I.
Getting Started

Kazumi Wada
National Statistics Center, JAPAN

Contents

1. Purpose of this chapter

This is the first chapter of the course which aims to illustrate the basic characteristics of R with simple exercises. These exercises help to get used to the software.

2. What is R?

- R is a language and environment for statistical computing and graphics.
- R is not a commercial software. It is available as "Free Software" under the terms of the Free Software Foundation's GNU General Public License in source code form. Its copyright has not been abandoned (copy left), but you can use, share, change and distribute copies.
  - GNU General Public License https://www.gnu.org/licenses/gpl-3.0.en.html
- R is developed and maintained by the R Development Core Team. The R Development Core Team founded the R Foundation, which is a non-profit organization.
- Please see the following site for more details.
  - The R Project.org https://www.r-project.org/index.html

3. Why we use R?

- R is free of charge and cross-platform software (Windows/Mac/Linux).
- R has a huge user community and enormous packages. Anybody including us can contribute a package we developed, so many statistical methods in the world may have been implemented and provided by someone. And they may have already been used and documented somewhere on the web.
- R has excellent graphical facilities, which is easy to customize.

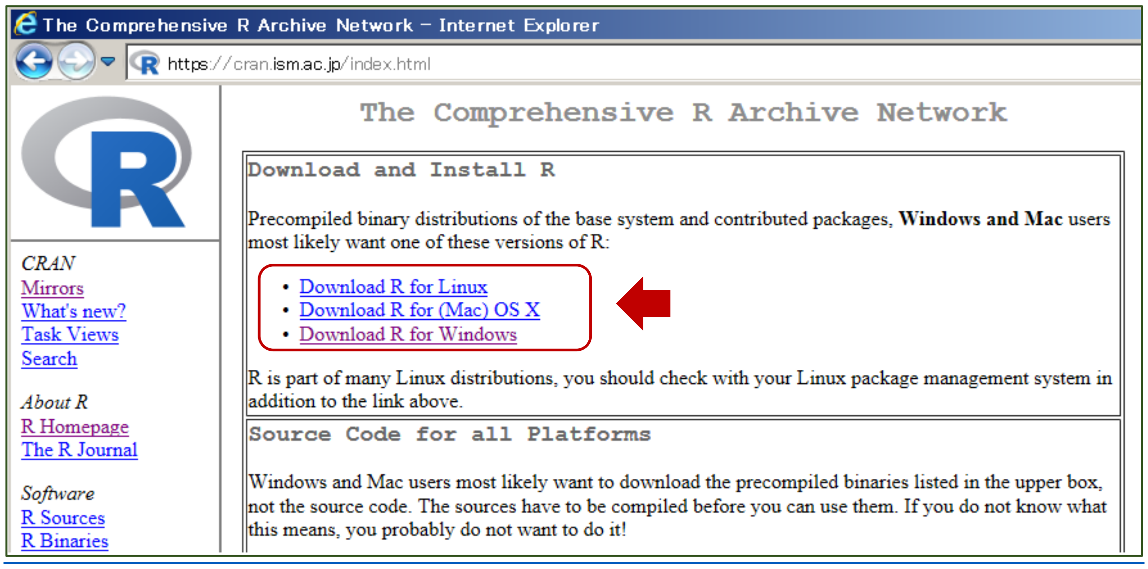
4. Installation

① R can be obtained via CRAN (The Comprehensive R Archive Network). There are many mirror sites in various locations for downloading software. Usually, it is better to choose one near you geographically to make download faster.

- CRAN https://cran.r-project.org/
- CRAN Mirrors http://cran.r-project.org/mirrors.html
- A mirror site in Japan: ISM https://cran.ism.ac.jp/index.html

② Among Linux, (Mac) OS X and Windows, choose an appropriate one for your PC to install R.

Screen 4.1:



The Comprehensive R Archive Network - Internet Explorer

https://cran.ism.ac.jp/index.html

The Comprehensive R Archive Network

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries

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.

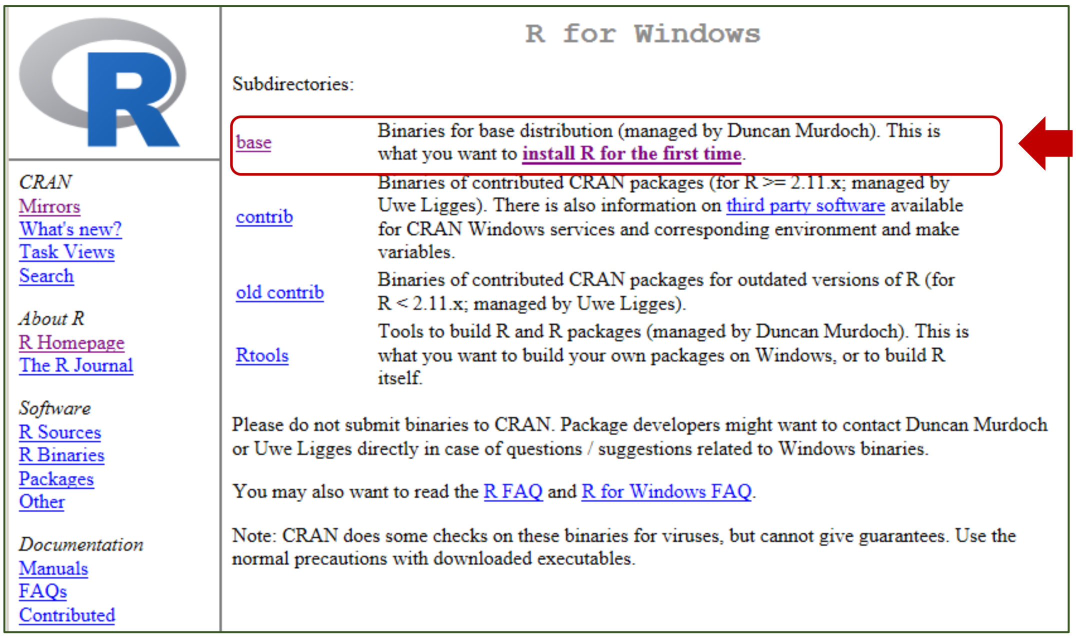
Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

③ In case of Windows, choose "Download R for Windows" in screen 4.1.

④ Then screen 4.2 is shown. Choose "base" or "install R for the first time." In either case, screen 4.3 will be displayed.

