

Introduction to R

January 21, 2020

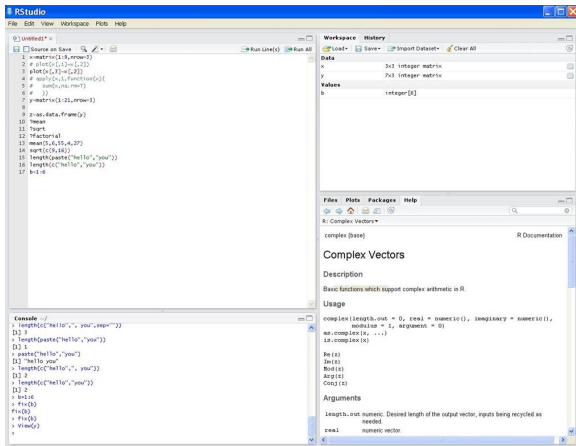
What is R?

- R is an integrated suite of software facilities for **data manipulation, simulation, calculation** and **graphical display**
 - Handles and analyzes data very effectively
 - Graphical capabilities for very sophisticated graphs and data displays
 - Preferred by statistics community
 - Weakness
 - Slower than other programming languages such as Perl, Java, C++)
 - Can be memory intensive

Rstudio

RStudio is a convenient interface and allows the user to run R in a more user-friendly environment

- <http://www.rstudio.com/products/rstudio/download/>



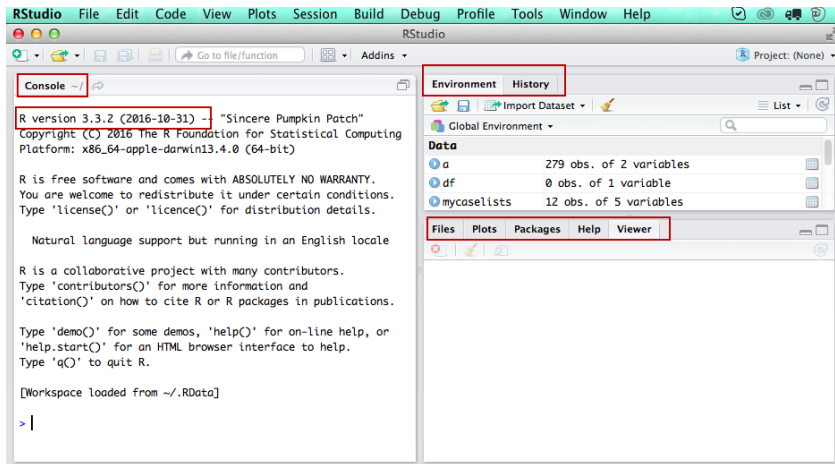
R Studio has with 4 Windows on the main screen

- Code: for coding
- Console: coding calculations and display results
- Work Space: list of objects and calculations that are created
- Five tab window
 - Files: stored and imported files
 - Plot: displayed area, graph view selection, export option
 - Packages: list of packages installed on system, option to search and install other packages
 - Help: more information about functions, arguments, user manuals, etc
 - Viewer: used to view local web content such as widgets, rCharts, and applications

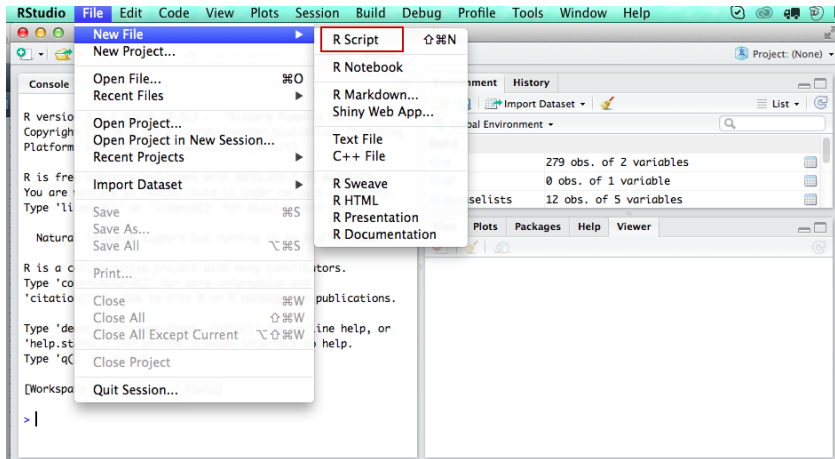
RStudio windows

The image shows the RStudio interface with four windows highlighted by blue rounded rectangles:

