Fundamentals of R

Understanding *R*

R is a programming language and software environment for statistical computation and graphics

R is an open-source program and companies as diverse as Google, Pfizer, Merck, Bank of America, the InterContinental Hotels Group and Shell use it.

History of *R*

R was created in 1996 by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand

According to them, the notion of devising something like *R* sprang up during a hallway conversation.

They both wanted technology better suited for their statistics students, who needed to analyse data and produce graphical models of the information.

Most comparable software had been designed by computer scientists and proved hard to use.

Naming of *R*

R is named partly after the first names of the first two *R* authors and partly as a play on the name of ‘*S*’

R is majorly an implementation of the *S* programming language. S is not an open-source project.

The *R* Programming Language

The core of *R* is an interpreted computer language which allows branching and looping as well as modular programming using functions.

The source code for the *R* software environment is written primarily in *C*, F*ortran*, and *R*. Most of the user-visible functions in *R* are written in *R*.

While *R* has a command line interface, there are several graphical front-ends available.

Advanced users can write *C*, *C++*, *Java*, *.NET* or *Python* (and many other) code to manipulate *R* objects directly.

Functionality Supported by *R*

The R distribution contains functionality for a large number of statistical procedures.

Among these are:
- linear and generalized linear models,
- nonlinear regression models,
- time series analysis,
- classical parametric and nonparametric tests,
- clustering
- smoothing

There is also a large set of functions which provide a flexible graphical environment for creating various kinds of data presentations.

Additional modules (“add-on packages”) are available for a variety of specific purposes. Close to 12,000 different packages reside on just one of the many Web sites devoted to *R*, and the number of packages has grown exponentially to more than 20000.

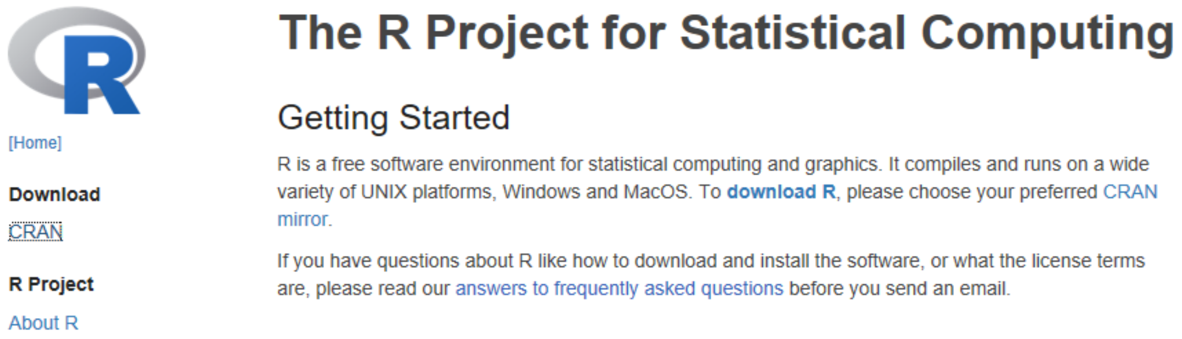
Understanding Comprehensive R Archive Network (CRAN)

- The "Comprehensive R Archive Network" (CRAN) is a collection of sites which carry identical material, consisting of the R distribution(s), the contributed extensions, documentation for R, and binaries.
- The CRAN master site at WU (Wirtschaftsuniversität Wien) in Austria can be found at the URL

  https://CRAN.R-project.org/
- It is mirrored daily to many sites around the world. Indian site is hosted at IIT Madras
- From CRAN, one can obtain (amongst other things) the latest official release of R and prebuilt binaries for various operating systems (Linux, Mac OS Classic, macOS, and MS Windows)

How to Install R on Operating Systems

1. Go to the site https://www.r-project.org/
2. Click on CRAN link below Download



[Home]