Screen 4.2:



R for Windows

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

Subdirectories:

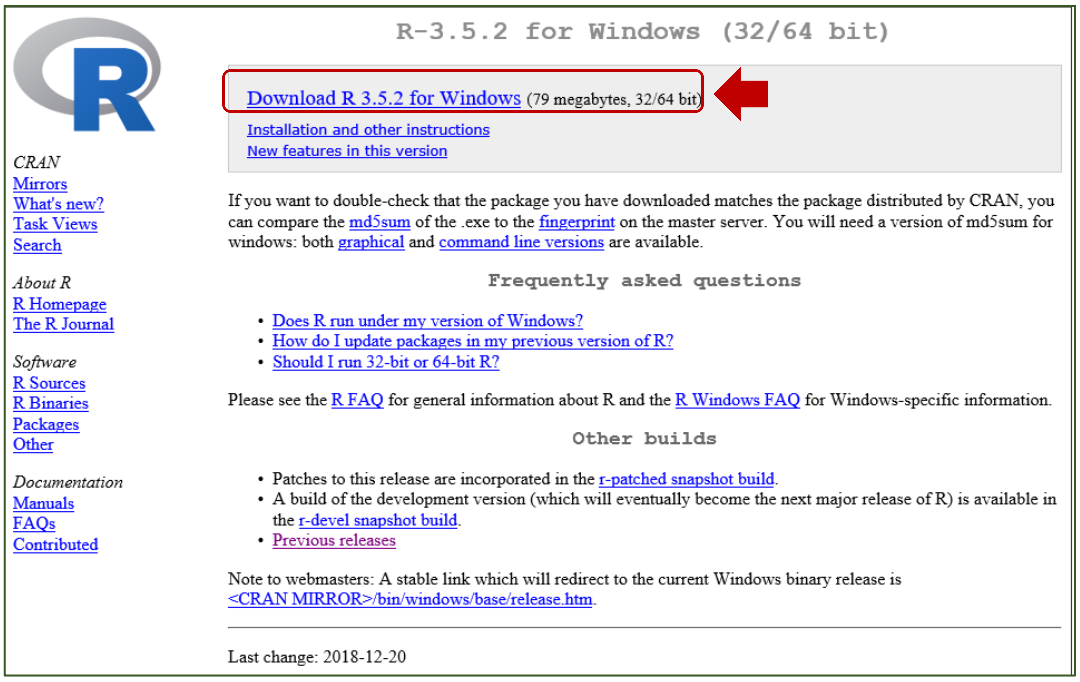
base	Binaries for base distribution (managed by Duncan Murdoch). This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 2.11.x; managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 2.11.x; managed by Uwe Ligges).
Rtools	Tools to build R and R packages (managed by Duncan Murdoch). This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Duncan Murdoch or Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the R FAQ and R for Windows FAQ.

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

Screen 4.3:


R-3.5.2 for Windows (32/64 bit)

Download R 3.5.2 for Windows (79 megabytes, 32/64 bit)
Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server. You will need a version of md5sum for windows: both graphical and command line versions are available.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Other builds

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is <CRAN MIRROR>/bin/windows/base/release.htm.

Last change: 2018-12-20

⑤ Choose "Download R X.X.X for Windows," then the setup program "R-X.X.X-win.exe" will be downloaded. Save the file in any place you like. "X.X.X" is a latest version number. If the latest version is 3.5.2, the downloaded file name will be "R-3.5.2-win.exe."

⑥ Run the setup program "R-X.X.X-win.exe" and then installation will start.

⑦ In the next screen, you can choose the language.

⑧ The setup wizard starts. Please choose default settings.

⑨ If the OS of your PC is 64bit, you can install both 32-bit files and 64-bit files in addition to the core files. If your OS is 32-bit, you cannot install 64-bit files.

⑩ After installation, you will see the R icon on your desktop.

☆ 32-bit version and 64-bitversion

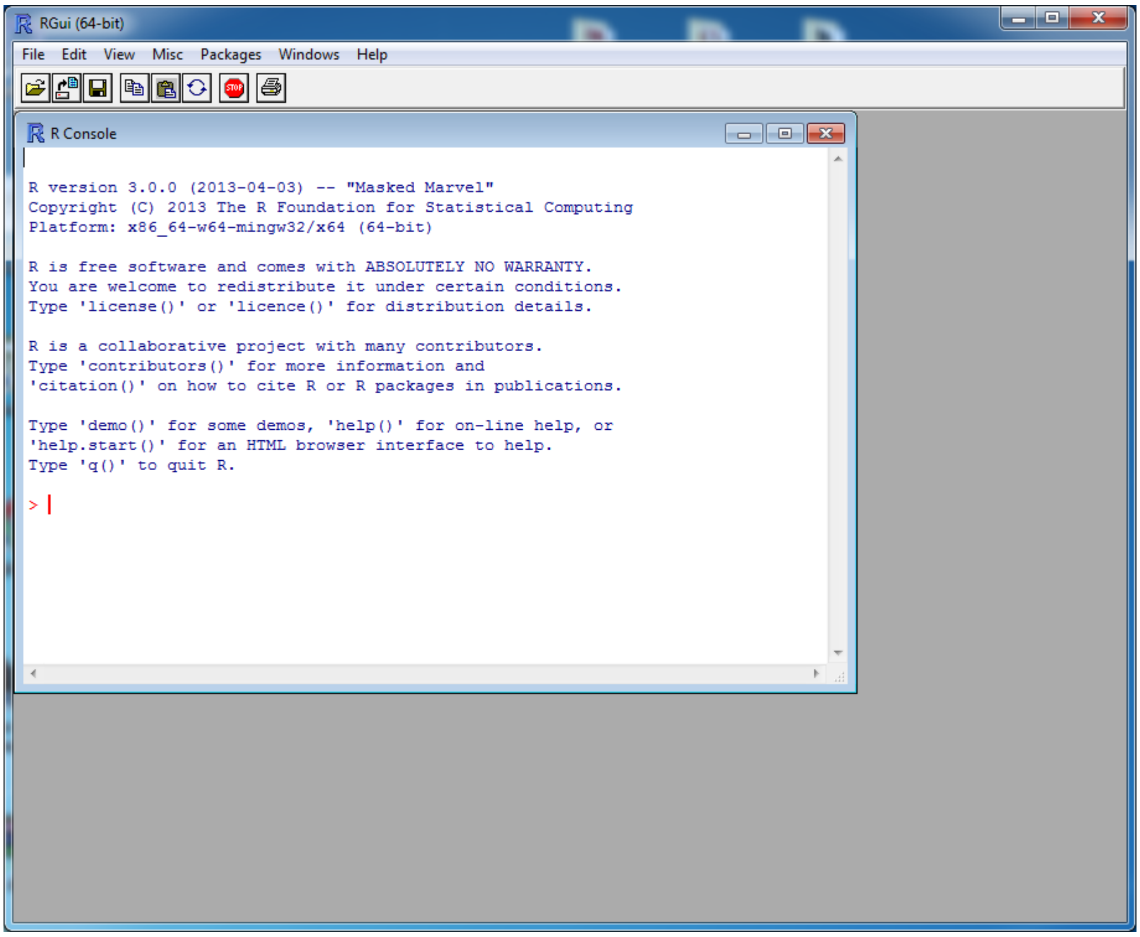
CPU	OS	R
32-bit	32-bit	32-bit
64-bit	32-bit	32-bit
	64-bit	64-bit *

* Usually 64-bit version of R is sufficient in this case. You may wish to install 32-bit version of R to use an old package (packages) for 32-bit version on your 64-bit OS.