- Code**: The main editor window showing R code for data manipulation and plotting.
- Console**: The bottom-left window showing the execution of the code, including the command `boxplot(x,b,col = c("red","green"))`.
- Workspace**: The top-right window showing the environment with variables like `a`, `b`, `code`, `codeID`, `colmedian`, `datacfrow`, `f`, `genetymbo1`, `s`, `IndexActive`, and `IndexInactive`.
- Five tab window**: A window on the right showing a boxplot with two groups, labeled 1 and 2. Group 1 has a red box and Group 2 has a green box. The y-axis ranges from 0 to 80.



RStudio: create a new R script



RStudio: run R commands

The screenshot displays the RStudio interface with the following components:

- Script Editor:** Contains an R script with the following code:

```
12 # Any text after # is not evaluated.
13
14 ##### Lesson 1. Histogram
15 #####
16
17
18 #import a library named RColorBrewer
19 library(RColorBrewer)
20
21 #import a data set named VADeaths, you can type VADeaths to see the detail
22 data(VADeaths)
23
24 #set graphical parameters, two rows and 3 graphical in each row
25 par(mfrow=c(1,1))
26
27 #draw Histogram
28 # breaks : the number of cells for the histogram
29 # col : color of the bars in the graph
30 # main : title of the graph
31 hist(VADeaths,breaks=12, col=brewer.pal(3,"Set3"),main="Set3 3 colors")
32 hist(VADeaths,breaks=4, col=brewer.pal(3,"Set2"),main="Set2 3 colors")
33
```

A red box highlights the `Run` button in the top toolbar and the `hist` function calls on lines 31 and 32 of the script.
- Environment Pane:** Shows the loaded objects:
 - `a`: 279 obs. of 2 variables
 - `df`: 0 obs. of 1 variable
 - `mycaselists`: 12 obs. of 5 variables
 - `myclinicaldata`: 279 obs. of 4 variables
 - `mygeneticprofile`: 7 obs. of 6 variables
- Values Pane:** Shows the values of the objects:
 - `myconcerstudy`: "hnscc_tcga_pub"
 - `mycaselist`: "hnscc_tcga_pub_3way_complete"
 - `mycncls`: Classes 'CGDS', 'Object', atomic [1:17 NA]
- Console:** Displays the R startup message:

```
you are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

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.

[Workspace loaded from ~/.RData]
```


RStudio: run a R command

Environment History

Global Environment

Data

- a: 279 obs. of 2 variables
- df: 0 obs. of 1 variable
- mycaselists: 12 obs. of 5 variables
- myclinicaldata: 279 obs. of 4 variables
- mygeneticprofile: 7 obs. of 6 variables
- VADeaths: num [1:5, 1:4] 11.7 18.1 26.9 41 66 8.7 11.7 20.3 30.9 54...

Values

- mycancerstudy: "hnscc_tcga_pub"
- mycaselist: "hnscc_tcga_pub_hmv_complete"

Files Plots Packages Help Viewer

Zoom Export Publish

```
13 #####
14 ##### Lesson 1. Histogram
15 #####
16 #####
17
18 #import a library named RColorBrewer
19 library(RColorBrewer)
20
21 #import a data set named VADeaths, you can type VADeaths to see the detail
22 data(VADeaths)
23
24 #set graphical parameters, two rows and 3 graphical in each row
25 par(mfrow=c(1,1))
26
27 #draw Histogram
28 # breaks : the number of cells for the histogram
29 # col : color of the bars in the graph
30 # main : title of the graph
31 hist(VADeaths,breaks=12, col=brewer.pal(3,"Set3"),main="Set3 3 colors")
32 hist(VADeaths,breaks=4, col=brewer.pal(3,"Set2"),main="Set2 3 colors")
33 hist(VADeaths,breaks=8, col=brewer.pal(3,"Set1"),main="Set1 3 colors")
34
```