Download

CRAN

R Project

About R

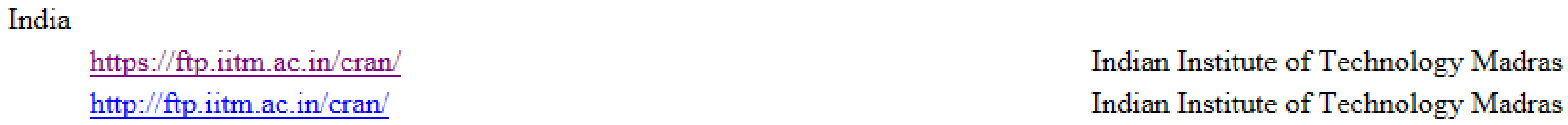
The R Project for Statistical Computing

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.

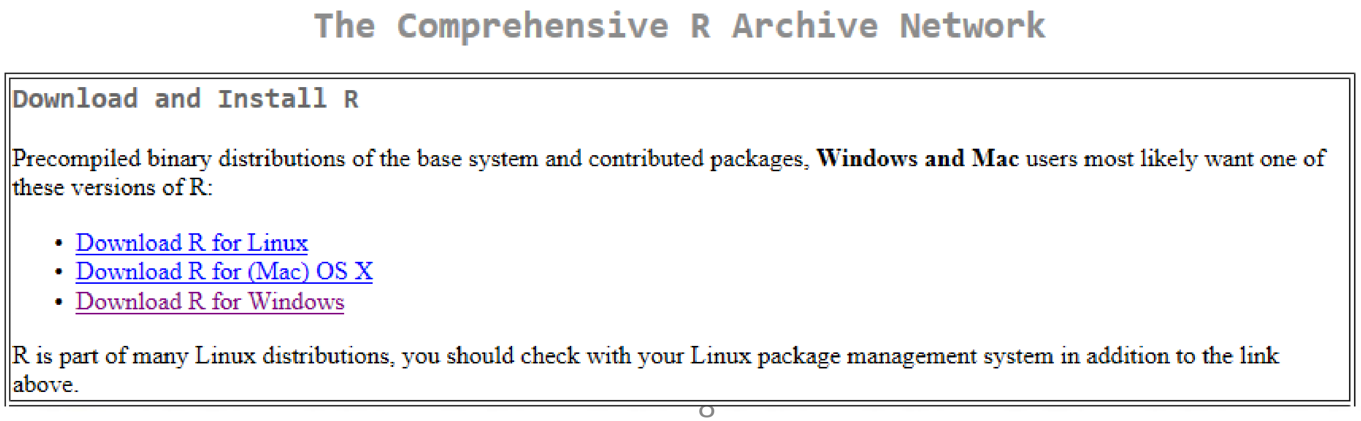
3. Select the CRAN mirror closest to you



India

https://ftp.iitm.ac.in/cran/	Indian Institute of Technology Madras
http://ftp.iitm.ac.in/cran/	Indian Institute of Technology Madras

4. Download R and install for your operating system



The Comprehensive R Archive Network

Download and Install R

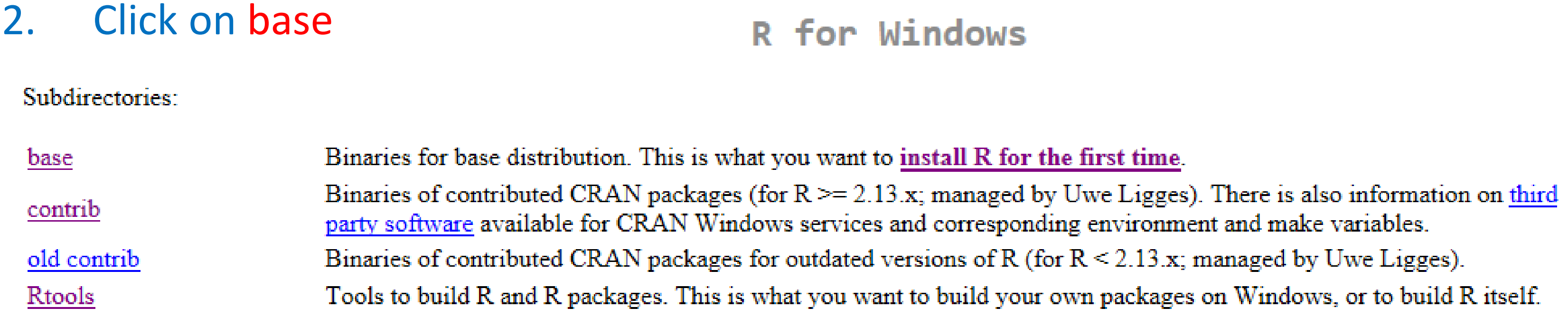
Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