5. Start and terminate R

- Click on the R icon, then you will see the R console.

Screen 5.1:



```
R version 3.0.0 (2013-04-03) -- "Masked Marvel"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

- The symbol ">" is called a prompt. It indicates you can type commands.
- R will be terminated if you type "q()" and press return key after the prompt

and then choose if you would like to save the workspace or not. In most cases it is not necessary, but you can save all the variables and functions you created in a file by choosing "yes."

[Exercise 1] Use R as a calculator

```
1 + 2 + 5 + 10
4 * 185
(15 + 30) * 88 / 30
sum(1:100)                    # sum of integers from 1 to 100
sqrt(10)                      # square root of 10
1+2+3; 10+987/400
```

Memo:

- Type a command after the prompt ">" and press return key to execute it.
- A hashmark "#" starts comments within the same line.
- A semi-colon ";" shows the end of a command. It enables to put plural commands in a line.
- The upward arrow key recalls previous commands.
- Escape key abandons a typing command.

6. Variables in R

Some programming languages require declaration of variable names and types before use. In case of R, variables are called objects and automatically defined according to the data stored without declaration.

If you wish to type a very long statement, you can press enter to move the next line and continue. And press escape key to cancel a command before executing it.

Basic rules of variable names:

- A combination of alphanumeric, period "." and underscore "_".
- R is case sensitive.
- Variable names should not contain blank spaces.
- A figure or underscore cannot be used as a first letter of variable names.
- The following symbols and characters cannot be used.
  ?, $, %, ^, &, *, (,), -, #, ?, , , <, >, /, |, ¥, [,] ,{,};
- Following reserved words cannot be used.
  break else FALSE for function if in Inf NA NaN next NULL repeat TRUE while
- Variable names are effectively unlimited in length.

Store data into variables

- The symbols "<-" are used to store data into variables.
- Another symbol "=" can also be used.

[Exercise 2] Make variables (vector objects)

```
a <- 10
a                                   # display the value(s) of variable a
b <- rep(1, times=5)                # repeat 1 five times
b
c <- 1:10                           # integers from 1 to 10
d <- seq(1, 10, by=2)
d                                   # [1] 1 3 5 7 9
(e <-rep(c(1, 3, 4), times=3))      # outer parentheses show values to store
(e
# If you press enter before completing a statement,
# “+” will be displayed in the next line instead of the prompt.
)                          # Then you can type the rest of the statement.
```

7. Data types

Following table shows some data types. Function *typeof* or *mode* can be used to check them. There are functions to check and covert each data types as shown in the table below.

Data type	**Check**	**Convert**
numeric (integer and double*) * Double-precision floating point number	is.numeric()	as.numeric()
integer	is.integer()	as.integer()
double	is.double()	as.double()
character	is.character()	as.character()
logical: Boolean values TRUE or FALSE	is.logical()	as.logical()

[Exercise 3] Make variables of different types

```
# integer and double
x1 <- 1;           x2 <- 6.78

typeof(x1)    # double
typeof (x2)   # double

# convert to integer
as.integer(x1)   #   The value of x1 is not replaced.
as.integer(x2)   #   The decimal point is suppressed.

x1 <- as.integer(x1)     # replace the value of x1

# character
x3 <- "a";      x4 <- "abcdefghi"
typeof(x3);     typeof(x4)              # "character"

# logical
x5 <- TRUE;     x6 <- FALSE
typeof (x5);    typeof (x6)             # "logical"
```

Data types are automatically converted if necessary during calculation. One of the most frequently occurring problems is related to the data type. For example, numerical data files contaminated with some non-numerical characters are automatically determined as character data, and R cannot convert character data to numeric, so they cannot be used for calculation. We can avoid such a problem by checking data type before calculation.

[Exercise 4] Use various data type for calculation

```
# Use variables x1 to x6 made by Exercise 3
x1 * 1.5      # OK
x2 * 1.5      # OK
x3 * 1.5      # Not Applicable
x4 * 1.5      # Not Applicable
x5 * 1.5      # OK                  # True: 1; False: 0
x6 * 1.5      # OK

x7 <- x1 * 1.5                # x1 is integer.