Console

~/Mary_Yang/Workshop/

help.start() for an online browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

```
> plot(c(1,2),c(2,3))
> setwd("~/Mary_Yang/Workshop")
> library(RColorBrewer)
>
> #import a data set named VADeaths, you can type VADeaths to see the detail
> data(VADeaths)
>
> #set graphical parameters, two rows and 3 graphical in each row
> par(mfrow=c(1,1))
> hist(VADeaths,breaks=12, col=brewer.pal(3,"Set3"),main="Set3 3 colors")
>
```

Set3 3 colors

Frequency

VADeaths

VADeaths Bin	Frequency
10-12.5	2.0
12.5-15	3.0
15-17.5	3.0
17.5-20	2.0
20-22.5	1.0
22.5-25	1.0
25-27.5	1.0
27.5-30	2.0
30-32.5	1.0
32.5-35	1.0
35-37.5	2.0
37.5-40	1.0
40-42.5	1.0
42.5-45	2.0
45-47.5	0.0
47.5-50	0.0
50-52.5	1.0
52.5-55	1.0

- R is a **case-sensitive**, interpreted language
 - For example, a and A are two different objects.
- Two ways to run R command
 - Enter commands into the R console window at the command prompt (`>`)
 - Create R-scripts in an editor and save them in a file (filename.R) for later re-use.

Getting Started with R

- Data management
 - Data importing.
 - Write data into files.
 - Save data as an image.
- Data process
 - Types of data.
 - Arithmetic of data.
- Functions
- Packages

- Data can be entered from the console.
- Larger data often be read as values from external files rather than entered at the keyboard.
- Different data formats that can be imported into R and different functions to call them.
 - .csv
 - .txt
 - HTML table
 - Excel

- CSV

```
df = read.csv(file_name.csv)
df = read.csv2(file_name.csv)
```

- TXT

```
df = read.table(file_name.txt)
```

Building-in data set in R

- R also contains many datasets that are built-in to the software
- These datasets are stored as data frames. To see the list of datasets, use

```
> data()
```

- Then, a window will open and the available datasets are listed

Building-in data set in R : an example

Example: to open the dataset called *Orange*

```
> data (Orange)
```

After doing so, the data frame *Orange* is now in your workspace.

```
# To learn more about this data, type
```

```
> ?Orange
```

```
> Orange
```

```
  Tree age circumference
```

```
1 1 118 30
```

```
2 1 484 58
```

```
3 1 664 87
```

```
4 1 1004 115
```

```
5 1 1231 120
```

```
6 1 1372 142
```

```
7 1 1582 145
```

Data Export

- CSV:

```
write.csv(data, file = "path/file_name.csv")
```

- TXT

```
write.table(data, file = "path/file_name.txt", row.names =  
  FALSE, col.names = TRUE)
```

- The argument **row.names = FALSE** makes that no row names are written to the file. Because nothing is specified about col.names, the default option **col.names = TRUE** is chosen and column names are written to the file.

More options

```
help (write.csv)
?write.csv
?write.table
```

The screenshot shows the RStudio interface. The editor window on the left contains R code for a histogram. The console window at the bottom left shows the command `?write.table` entered. The right-hand pane displays the R Documentation for `write.table (utils)`. The `Help` tab is selected in the top navigation bar. The documentation includes a description of the function and its usage, with the function signature `write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ", eol = "\n", na = "NA", dec = ".", row.names = TRUE, col.names = TRUE, qmethod = c("escape", "double"), fileEncoding = "")` highlighted in a red box.

```
12 # Any text after # is not evaluated.
13
14 #####
15 #### Lesson 1. Histogram
16 #####
17
18 #import a library named RColorBrewer
19 url |
20 hist(VADeaths,breaks=4 ,col=brewer.pal(3,"Set2"),main="Set2 3 colors")
21
19:5 (Untitled) R Script
```

Environment History

Files Plots Packages **Help** Viewer

R: Data Output Find in Topic

write.table (utils) R Documentation

Data Output

Description

write.table prints its required argument x (after converting it to a data frame if it is not one nor a matrix) to a file or [connection](#).

