R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R. http://www.r-project.org/about.html

According to Wikipedia

- R is a free software programming language and a software environment for statistical computing and graphics. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls and surveys of data miners are showing R's popularity has increased substantially in recent years.

To Summarise

- R is the most amazing free statistical software ever!
- This recent video by Revolution Analytics does a great job in summarizing R

!

1.1 Why should we learn R?

R follows a type inference [1] coding structure and provides a wide variety of statistical and graphical techniques, including;

- Linear and non-linear modelling
- Univariate & Multivariate Statistics
- Classical statistical tests
- Time-series analysis/ Econometrics
- Simulation and Modelling
- Datamining-classification, clustering etc.
- For computationally intensive tasks, C, C++, and Fortran code can be linked and called at run time.
- R is easily extensible through functions and extensions, and the R community is noted for its active contributions in terms of packages.

```
# Number of R Packages
length(available.packages(repos = "http://cran.us.r-project.org")[, 1])
```

```
[1] 15900
```

- Total 15900 packages and counting

1.2 Installing R and RStudio on Windows

- The latest version of R can be download from the R homepage.
- R download page: http://www.cran.r-project.org/bin/windows/base/ The page also provides some instructions and FAQ's on R installation.
- RStudio IDE (*IDE: Integrated Development Environment*) is a powerful and productive user interface for R.
- It's free and open source, and works great on Windows, Mac, and Linux

1.3 RStudio GUI/IDE

- RStudio GUI is composed of 4 panes which can be rearranged according to the requirements.
- There are a lot of short introductions to RStudio available online so we will not go into more details.

[1] Type inference refers to the automatic deduction of the type of an expression in a programming language.

The figure below gives the snapshot of RStudio GUI.

- A short intro to RStudio !
  RStudio Overview - 1:30 from RStudio, Inc. on Vimeo.

1.4 Installing Packages

- The easiest way to install packages is to do it via R console. The command install.packages("package name") installs R packages directly from internet. Other options to install various dependencies to a package can be easily specified when calling this function. A call to this function asks the user to chose a CRAN mirror at the first instance.
- Run the following to install Quantreg package on R. Also use the help function to get the details.

```
# Install a package using RStudio Console
install.packages("quantreg", dependencies = c("Depends", "Suggests"))
```

```
install.packages(c("zoo", "reshape2", "quantreg", "e1071", "foreign", "psych", "pastecs",
    "ggplot2", "stargazer", "formatR", "plm", "xts", "tseries", "fArma"), dependencies = TRUE)
# to be updated
```

1.5 Getting Help

As R is constantly evolving and new functions/packages are introduced every day it is good to know sources of help. The most basic help one can get is via the help() function. This function shows the help file for a function which has been created by package managers.

```
help("function name")
```

- The following can be used to search for a function etc.

```
#Replace the 'search string' with the expression you want to search
??search string
```

- All the R packages (with few exceptions) have a user's manual listing the functions in a package. This can be downloaded in PDF format from the R package download page[2].
- R also provides some search tools given at http://cran.r-project.org/search.html The R Site search is helpful in searching for topics related to problem in hand.

[2]For example reference manual for quantreg package is at http://cran.r-project.org/web/packages/quantreg/quantreg.pdf