typeof(x7)      # x7 is automatically converted to double
```

Memo:

- Data type could be a cause of troubles in calculation.
- Logical data can be used for calculation.

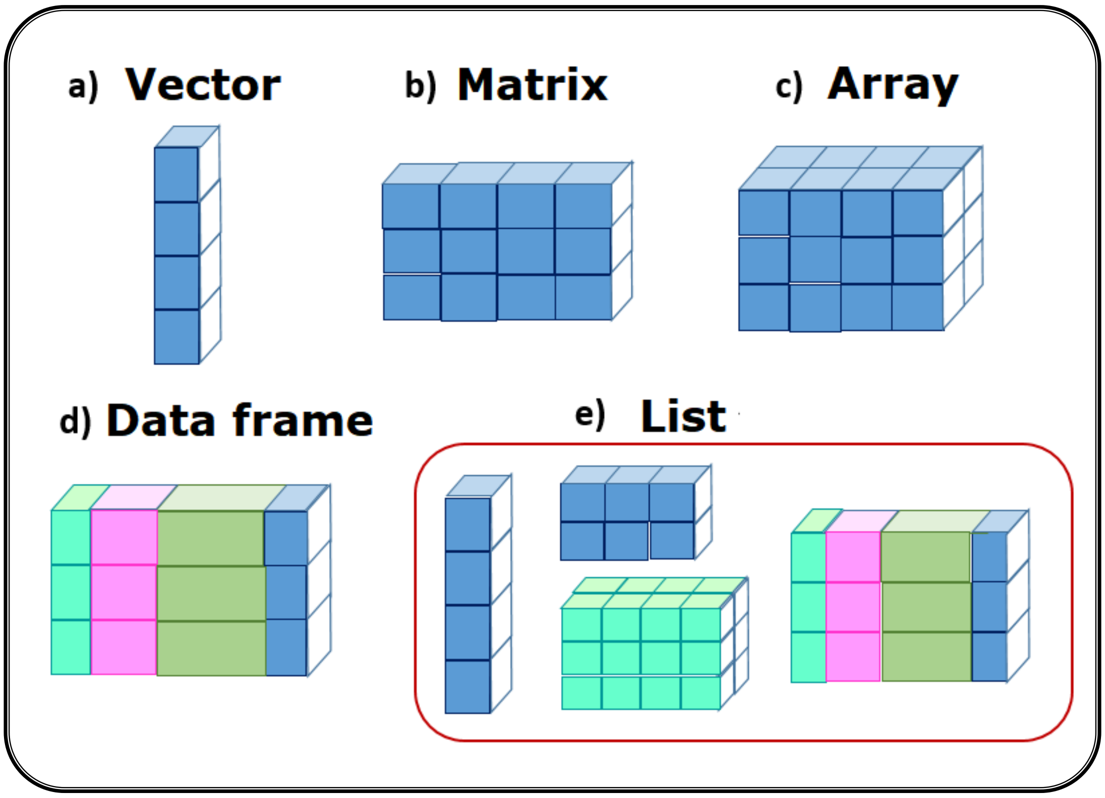
8. Data structure

Vector, matrix, array, data frame and list are major data structures. A scalar (a figure) is treated as a vector of one element in R.

Function *class* displays the data structure of a variable. The function also show the data type when the variable is a vector. There are some other functions to make, check and convert various data structures as shown in the table below.

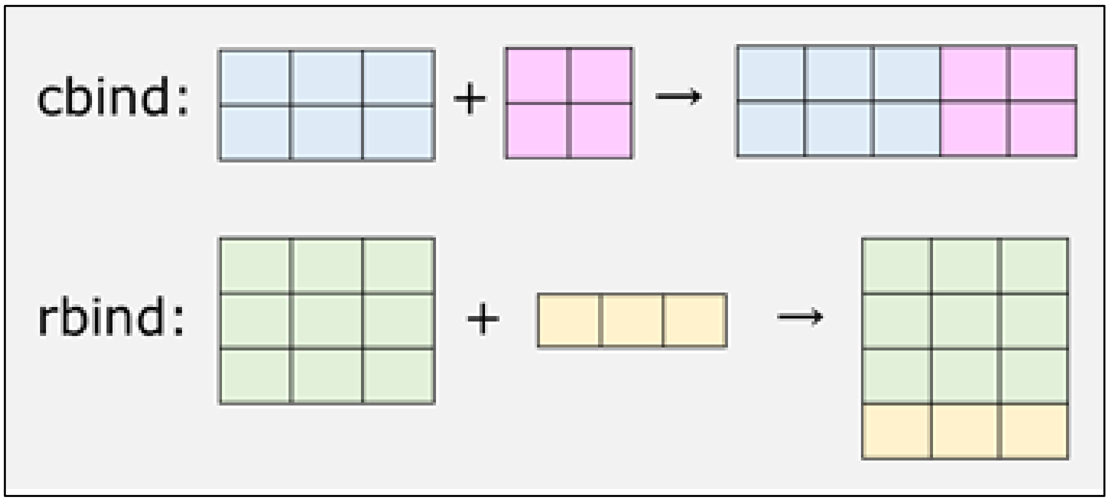
Data Structure	Make	Check	Convert
vector	c()	is.vector()	as.vector()
matrix	matrix()	is.matrix()	as.matrix ()
array	array()	is.array()	as.array()
list	list()	is.list()	as.list()
data frame	data.frame()	is.data.frame()	as.data.frame()

Figure8.1 Image of different data structures



Function *cbind* and *rbind* are provided to convine matrices and vectors.

Figure8.2 Image of function *cbind* and *rbind*



The following tables are the lists of functions for vectors and matrices.

Function for vector	**Description**
sum	Sum of all the element values
prod	Product of all the element values
max	Maximum value of the elements
min	Minimum value of the elements
length	Number of elements
sort	Sort the vector into ascending order
mean	Arithmetic mean of the elements

Function for matrix	**Description**
dim	Dimension of the matrix
t	Transpose
solve	Inverse matrix
eigen	Eigenvalues and eigenvectors
rowMeans	Calculate row means
rowSums	Calculate row sums
colMeans	Calculate column means
colSums	Calculate column sums

[Exercise 5] Make variables of different structures

```
# numeric and character vectors ----------------------------
y1 <- c(1, 2, 3)
y2 <- c("a", "bcd", "ef", "ghijkl")
is.vector(y1);   is.vector(y2)     # check their structures
y1[2]                              # show the second element
y2[c(1, 3)]                         # the first and third elements
y1[-1]                              # elements except for the first one

# matrices -------------------------------------------------
# numeric matrix
y3 <- matrix(1:9, nc=3)
y3                                 # elements are placed in vertical order
y3[2,2]                             # 2nd row of the 2nd columns
y3[3,]                              # all elements in the 3rd row
y3[,3]                              # all elements in the 3rd column
y3 <- matrix(1:9, nc=3, nr=4)
y3                                  # elements are recycled

# character matrix
(y4 <- matrix(c("a", "bc", "def", "ef", "g", "h"), nr=2))

# arrays ---------------------------------------------------
(y5 <- array(1:24, c(2,3,4)))
y5[1,1,1]

# list -----------------------------------------------------
(y6 <- list(y1, y2, y3, y4, y5))
y6[[2]][2]
names(y6) <- c("y1", "y2", "y3", "y4", "y5")
                        # components of a list can be named
y6
y6$y5[1,2,1]
str(y6)                 # show the structure of the list

# cbind, rbind ---------------------------------------------
(z1 <- c(1,2,3))                    # vector with 3 elements
(z2 <- matrix(1:12, nc=3, nr=4))    # 4×3 matrix
(z3 <- rbind(z1, z2))               # 5×3 matrix
(z4 <- c(1,2,3,4,5))                # vector with 5 elements
(z5 <- cbind(z3, z4))               # 5×4 matrix
z5[2,4]                             # 2
```

9. Factors

A factor is a vector object used to store a categorical variable. R provides both ordered and unordered factors. A factor has the following characteristics.

- A factor looks like characters, but they are displayed without quotation. They are in fact treated like numerical values, but cannot be used for calculation.
- A factor object is consists of plural levels.
- A factor object can be checked and transformed using *is.factor* and *as.factor* respectively.

[Exercise 6] Make factor objects

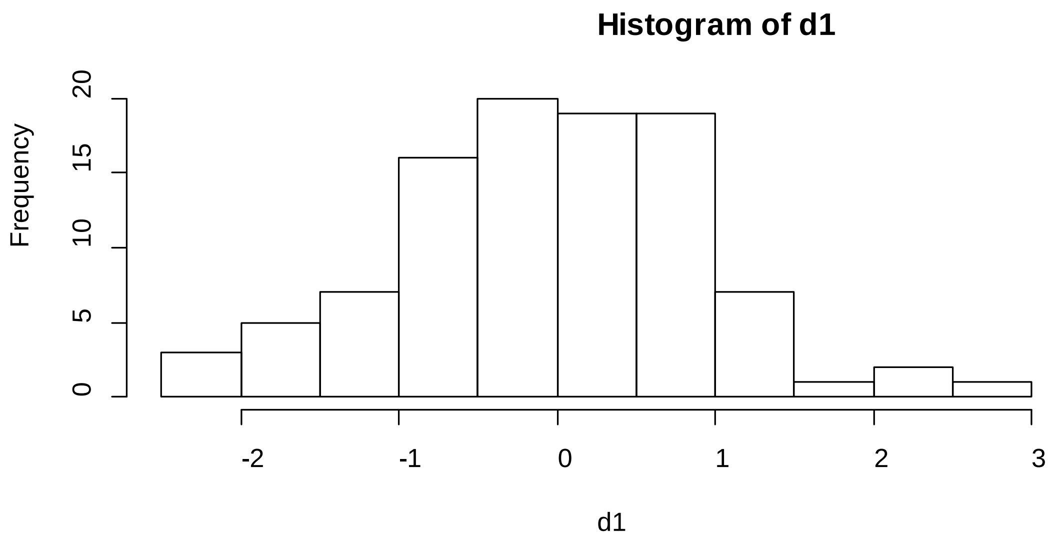
```
y <-factor(c("A", "AB", "A", "A", "B", "O", "O"))
y
typeof(y)                       # data type (integer)
class(y)
str(y)                          # structure of the object
(as.character(levels(y))[y])    # convert : "A" "AB" "A" "A" "B" "O" "O"
y1 <- factor(c("01", "98", "12"))   #   create a numeric factor
(y2 <- as.numeric(levels(y1))[y1]) # convert
# [1] 1 98 12
```

10. Basic visualization

10. 1 Histograms

Following code is for making 100 observations following the standard normal distribution, and plot the data in a simple histogram. The help file of the function inform you how to customize the plot.