Usage

```
write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",
            eol = "\n", na = "NA", dec = ".", row.names = TRUE,
            col.names = TRUE, qmethod = c("escape", "double"),
            fileEncoding = "")
```

write.csv(...)

Saving Data

Save the data into image for reuse by R

```
save(data, file = "data.RData")  
load("data.RData")
```

Data process: computing with R

- When variables are used, they need to be **initialized** with numbers.
- The assignment operator is “ \leftarrow ”. Alternatively, as of R version 1.4.0, you can use “=” as the assignment operator.

```
A = 5
```

```
A <- 5
```

Data Types

Types	Examples
Integer: Natural numbers	1, 2, 3
Numeric: Decimal values	1.5, 2.2, 3.7
Logical: Boolean values	TRUE or FALSE (T or F)
Character: Text or string values	"cat" "blue"

Data Structures

- R has a wide variety of data structures:
 - vector
 - matrix
 - data frame
 - list

	Homogeneous	Heterogeneous
1d	Vector	List
2d	Matrix	Data frame
nd	Array	

R function: Control structure

- R includes the usual control-flow statements (like conditional execution and looping) found in most programming languages. These include (the syntax can be found in the help file accessed by **?Control**):

```
if(cond) expr
if(cond) cons.expr else alt.expr

for(var in seq) expr
while(cond) expr
repeat expr
break
next
```

Logical operators

- Many of these statements require the evaluation of a logical statement, and these can be expressed using logical operators:

Operator	Meaning
==	Equal to
!=	Not equal to
<, <=	Less than, less than or equal to
>, >=	Greater than, greater than or equal to
&	Logical AND
	Logical OR

User-defined function

- R users can define their own function. The general format for creating a function is

```
functionName <- function(arg1, arg2, ...) { R code }
```

- In the above, **functionName** is any allowable object name and **arg1**, **arg2**, ... are function arguments.
- As with any R function, they can be assigned default values.
- When you write a function, it is saved in your workspace as a function object.

A user-written function

The screenshot shows the RStudio interface with a script editor, console, and environment pane. A red box highlights the 'Run' button in the toolbar, with a tooltip that says 'Run the current line or selection (R+Enter)'. Another red box highlights the function definition in the script editor, and a third red box highlights the function 'f1' in the 'Functions' section of the Environment pane.

```
f1 = function(a, b){  
  #This function returns the maximum of two scalars  
  #for the statement that they are equal  
  
  if (is.numeric(c(a, b))){  
    if (a < b) return(b)  
    if (a > b) return(a)  
    else print("The values are equal")  
  }  
  else print("Character inputs are equal")  
}
```

Environment

Object	Class	Attributes
mydata	data.frame	4 obs. of 3 variables
mygeneticprofile	data.frame	7 obs. of 6 variables
Orange	data.frame	35 obs. of 3 variables
x	integer	int [1:3, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
a	integer	int [1:7] 1 2 3 4 5 6 7
cells	numeric	num [1:4] 1 26 24 68
cnames	character	chr [1:2] "C1" "C2"
d	numeric	num [1:4] 7 9 5 20
data_df	list	List of 0
e	character	chr [1:4] "red" "white" "red" NA
f	logical	logi [1:4] TRUE TRUE TRUE FALSE
mycancerstudy	character	"hnscc_tcga_pub"
mycaselist	character	"hnscc_tcga_pub_3way_complete"
mycgds	Classes 'CGDS', 'Object' atomic	[1:1] NA
mylist	list	List of 4
rnames	character	chr [1:2] "R1" "R2"
t	list	List of 4
table	list	List of 0
tables	list	List of 0
u	character	"<DOCTYPE html>\n<html class=\"client-nojs\" lang=\"en\" dl_
ulr	character	"<html>\n<head><title>400 Bad Request</title></head>\n<nb_
url	character	"https://en.wikipedia.org/wiki/List_of_countries_and_depende_