How to Install R on Windows from CRAN website

1. Click on Download R for Windows
2. Click on base

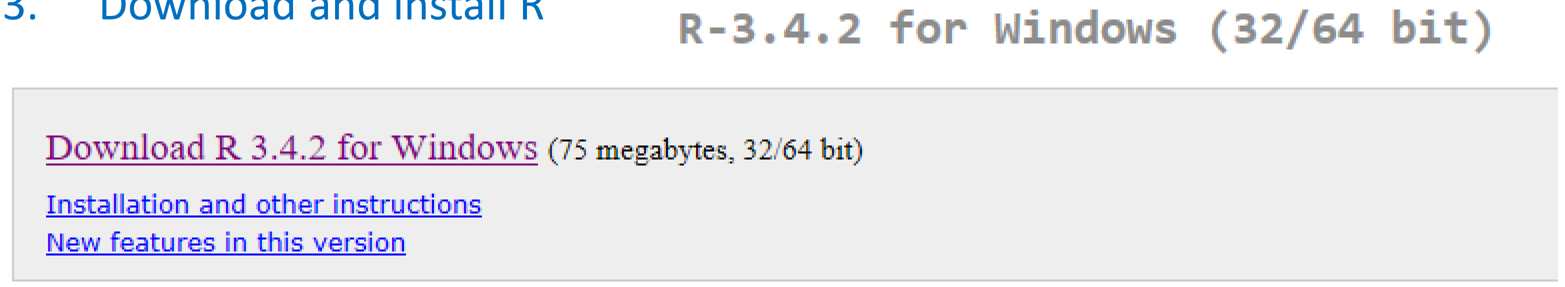


R for Windows

Subdirectories:

base	Binaries for base distribution. This is what you want to install R for the first time.
contrib	Binaries of contributed CRAN packages (for R >= 2.13.x; managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).
Rtools	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

3. Download and install R



R-3.4.2 for Windows (32/64 bit)

Download R 3.4.2 for Windows (75 megabytes, 32/64 bit)

Installation and other instructions

New features in this version

4. The distribution is distributed as an installer R-3.4.2-win.exe. Just run this for a Windows-style installer. There are two versions of the R executable in R-3.4.2\bin\i386 (32-bit) and R-3.4.2\bin\x64 (64-bit). By default only the first is installed on 32-bit versions of Windows, and both on 64-bit OS.
5. Double click on R icon to open the R GUI/console