```
# make 100 observation following the standard normal distribution
set.seed(106)    # using a seed, we can get the same random numbers
d1 <- rnorm(100)
hist(d1)          # basic histgram
help(hist)        # open the help file
```

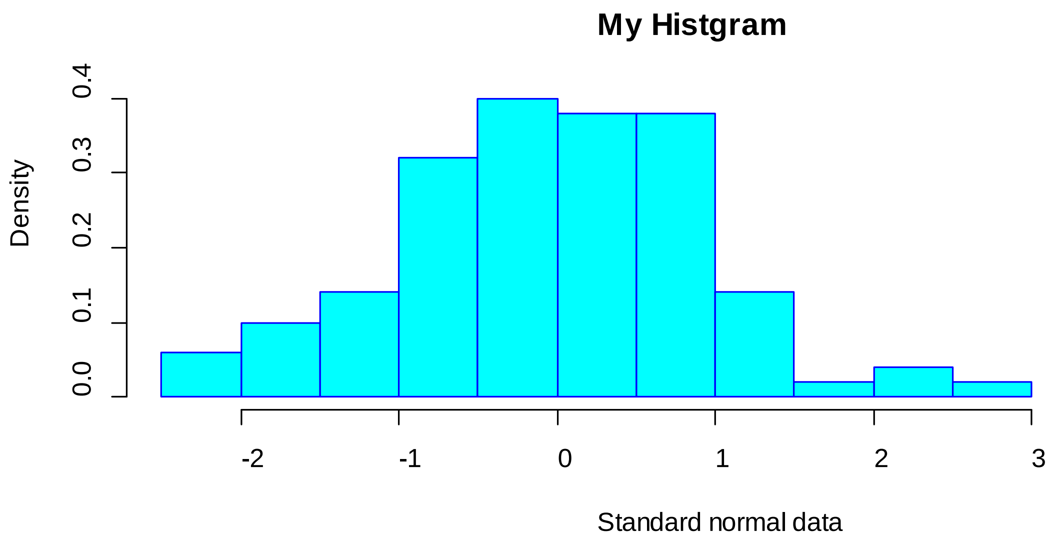


```
# Give a different title
hist(d1, main= "My Histgram")

# Change the break
hist(d1, main= "My Histgram", breaks="Scott")

# Make it a density plot
hist(d1, freq=FALSE, main= "My Histgram", xlab= "Standard normal data")

# Paint the bars
hist(d1, freq=FALSE, main= "My Histgram", xlab= "Standard normal data",
    col="cyan", border="blue")
```



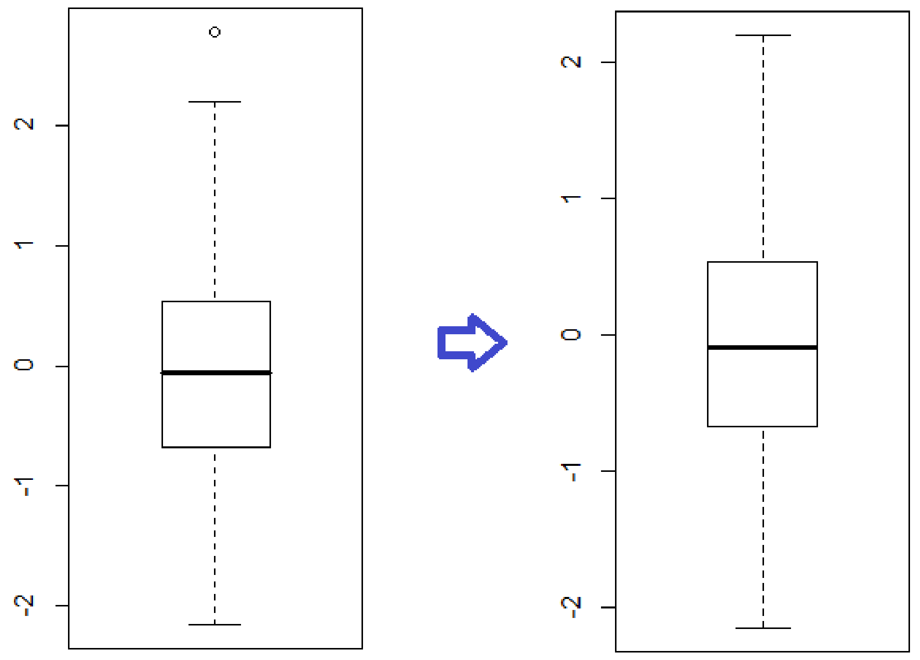
10. 2 Box plots

Box plots are useful to detect outliers. In these observation, there is an extreme value.

```
# set.seed(106)
# d1 <- rnorm(100)
boxplot(d1)          # basic histgram

# Fundamental statistics
summary(d1)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-2.14703 -0.67224 -0.04909 -0.03313  0.54544  2.77503