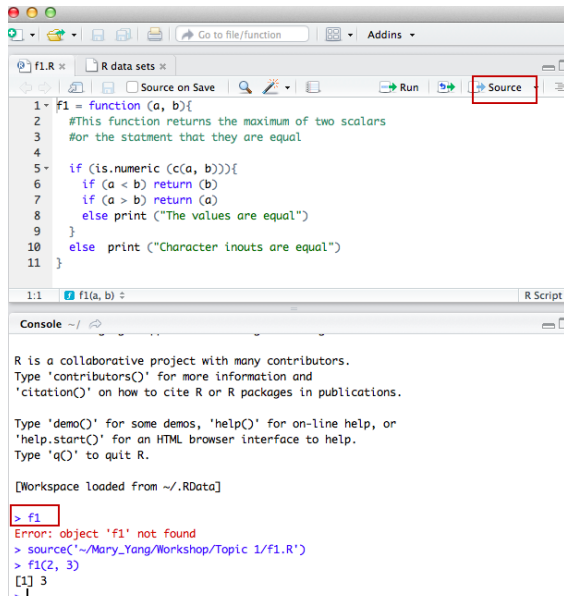
Functions

Function	Attributes
f1	function(a, b)

User defined function

- The function object **f1** will remain in your workspace until you remove it or quit R session.
- You can save these commands in R script (f1.R) for later-use.

Save user-written function for later-use



The screenshot shows the RStudio interface. The top toolbar has a red box around the 'Source' button. The editor window contains the following R code:

```
1 f1 = function (a, b){  
2   #This function returns the maximum of two scalars  
3   #or the statement that they are equal  
4  
5   if (is.numeric (c(a, b))) {  
6     if (a < b) return (b)  
7     if (a > b) return (a)  
8     else print ("The values are equal")  
9   }  
10  else print ("Character inputs are equal")  
11 }
```

The console window shows the following output:

```
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.  
  
[Workspace loaded from ~/.RData]  
> f1  
Error: object 'f1' not found  
> source('~/.Mary_Yang/Workshop/Topic 1/f1.R')  
> f1(2, 3)  
[1] 3
```

- It is recommended that you write and edit all of your R code in a script before you run it in the console.
- It creates a reproducible record of your work.
- You can save your script and then use it to rerun your entire analysis.
- Scripts are also very handy for editing and proofreading your code

- When you open an R Script (File → New File → R Script in the menu bar), RStudio creates a fourth pane above the console where you can write and edit your code.
- RStudio comes with many built-in features that make it easy to work with scripts.
 - R will run whichever line of code your cursor is on by clicking the **Run** button
 - If you have a whole section highlighted, R will run the highlighted code by clicking the **Run** button
 - Alternatively, you can run the entire script by clicking the **Source** button.
- To save a script, click the scripts pane, and then go to File → Save As in the menu bar

Built-In Function: numeric function

Function	Description
<code>abs(x)</code>	absolute value
<code><u>sqrt</u>(x)</code>	square root
<code>ceiling(x)</code>	<code>ceiling(3.475)</code> is 4
<code>floor(x)</code>	<code>floor(3.475)</code> is 3
<code><u>trunc</u>(x)</code>	<code><u>trunc</u>(5.99)</code> is 5
<code>round(x, digits=n)</code>	<code>round(3.475, digits=2)</code> is 3.48
<code><u>signif</u>(x, digits=n)</code>	<code><u>signif</u>(3.475, digits=2)</code> is 3.5
<code>cos(x), sin(x), tan(x)</code>	
<code>log(x)</code>	natural logarithm
<code>log10(x)</code>	common logarithm
<code>exp(x)</code>	<code><u>e</u>^x</code>

Built-In Function: character function

Function	Description
<code>substr(x, start=n1, stop=n2)</code>	Extract or replace substrings in a character vector. <code>x <- "abcdef"</code> <code>substr(x, 2, 4)</code> is "bcd" <code>substr(x, 2, 4) <- "22222"</code> is "a222ef"
<code>grep(pattern, x)</code>	Search for <i>pattern</i> in <i>x</i> .
<code>gsub(pattern, replacement, x)</code>	perform replacement of matches determined by regular expression matching
<code>strsplit(x, split)</code>	Split the elements of character vector <i>x</i> at <i>split</i> . <code>strsplit("abc", "")</code> returns 3 element vector "a", "b", "c"
<code>paste(..., sep="")</code>	Concatenate strings after using <i>sep</i> string to separate them. <code>paste("x", 1:3, sep="")</code> returns <code>c("x1", "x2" "x3")</code> <code>paste("x", 1:3, sep="M")</code> returns <code>c("xM1", "xM2" "xM3")</code> <code>paste("Today is", date())</code>
<code>toupper(x)</code>	Uppercase
<code>tolower(x)</code>	Lowercase