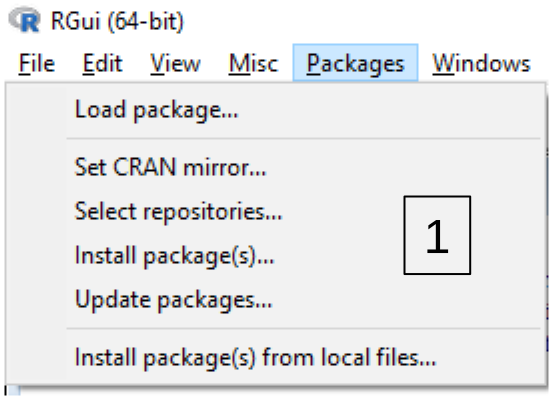
IDEs for R

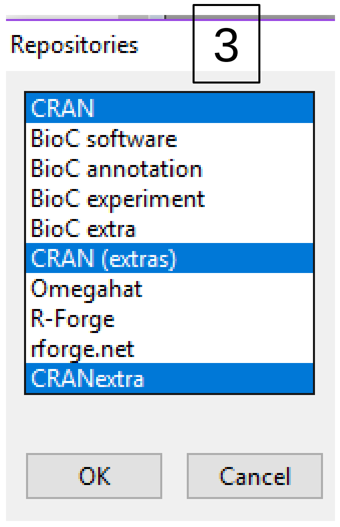
- There are various IDEs for R available:
  - R Studio
  - Red-R
  - Rattle
  - Visual Studio for R (Microsoft)
  - Eclipse (with Statet plugin for R) (Good for integration with non-R projects)
  - RKWard
  - TinnR
  - Emacs with ECB and ESS Package
  - Vim
- RStudio (http://www.rstudio.org/) is by far considered to be the leader by most users of R

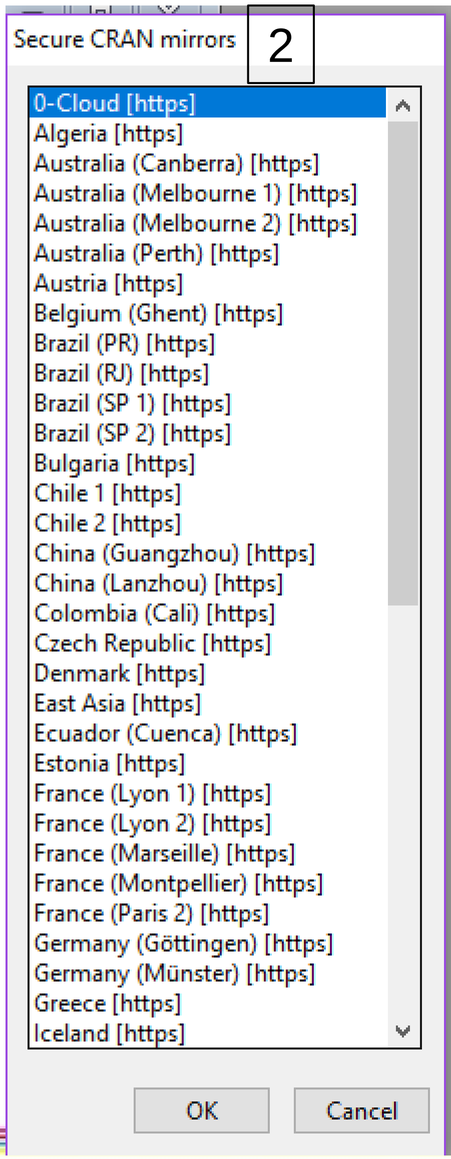
https://www.rstudio.com/products/rstudio/download/

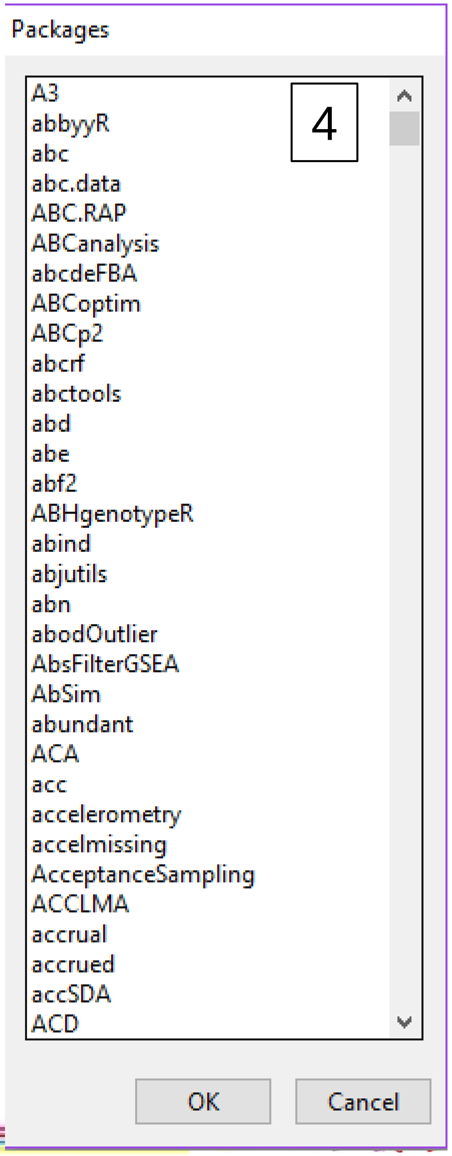
R Package Installation and Practice – R GUI