# Find the biggest observation
max(d1)              # [1] 2.775035
which(d1 == max(d1))
# [1] 84     # observation number
d1[84]        # [1] 2.775035
# plot d1 without the extreme value
boxplot(d1[-84])
```



10. 3 Scatter plots

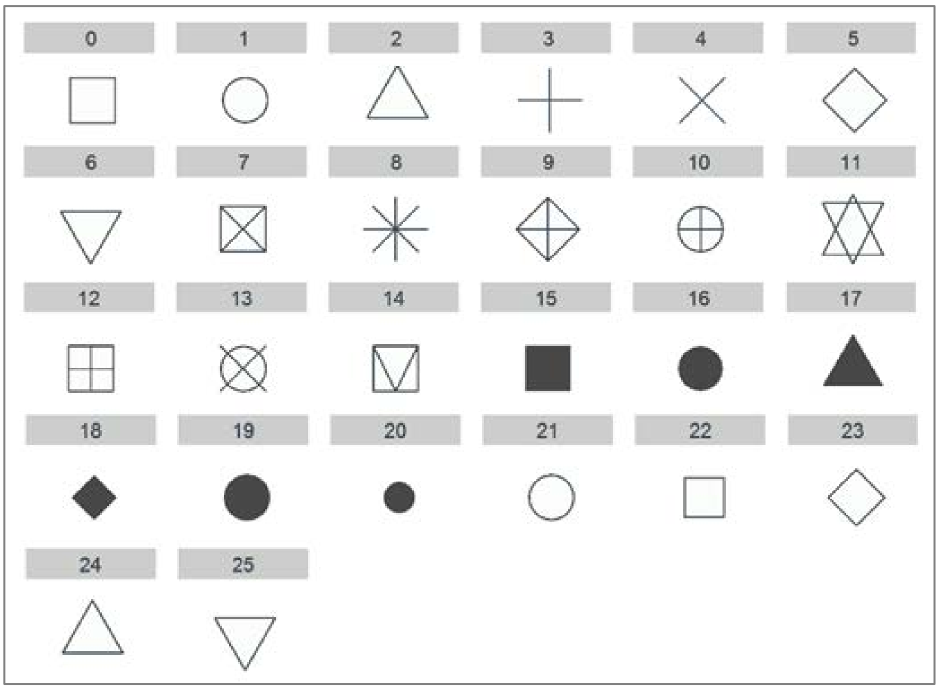
There are many built-in datasets in R. We can see the list using the command "data()". If we would like to load a specific dataset, indicate the dataset name like "data(dataset name)" to load it.

In this section, we use *cars* dataset which gives the speed of cars and the distances taken to stop.

```
data(cars)          # load the dataset
plot(cars)
help(plot)          # Useful information for customization

# Change plot symbols and the size
plot(cars, pch=8, col="magenta")
plot(cars, pch=16, col="magenta")
plot(cars, pch=16, col="magenta", cex=2)
plot(cars, pch=21, col="gray", cex=2, bg="magenta")
```

List of plot symbols



10. 4 Scatter plot matrices

Scatter plots are for two variables. Scatter plot matrices are used to grasp the image of multivariate datasets. In this section, we use another built-in dataset called iris.

The iris data set is about the measurements in centimeters of 150 flowers in 3 different species. The dataset contains 4 quantitative variables (sepal length, sepal width, petal length and petal width) and a categorical variable about species.

After loading the iris dataset, we make a scatterplot matrix of the four quantitative variables.

```
data(iris)                # call a dataset about iris flowers
help(iris)
iris                      # display the iris dataset
pairs(iris[, 1:4])        # display a scatterplot matrix
```

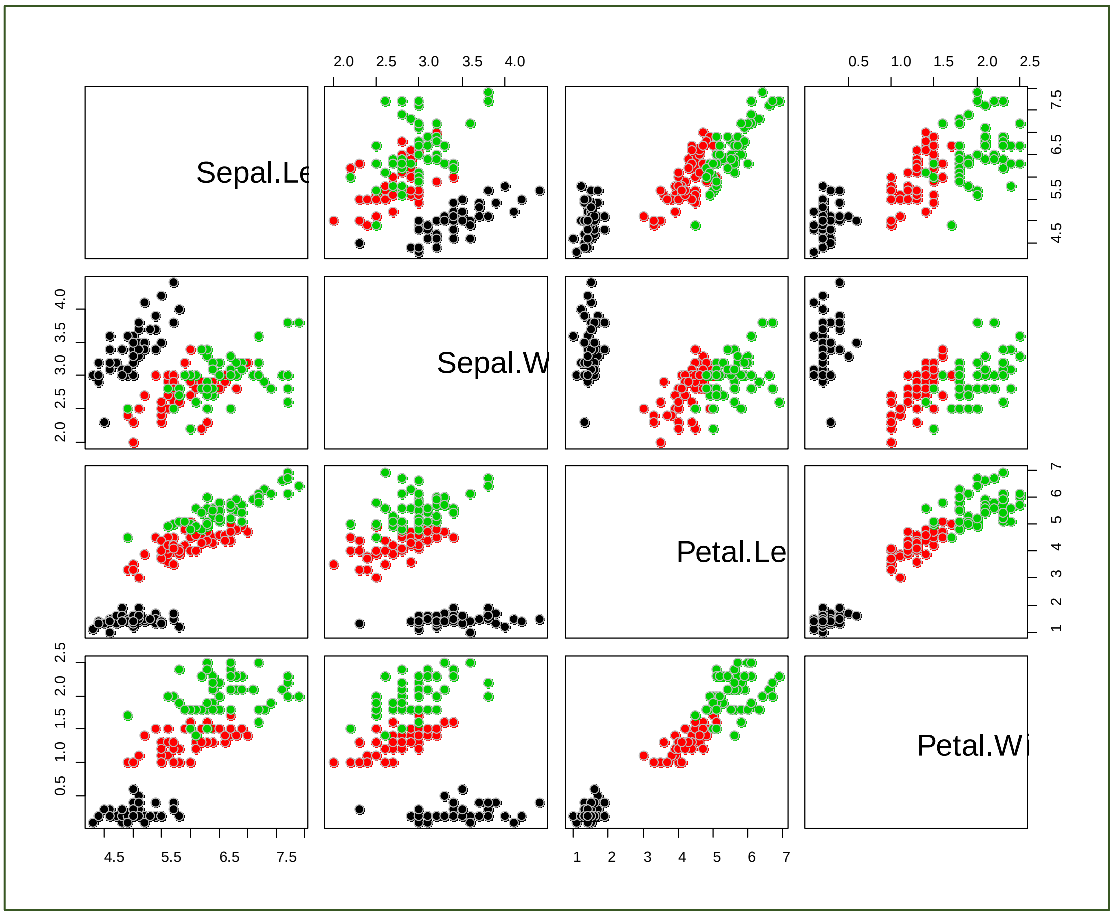
Next, give the observations different colors by species.

```
  # make indicies by species
ix1 <- which(iris[,5] == "setosa")       # data numbers of setosa
ix2 <- which(iris[,5] == "versicolor")   # data numbers of versicolor
ix3 <- which(iris[,5] == "virginica")    # data numbers of virginica

  # prepare a color index
f1 <- rep(NA, dim(iris)[1])     # 150 elements
f1[ix1] <- 1;      f1[ix2] <- 2;      f1[ix3] <- 3;
pairs(iris[,1:4], pch=21, bg=f1, col="gray", cex=1.5)
```

There are a few different way to indicate colors. One example is using color names such as "red", "blue" and "yellow." You can see the available color names by "colors()". Another way is using numbers as follows.

Number	Color	Number	Color
1	Black	6	Purple
2	Red	7	Yellow
3	Green	8	Gray
4	Blue	9	Black
5	cyan	10	Red



The following site gives you some additional information.

- ➢ Information about some graphic parameters
  https://www.statmethods.net/advgraphs/parameters.html
- ➢ R color cheatsheet
  https://www.nceas.ucsb.edu/~frazier/RSpatialGuides/colorPaletteCheatsheet.pdf
- ➢ R color styles
  http://bioinfo.umassmed.edu/bootstrappers/bootstrappers-courses/pastCourses/rCourse_2016-04/Additional_Resources/Rcolorstyle.html

11. Files and directories manipulation

The following functions are for basic manipulation of files and directories.