Built-In Function: statistical function

Function	Description
<code>mean (x)</code>	mean of object x
<code><u>sd</u> (x)</code>	standard deviation of object (x)
<code>median (x)</code>	median
<code>range (x)</code>	range
<code>sum(x)</code>	sum
<code>min (x)</code>	minimum
<code>max(x)</code>	maximum

Missing data

- In R, missing values are represented by the symbol NA Test for missing values:

```
#Test for missing values
```

```
> is.na (x)
```

```
#Excluding Missing Values from Analyses
```

```
> mean (x, na.rm = TRUE)
```

```
#Create a new dataset without missing data
```

```
> newData = na.omit (myData)
```

R packages

- R started with basic packages. To see the list of all available packages on systems, you can type into the R console window

```
> library()
```

- To install a new package, you can either click on Packages-install package(s), or type commands into the console window:

```
> install.packages ("gplots") # install a package called  
  gplots  
> library (gplots)} #load gplots packages
```

- To update a package called gplots

```
> update.packages ("gplots")
```

R Package

The screenshot shows the RStudio interface with the following components:

- Console:** Displays the execution of R commands to install the 'gplots' package and its dependencies. The output shows the download of 499 KB of data.
- Environment/History:** Shows the 'ade4' package installed from the CRAN repository.
- Files/Plots/Packages:** The 'Packages' tab is active, showing a list of installed and available packages. The 'gplots' package is highlighted in the list.
- Dialog Box:** A 'Configure Repositories' dialog box is open, showing the 'Repository (CRAN)' and the 'gplots' package selected for installation.

Console Output:

```
[32] "Top_Subnetwork_0.34_G0terms_Test.xls"
[33] "ytb.uid"
> library("biomaRt", lib.loc="/Library/Frameworks/R.framework/Versions/3.3/Resources/library")
> detach("package:biomaRt", unload=TRUE)
> install.packages("gplots")
trying URL 'https://cran.rstudio.com/bin/macosx/mavericks/con
Content type 'unknown' length 511025 bytes (499 KB)
downloaded 499 KB

The downloaded binary packages are in
/var/folders/yr/959dr91x20706jth9h5xtvc0000gn/T//Rtmpk8BTkI/downloaded_packages
> |
```

Environment/History:

Name	Description	Version
ade4	Analysis of Ecological Data : Exploratory and Euclidean Methods in Environmental Sciences	1.7-5

System Library:

Name	Description	Version
ade4	Analysis of Ecological Data : Exploratory and Euclidean Methods in Environmental Sciences	1.7-5
ade4	Methods for Affymetrix Oligonucleotide Arrays	1.52.0
ade4	Tools for parsing Affymetrix data files	1.44.0
ade4	A data package	1.16.0
ade4	Annotation for microarrays	1.52.1
ade4	Annotation Database Interface	1.36.2
ade4	Code for Building Annotation	1.16.1
ade4	Analyses of Phylogenetics and Evolution	4.1
ade4	Light-Weight Methods for Normalization and Utilization of Microarray Data using Only Basic R Data Types	3.4.0
ade4	Pre and post assertions.	0.1
ade4	Post C++ Header Files	1.62.0-1
ade4	Cluster Algorithms	1.2.0
ade4	BiocBase: Base functions for Bioconductor	2.34.0
ade4	BiocGenerics: S4 generic functions for Bioconductor	0.20.0
ade4	BiocInstaller: Install/Update Bioconductor, CRAN, and github Packages	1.24.0
ade4	BiocParallel: Bioconductor facilities for parallel evaluation	1.8.1
ade4	biomaRt: Interface to BioMart databases (e.g. Ensembl, COSMIC, Wormbase and Gramene)	2.30.0
ade4	Biostrings: String objects representing biological sequences, and matching algorithms	2.42.1
ade4	bitops: Bitwise Operations	1.0-6

The Workspace

- All variables or “objects” created in R are stored in what’s called the workspace
- To see what variables are in the workspace, you can use the function `ls()` to list them.
- To remove objects from the workspace use the `rm()` function:

```
# delete a object called  
> rm (X)
```

Manipulating file paths in R

Pathnames in R are written with forward slashes “/”, although in windows backslashes, “\”, are used.