1. Open R GUI and Click on Package
2. Set CRAN mirror
3. Select Repositories
4. Install Packages (Select packages and click OK)



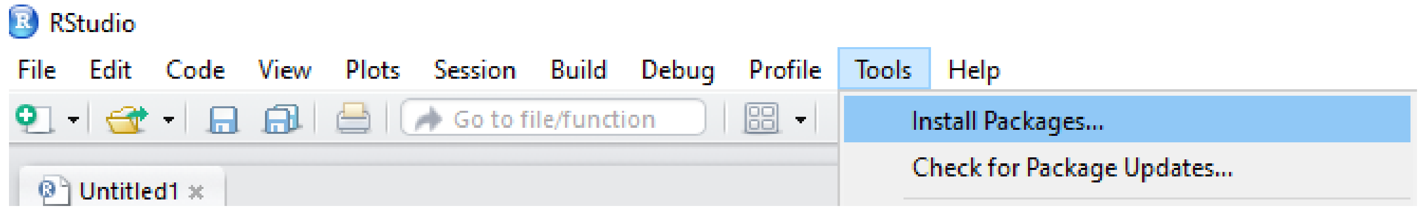




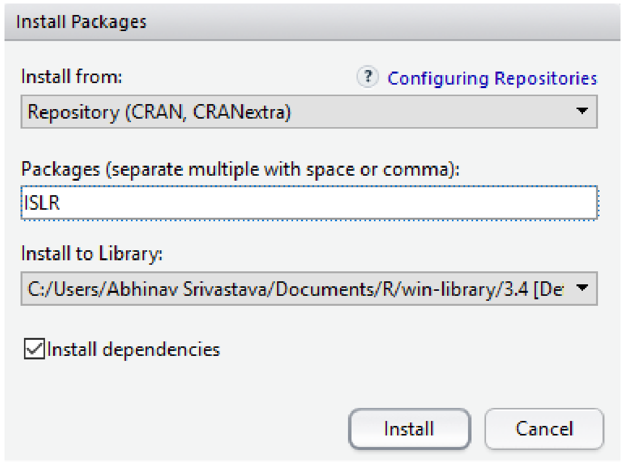


R Package Installation and Practice – R Studio

1. Open R Studio
2. Select Tools→ Install Packages


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Untitled1
Install Packages...
Check for Package Updates...

3. In Install from select Repository(CRAN, CRANextra) → Provide Package name → leave other parameter as default and click on Install



Install Packages

Install from: Repository (CRAN, CRANextra)