```
#open a DOS window
shell("start")

# open file explorer
shell("explorer")

#change directory
setwd("mydirectory")

#make directory
dir.create("mydirectory")

#delete file
file.remove("myfile")
```

It is also possible to give DOS commands from R using function shell.

```
# show current directory
getwd()

#list files in the directory
list.files()

#select pdf files among them
F1 <- list.files()
(F2 <- F1[grep(".pdf", F1)])

# another way to select pdf files in the current directory
list.files(path=getwd(), pattern="*.pdf")

#get the file sizes and order the files by the size
F2.size   <- file.info(F2)$size
F2.ix <- order(F2.size)
F3 <- F2[F2.ix]
```

11. Packages

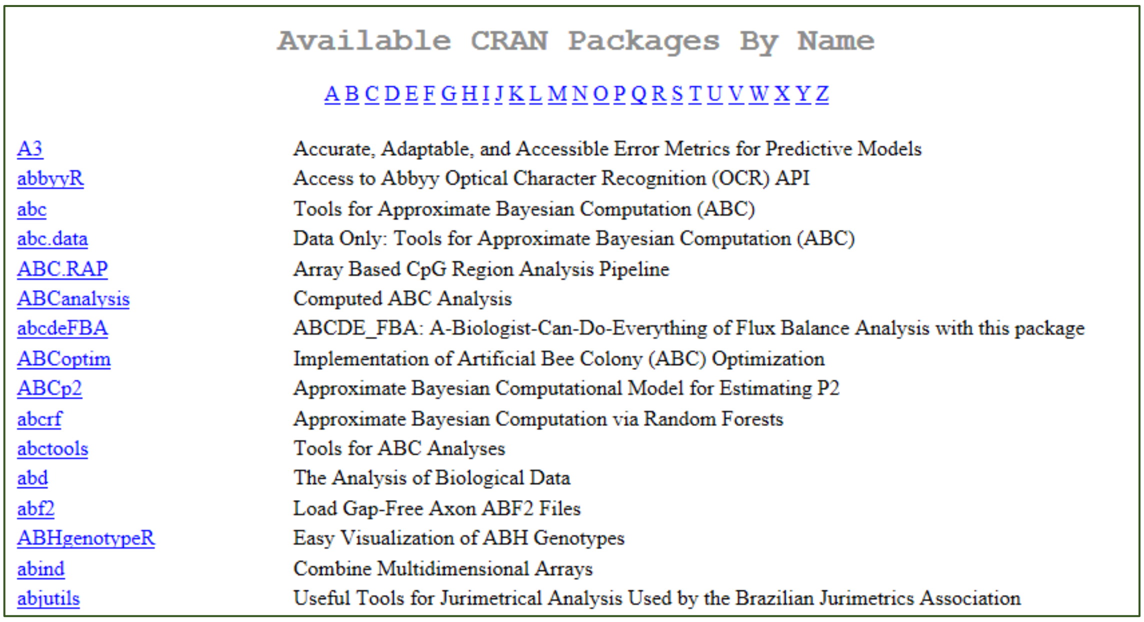
11. 1 What is a package?

All R functions and datasets are stored in *packages*. Some basic packages are installed with R and automatically loaded when you start an R session. Others are available only when stored packages are loaded. Followings are some features of a package.

- A user can freely create and use one's own packages. It is also possible to distribute them by one's own for other users. They are called contributed packages.
- CRAN provides thousands of contributed packages. Some implements statistical methods for a specific purpose, some other may give access to data or sometimes designed to complement textbooks.
- The function "library()" shows the list of currently installed packages on the PC.

Available CRAN packages are shown by name at [https://cran.r-project.org/web/packages/available_packages_by_name.html]. Screen 11.1 shows the image. If you click on a name of the packages in the left side of the page, screen 11.2 is displayed. A manual of the package and binaries for install are provided with some description about the package in addition of the information about dependency.

Screen 11.1 Available CRAN packages by name

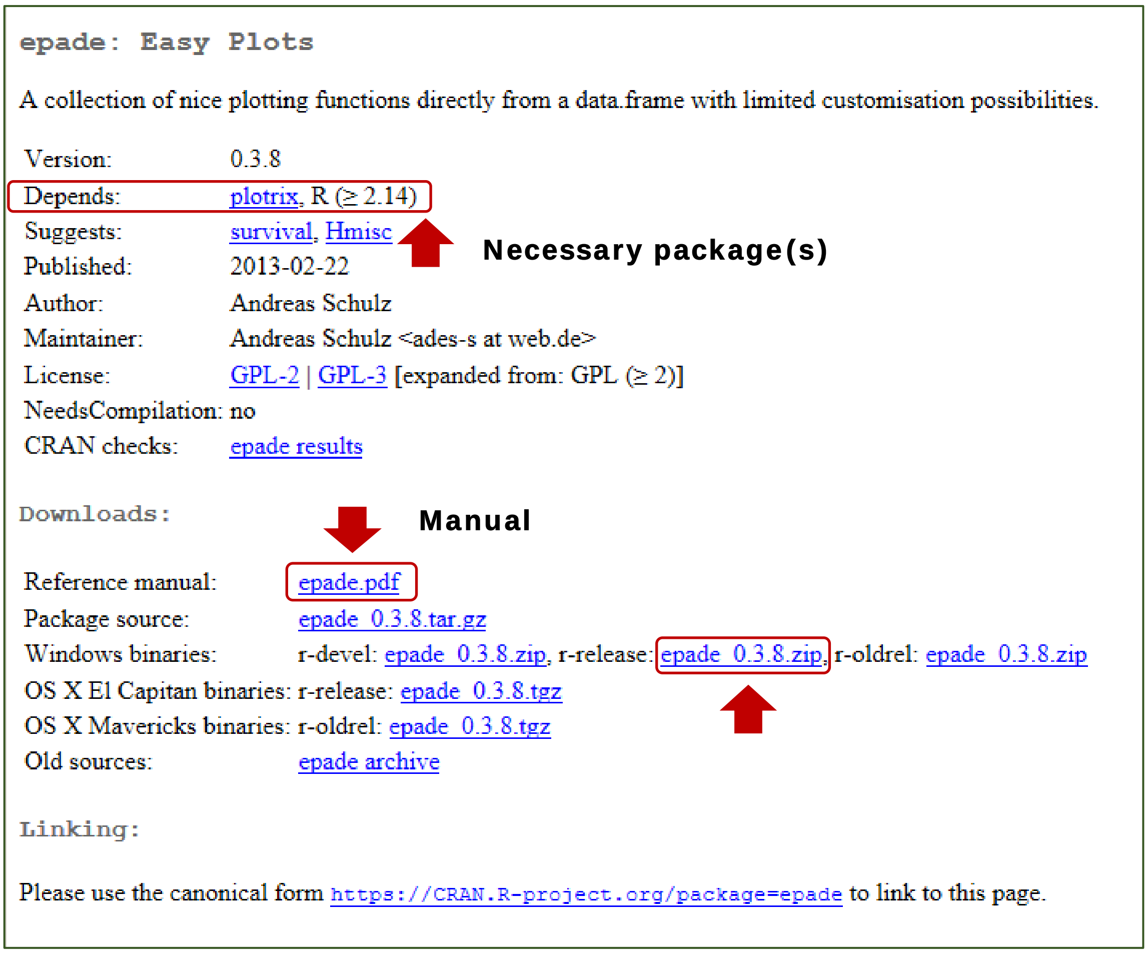


Available CRAN Packages By Name

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A3	Accurate, Adaptable, and Accessible Error Metrics for Predictive Models
abbyyR	Access to Abbyy Optical Character Recognition (OCR) API
abc	Tools for Approximate Bayesian Computation (ABC)
abc.data	Data Only: Tools for Approximate Bayesian Computation (ABC)
ABC.RAP	Array Based CpG Region Analysis Pipeline
ABCanalysis	Computed ABC Analysis
abcdeFBA	ABCDE_FBA: A-Biologist-Can-Do-Everything of Flux Balance Analysis with this package
ABCoptim	Implementation of Artificial Bee Colony (ABC) Optimization
ABCp2	Approximate Bayesian Computational Model for Estimating P2
abcrf	Approximate Bayesian Computation via Random Forests
abctools	Tools for ABC Analyses
abd	The Analysis of Biological Data
abf2	Load Gap-Free Axon ABF2 Files
ABHgenotypeR	Easy Visualization of ABH Genotypes
abind	Combine Multidimensional Arrays
abjutils	Useful Tools for Jurimetrical Analysis Used by the Brazilian Jurimetrics Association

Screen 11.2


epade: Easy Plots

A collection of nice plotting functions directly from a data.frame with limited customisation possibilities.

Version:	0.3.8
Depends:	plotrix, R (≥ 2.14)
Suggests:	survival, Hmisc
Published:	2013-02-22
Author:	Andreas Schulz
Maintainer:	Andreas Schulz <ades-s at web.de>
License:	GPL-2 | GPL-3 [expanded from: GPL (≥ 2)]
NeedsCompilation:	no
CRAN checks:	epade results

Downloads:

Reference manual:	epade.pdf
Package source:	epade_0.3.8.tar.gz
Windows binaries:	r-devel: epade_0.3.8.zip, r-release: epade_0.3.8.zip, r-oldrel: epade_0.3.8.zip
OS X El Capitan binaries:	r-release: epade_0.3.8.tgz
OS X Mavericks binaries:	r-oldrel: epade_0.3.8.tgz
Old sources:	epade archive

Linking:

Please use the canonical form https://CRAN.R-project.org/package=epade to link to this page.

11. 2 Installation

■ Installation of a package from menu with the Internet

Select "packages" from the menu bar.

=> Select "Install package(s) ...

=> Choose a mirror close to you geographically

=> Select necessary package(s)

* Dependent packages are automatically installed together with the target package(s).

* It is possible to select more than one packages at once.

■ Installation of a package by a command with the Internet

Type the command 'install.packages("package name")' after the prompt. It is also possible to indicate a mirror site like install.packages("CHAID", repos="http://xxx.xxx.xxx")'. The package including dependent ones are installed all together.

■ Installation of a package using a zip file

This is for installing packages to the PCs without the Internet connection.

Choose a necessary package at “Available CRAN packages are shown by name” [https://cran.r-project.org/web/packages/available_packages_by_name.html]. This step needs the Internet connection.

=> Download the windows binary (the zip file of "r-release")

If you need more than packages, or your target package has some dependent packages, download all the necessary zip files. Then copy all the zip files to the target PC to be installed. This step does not need the Internet.

=> Select "packages" from the menu bar.

=> Select "Install package(s) from local zip files. "

=> Indicate the zip files to be installed.

Instead of using menu bar, we can install all the zip files in a directory as follows.