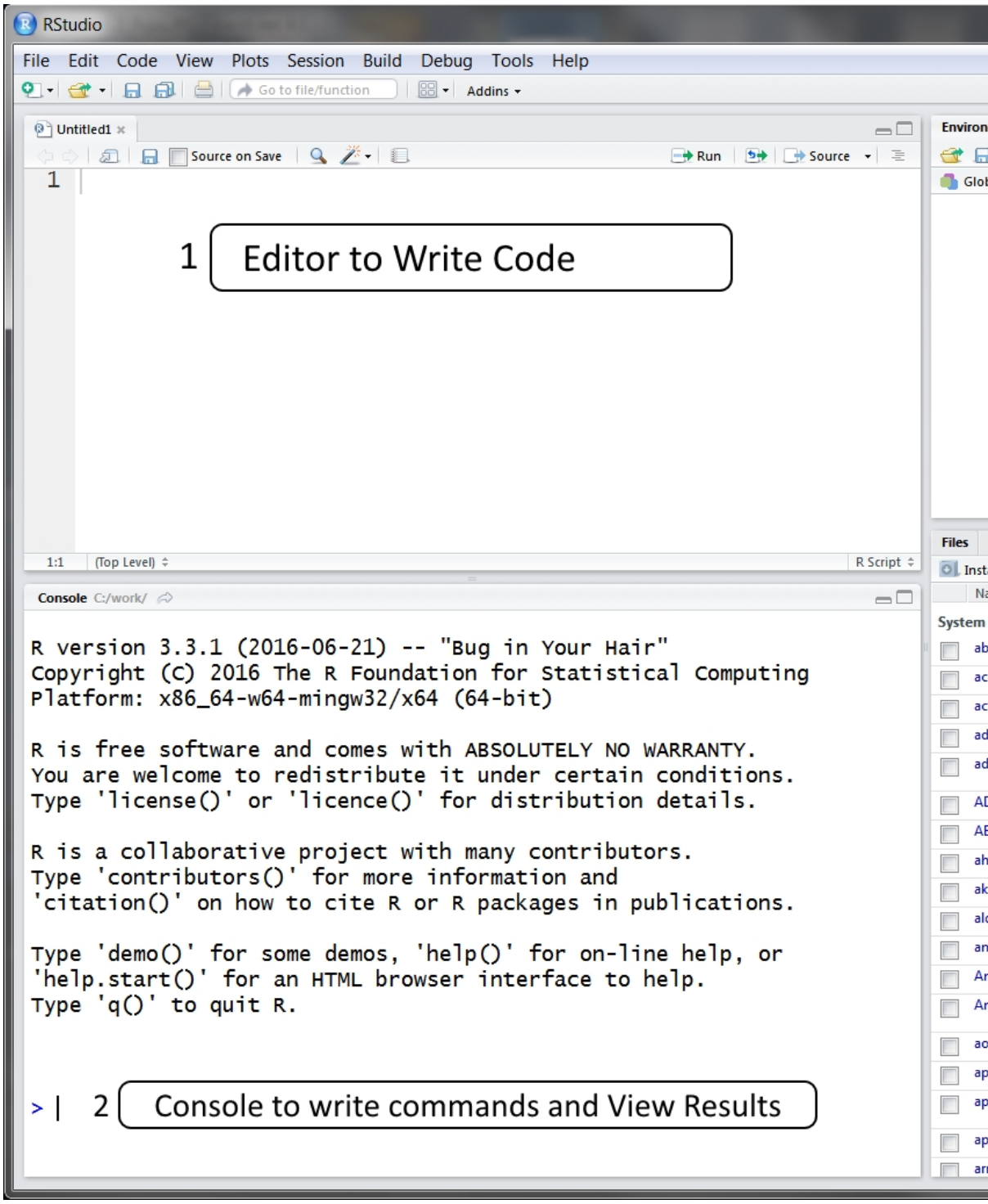


Figure 1.1: RStudio IDE

- Other than these various good R related blogs are on the internet which can be really helpful. A combined upto date view of 452 contributed blogs can be found at R-bloggers[3].
- Over all there quite a big community of R Users and help can be found for most of the topics.

1.6 Task Views in R-Introduction & Installation

- Task Views in R provide packages grouped together according to a generalized task they are used for.
- Table below gives the name of task views available[4].
- The following commands install the package *ctv* and then Finance task view.

```
# install package task views
install.packages("ctv")
library("ctv")  #R function library() is used to call a package
# install Finance task view
install.views("Finance")
```

1.7 R core packages

- R comes with few bundled core packages which provide various data analytics/statistical capabilities to R. The base package in R has basic functions and operators which are required for analytical programming, stats is another example of core R packages.

```
# List of R core packages
row.names(installed.packages(priority = "base"))
```