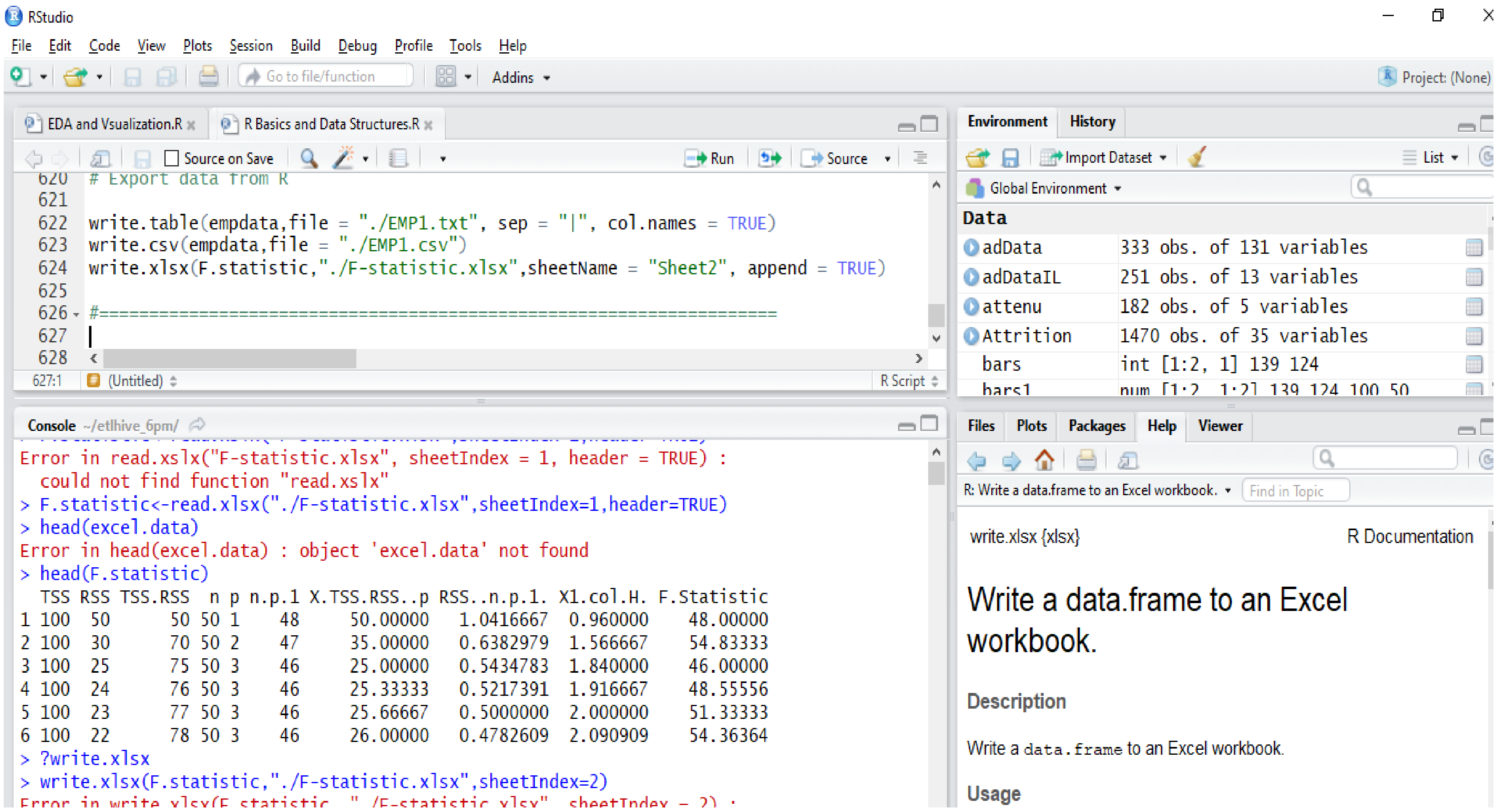
Packages (separate multiple with space or comma): ISLR

Install to Library: C:/Users/Abhinav Srivastava/Documents/R/win-library/3.4 [De

[✓] Install dependencies

4. Package with all dependent packages will be downloaded and installed.

R Studio GUI



```
# Export data from R

write.table(empdata,file = "./EMP1.txt", sep = "|", col.names = TRUE)
write.csv(empdata,file = "./EMP1.csv")
write.xlsx(F.statistic,"./F-statistic.xlsx",sheetName = "Sheet2", append = TRUE)

#===================================================================