#To set a working directory in R:

```
> setwd("Directory name")
```

#To print the current working directory:

```
> getwd()
```

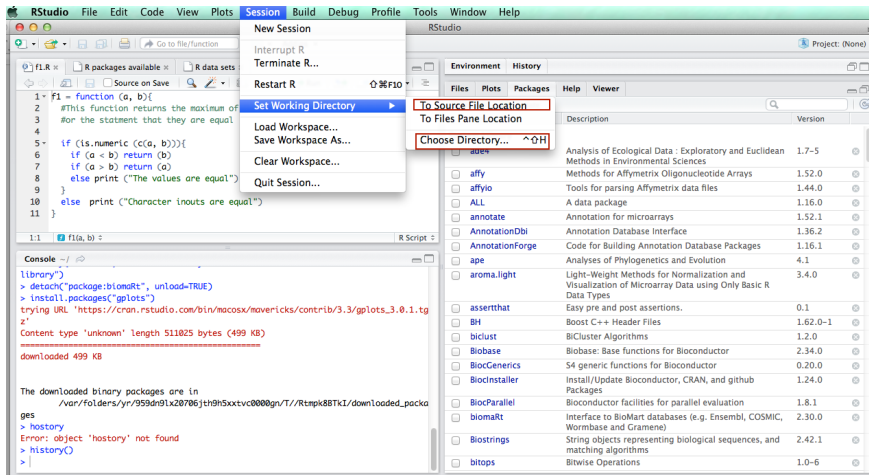
#To show the files in the current working directory:

```
> dir()
```

#To create a directory for your project - good way to organize your files

```
> dir.create("R Workshop 2017")
```

Manipulating file paths in RStudio



Getting Help

- R has an extensive help facility
- Apart from the Help window launched from the Help menu, it is also available from the command line prompt
- For instance

```
#explain about square root function  
> help(sqrt)  
> ?sqrt
```

- Most of the help files also include examples. You can run all of them by using the **example()**

```
#run all the examples from the matrix help file  
> example(matrix)  
  
#run all the examples from the plot help file  
> example(plot)
```

- Bioconductor is an open source and open development software project for the analysis of biomedical and genomic data.
- The project was started in the Fall of 2001 and includes developers in many countries

- Provide access to powerful statistical and graphical methods for the analysis of genomic data.
- Facilitate the integration of biological metadata (GenBank, GO, Entrez Gene, PubMed) in the analysis of experimental data.
- Allow the rapid development of extensible, interoperable, and scalable software.
- Promote high-quality documentation and reproducible research.
- Provide training in computational and statistical methods.

Bioconductor packages

- General infrastructure
 - Biobase, Biostrings, biocViews
- Annotation:
 - annotate, annaffy, biomaRt, AnnotationDbi
- Graphics/GUIs:
 - geneplotter, hexbin, limmaGUI, exploRase
- Pre-processing:
 - affy, affycomp, oligo, makecdfenv, vsn, gcrm, limma
- Differential gene expression:
 - genefilter, limma, ROC, siggenes, EBArrays, factDesign
- GSEA/Hypergeometric Testing
 - GSEABase, Category, GOstats, topGO

Bioconductor packages

- Graphs and networks:
 - graph, RBGL, Rgraphviz
- Flow Cytometry:
 - flowCore, flowViz, flowUtils
- Protein Interactions:
 - ppiData, ppiStats, ScISI, Rintact
- Sequence Data:
 - Biostrings, ShortRead, rtracklayer, IRanges, GenomicFeatures, VariantAnnotation
- Other data:

Bioconductor: Install packages

- To install core packages, type the following in an R command window:

```
> if (!requireNamespace("BiocManager", quietly = TRUE))  
  install.packages("BiocManager")  
> BiocManager::install()
```

- Install specific packages, e.g., GenomicFeatures and AnnotationDbi, with

```
BiocManager::install(c("GenomicFeatures", "AnnotationDbi"))
```