```
setwd("the directory name with zip files")
p.list <- list.files(path=getwd(), pattern="*.zip")
lapply(p.list, install.packages, repos=NULL)
```

11. 3 Usage

Installation of a package is required once at the beginning. Loading package(s) are needed in each session before use by function "require(package name)" or "library(package name)".

Additional information about installing packages:
https://www.r-bloggers.com/how-to-install-packages-on-r-screenshots/

[Exercise 7] Use *epade* package

In this exercise, we make a 3d histgram of the petal width from the iris dataset using package *epade*.

```
# data(iris)         # it is not necessary if you've already load the dataset

hist(iris[,4])       #   making a histogram using the 4th variable

#   install.packages("epade")
require(epade)                   #   load the package

# 3D histogram (1)
bar3d.ade(iris[,5], iris[,4])   #   plot the 4th variables by species in default settings

# 3D histogram (2)
bar3d.ade(iris[,5], iris[,4], lcol="white", alpha=0.2)
 # improved the appearance.
 # lcol: the bar edge color;     alpha: transparency,     See the help.
help(bar3d.ade)                                        # display help

# 3D histogram (3)
bar3d.ade(iris[,5], iris[,4], lcol="white", wall=1, alpha=0.2,
  xlab=colnames(iris)[5], zlab=colnames(iris)[4])
 # wall=1: gray,   xlab and zlab:   titles of the x axis and z axis

# 3D histogram (4)   improve the title of z axis
bar3d.ade(iris[,5], iris[,4], lcol="white", wall=1, alpha=0.2,
    xlab=colnames(iris)[5], zlab="")   # remove title of z axis
text(locator(1), colnames(iris)[4])      # add the title of z axis to an arbitral location
```

■ About the iris dataset

This is one of the most famous dataset which gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

Record No.	**Sepal.Length**	**Sepal.Width**	**Petal.Length**	**Petal.Width**	**Species**
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
…	…	…	…	…	…
50	5.0	3.3	1.4	0.2	setosa
51	7.0	3.2	4.7	1.4	versicolor
52	6.4	3.2	4.5	1.5	versicolor
…	…	…	…	…	…
100	5.7	2.8	4.1	1.3	versicolor
101	6.3	3.3	6.0	2.5	virginica
102	5.8	2.7	5.1	1.9	virginica
…	…	…	…	…	…
150	5.9	3.0	5.1	1.8	virginica

Anderson, Edgar (1935). The irises of the Gaspe Peninsula, *Bulletin of the American Iris Society*, **59**, 2–5.

Fisher, R. A. (1936) The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7**, Part II, 179–188.