```

Data		
adData	333 obs. of 131 variables	▦
adDataIL	251 obs. of 13 variables	▦
attenu	182 obs. of 5 variables	▦
Attrition	1470 obs. of 35 variables	▦
bars	int [1:2, 1] 139 124	▦
bars1	num [1:2, 1:2] 139 124 100 50	▦

```
Error in read.xslx("F-statistic.xlsx", sheetIndex = 1, header = TRUE) :
  could not find function "read.xslx"
> F.statistic<-read.xlsx("./F-statistic.xlsx",sheetIndex=1,header=TRUE)
> head(excel.data)
Error in head(excel.data) : object 'excel.data' not found
> head(F.statistic)
  TSS RSS TSS.RSS  n p n.p.1 X.TSS.RSS..p RSS..n.p.1. X1.col.H. F.Statistic
1 100  50      50 50 1    48     50.00000   1.0416667  0.960000    48.00000
2 100  30      70 50 2    47     35.00000   0.6382979  1.566667    54.83333
3 100  25      75 50 3    46     25.00000   0.5434783  1.840000    46.00000
4 100  24      76 50 3    46     25.33333   0.5217391  1.916667    48.55556
5 100  23      77 50 3    46     25.66667   0.5000000  2.000000    51.33333
6 100  22      78 50 3    46     26.00000   0.4782609  2.090909    54.36364
> ?write.xlsx
> write.xlsx(F.statistic,"./F-statistic.xlsx",sheetIndex=2)
Error in write.xlsx(F.statistic, "./F-statistic.xlsx", sheetIndex = 2) :
```

write.xlsx {xlsx} R Documentation

Write a data.frame to an Excel workbook.

Description

Write a `data.frame` to an Excel workbook.

Usage

Understanding R Programming

- R is an integrated suite of software facilities for data manipulation, calculation and graphical display. Among other things it has
  - an effective data handling and storage facility,
  - a suite of operators for calculations on arrays, in particular matrices,
  - a large, coherent, integrated collection of tools for data analysis,
  - graphical facilities for data analysis and display either directly at the computer or on hard-copy, and
  - a well developed, simple and effective programming language which includes conditionals, loops, user defined functions and input and output facilities.
- The term “environment” is intended to characterize it as a fully planned and coherent system

Getting Help in R

- To get information on any named function for example “solve” you use:

```
>help(solve)    or   >?solve
```

- For a feature specified by special characters, the argument must be enclosed in double or single quotes, making it a “character string”: This is also necessary for a few words with syntactic meaning including if, for and function.

```
>help(“[[“)      >help(“if”)
```

- On most R installations help is available in HTML format by running

```
>help.start()
```

To find a specific key word, use ??

```
> ??logit
```

- apropos() function will return a list of all functions known to R containing the search term (using partial match)

```
>  apropos(“mean”)
```

Naming in R

- R has simple syntax
- It is case sensitive so A and a are different symbols and would refer to different variables
- Normally all alphanumeric symbols are allowed (plus '.' and '_')
- Name must start with '.' or a letter, and if it starts with '.' the second character must not be a digit

R Elementary Commands

- Elementary commands consist of either expressions or assignments.
- If an expression is given as a command, it is evaluated, printed, and the value is lost.
- An assignment also evaluates an expression and passes the value to a variable but the result is not automatically printed.

Assignment Operator ‘<-’

- Notice that the assignment operator (‘<-’), which consists of the two characters ‘<’ (“less than”) and ‘-’ (“minus”) occurring strictly side-by-side and it ‘points’ to the object receiving the value of the expression.
- In most contexts the ‘=’ operator can be used as an alternative.
- Assignment can also be made using the function assign(). An equivalent way of making the same assignment as above is with:

```
> assign("x", 1)
```

- The usual operator, <-, can be thought of as a syntactic short-cut to this.

Expression command

- If an *expression* is used as a complete command, the value is printed and lost

- So now if we were to use the command

```
> 1/x
```

  the reciprocals of the five values would be printed at the terminal

- The value of x will remain unchanged

Studying Operators in R

- Arithmetic Operators: +, -, *, /, ^ (for raising to power)

  %% Remainder, %/% Quotient

- Logical Operator:
  - The logical operators are <, <=, >, >=, == for exact equality and != for inequality. They are sometimes called **relational** operators also.
  - In addition if c1 and c2 are logical expressions, then c1 & c2 is their intersection (“and”), c1 | c2 is their union (“or”), and !c1 is the negation of c1
  - Long form of & and | i.e. && and || is also used which evaluates left to right, the first term only and returns the result. c1 & &c2 (“and”), c1 || c2 is (“or”)
  - Logical vectors may be used in ordinary arithmetic, in which case they are coerced into numeric vectors, FALSE becoming 0 and TRUE becoming 1.
  - The elements of a logical vector can have the values TRUE, FALSE, and NA