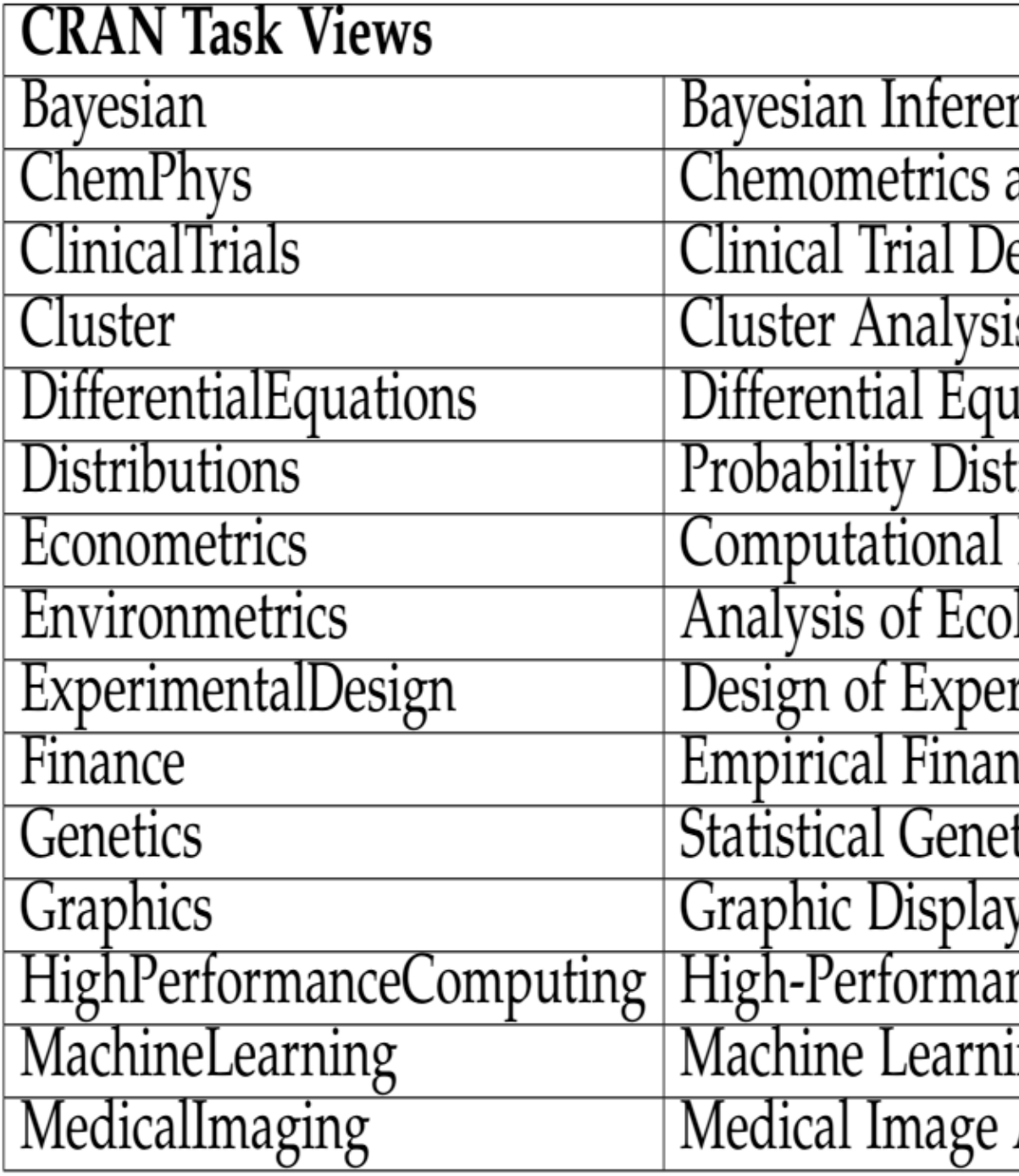
```
 [1] "base"      "compiler"  "datasets"  "graphics"  "grDevices" "grid"
 [7] "methods"   "parallel"  "splines"   "stats"     "stats4"    "tcltk"
[13] "tools"     "utils"
```

1.8 Example-1 Hello R!

```
message("Hello R!")  #use to display messages
```

```
Hello R!
```

[3] Go to [www.r-bloggers.com]: www.r-bloggers.com
[4] This list of available task views can be found at http://cran.r-project.org/web/views/



CRAN Task Views	
Bayesian	Bayesian Inferen
ChemPhys	Chemometrics a
ClinicalTrials	Clinical Trial De
Cluster	Cluster Analysis
DifferentialEquations	Differential Equ
Distributions	Probability Dist
Econometrics	Computational
Environmetrics	Analysis of Ecol
ExperimentalDesign	Design of Exper
Finance	Empirical Finan
Genetics	Statistical Genet
Graphics	Graphic Display
HighPerformanceComputing	High-Performan
MachineLearning	Machine Learni
MedicalImaging	Medical Image

Figure 1.2: Task Views

```
print("Hello R!")  #use to display variables/messages
```

```
[1] "Hello R!"
```

```
msg = "Hello R!"  #type inference no need to define strings!
print(msg)
```

```
[1] "Hello R!"
```

- R packages not just come with demo programs but the help files for the functions in R packages mostly have example codes for the particular function. R function example() is helpful in running the example code for a function. For running example code for in quantreg package type `example(rq, package=quantreg)`

Chapter 2

R Data Types and Data Structures

When human judgement and big data intersect there are some funny things that happen. -Nate Silver

2.1 Data Types

- As per R's official language definitions; in every computer language variables provide a means of accessing the data stored in memory.
- R does not provide direct access to the computer's memory but rather provides a number of specialized data structures we will refer to as objects. These objects are referred to through symbols or variables.