Grouping R Elementary Commands

- Commands are separated either by a semi-colon (‘;’), or by a newline.
- Elementary commands can be grouped together into one compound expression by braces (‘{’ and ‘}’).
- Comments can be put almost anywhere, starting with a hashmark (‘#’), everything to the end of the line is a comment.
- If a command is not complete at the end of a line, R will give a different prompt, by default

```
+
```

on second and subsequent lines and continue to read input until the command is syntactically complete.

Statements in R Programming

- A statement is the smallest standalone element which expresses some action to be carried out
- Statements in R can broadly be classified into two categories
  - Conditional Statements
  - Loops

Conditional Statements - If else () Function

- The language has available a conditional construction of the form

  if (expr_1) expr_2 else expr_3

- where expr_1 must evaluate to a **single** logical value. If expr_1 evaluates to TRUE then expr_2 is evaluated otherwise expr_3 is evaluated

- Another simpler form is below where FALSE branch is not evaluated

  if (expr_1) expr_2

- The “short-circuit” operators && and || are often used as part of the condition in an if statement
- Nested if else is also available in R

  if (expr_1) expr_2 else if (expr_3) expr_4 else expr_5

Switch Function

- This is a convenient form of multiple if-else statement.
- A switch statement allows a variable to be tested for equality against a list of values. Each value is called a Case and the variable is checked for each case one by one

Switch(expression, case1,case2,case3,…)

- If the value of variable is a character string then the string is matched to the names of the element and the value corresponding to first matching argument is returned
- In case of no match, if there is an unnamed element, it’s value is returned. If there is more than one match, then first value is returned
- If the value of variable is not a character string, then it is coerced to integer. The Case corresponding to this integer is evaluated and the value is returned. No default argument is available.

Loops in R

for Loop

- There is also a for loop construction which has the form

  > for (name in expr_1) expr_2

- where name is the loop variable
- expr_1 is a vector expression, (often a sequence like 1:20), and expr_2 is often a grouped expression
- expr_2 is repeatedly evaluated as name ranges through the values in the vector result of expr 1

```
> j=0
for (i in 1:10) {
+ j<-j+I
+ }
> j
[1] 55
```

Loops in R

- Other looping facilities include the following statements:

repeat expr

while (condition) expr

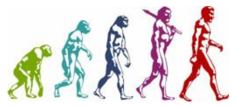
Break and Next Statement

- The break statement can be used to terminate any loop, possibly abnormally. This is the only way to terminate repeat loops.

- The next statement can be used to discontinue one particular cycle and skip to the “next”.

R Functions: Commonly used Functions and String Functions

- R comes with many preinstalled functions. All functions have name and take arguments in parentheses: function(…)
  - print(1)
  - exp(x=1)
  - log(x = 10, base = 10), log (x = 10, base = exp(1)), log (x=10)
  - sqrt(9)
  - abs(-5.5)
  - sin(pi/6)
  - factorial(5)
  - round(pi,2)
  - sign(-3), sign(0/0)



R Functions: Commonly used Functions and String Functions

String Functions: Character strings are enclosed by quotation marks “ ”

- print(“This is a character string”)
- substr(x, start=10, stop=15)
- strsplit(m1,split =" ")
- nchar(“This is a character string”)
- toupper(“This is a character string”)
- tolower(“This is a character string”)
- m1<-“This is a character string”;paste(m1,m1,sep = ", ") # concatenate
- m2<-c("do","do not","not")
- grep("do",m2)   # pattern matching
- m2<-c(m2,"1")
- grep("[a-z]",m2)
- gsub("do","did",m2)   # pattern replacement
- m2<-c(m2,“not do")
- regexpr("do",m2)   # position of match

Thank You

Abhinav Srivastava