2.1.1 Double

- Doubles are numbers like 5.0, 5.5, 10.999 etc. They may or may not include decimal places. Doubles are mostly used to represent a continuous variable like serial number, weight, age etc.

```
x = 8.5
is.double(x)  #to check if the data type is double
```

```
[1] TRUE
```

2.1.2 Integer

- Integers are natural numbers.

```
x = 9
typeof(x)
```

```
[1] "double"
```

```
# The following specifically assigns an integer to x

x = as.integer(9)
typeof(x)
```

```
[1] "integer"
```

2.1.3 Logical

- A variable of data type logical has the value TRUE or FALSE. To perform calculation on logical objects in R the FALSE is replaced by a zero and TRUE is replaced by 1.

```
x = 11
y = 10
a = x > y
a
```

```
[1] TRUE
```

```
typeof(a)
```

```
[1] "logical"
```

2.1.4 Character

- Characters represent the string values in R. An object of type character can have alphanumeric strings. Character objects are specified by assigning a string or collection of characters between double quotes (“ string”) . Everything in a double quote is considered a string in R.

2.1.5 Factor

-Factor is an important data type to represent categorical data. This also comes handy when dealing with Panel or Longitudinal data. Example of factors are Blood type (A , B, AB, O), Sex (Male or Female). Factor objects can be created from character object or from numeric object. -The operator c is used to create a vector of values which can be of any data type.

```
b.type = c("A", "AB", "B", "O")  #character object
# use factor function to convert to factor object
b.type = factor(b.type)
b.type
```

```
[1] A  AB B  O
Levels: A AB B O
```

```
# to get individual elements (levels) in factor object
levels(b.type)
```

```
[1] "A"  "AB" "B"  "O"
```

2.1.6 Date & Time

-R is capable of dealing calendar dates and times. It is an important object when dealing with time series models. The function `as.Date` can be used to create an object of class Date. - see `help(as.Date)` for more details about the format of dates.

```
date1 = "31-01-2012"
date1 = as.Date(date1, "%d-%m-%Y")
date1
```

```
[1] "2012-01-31"
```

```
data.class(date1)
```

```
[1] "Date"
```

```
# The date and time are internally interpreted as Double so the function typeof
# will return the type Double
typeof(date1)
```

```
[1] "double"
```

2.2 Data Structures in R

Every data analysis requires the data to be structured in a well defined way. These coherent ways to put together data forms some basic data structures in R. Every data set intended for analysis has to be imported in R environment as a data structure. R has the following basic data structures:

Vector

Matrix

Array

Data Frame

Lists

2.2.1 Vector

- Vectors are group of values having same data types.

- There can be numeric vectors, character vector and so on. Vectors are mostly used to represent a single variable in a data set.
- A vector is constructed using the function `c`. The same function `c` can be used to combine different vectors of same data type.

```
vec1 = c(1, 2, 3, 4, 5)
vec1
```

```
[1] 1 2 3 4 5
```

The `str` function can be used to view the data structure of an object

2.2.2 Matrices

- A matrix is a collection of data elements arranged in a two-dimensional rectangular layout. Like vectors all the elements in a matrix are of same data type.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

- The function `matrix` is used to create matrices in R. Note that all the elements in a matrix object are of same basic type. Lets create the matrix in the example above.

```
m1 = matrix(c(1, 2, 3, 4, 5, 6), nrow = 3, ncol = 2, byrow = TRUE)
# nrow-specify number of rows, ncol-specify number of columns, byrow-fill the
# matrix in rows with the data supplied
m1  #print the matrix
```

```
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

- A vector can be converted to matrix using `dim` function, e.g:

```
m2 = c(1, 2, 3, 4, 5, 6)
dim(m2) = c(3, 2)  #the matrix will be filled by columns
m2
```

```
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

```
# use dim to get the dimension (#rows and #columns) of a matrix
dim(m1)
```