Introduction to R

January 21, 2020

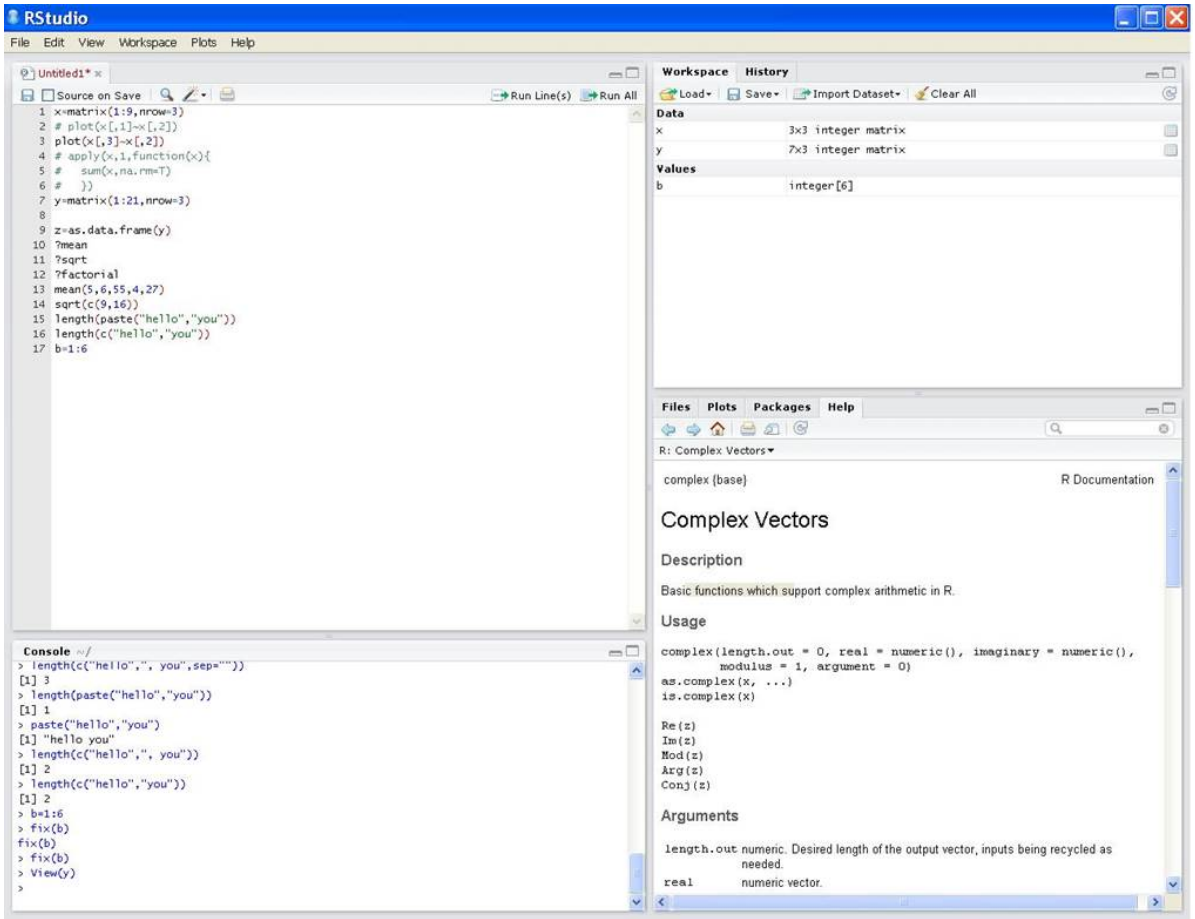
What is R?

- 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/**

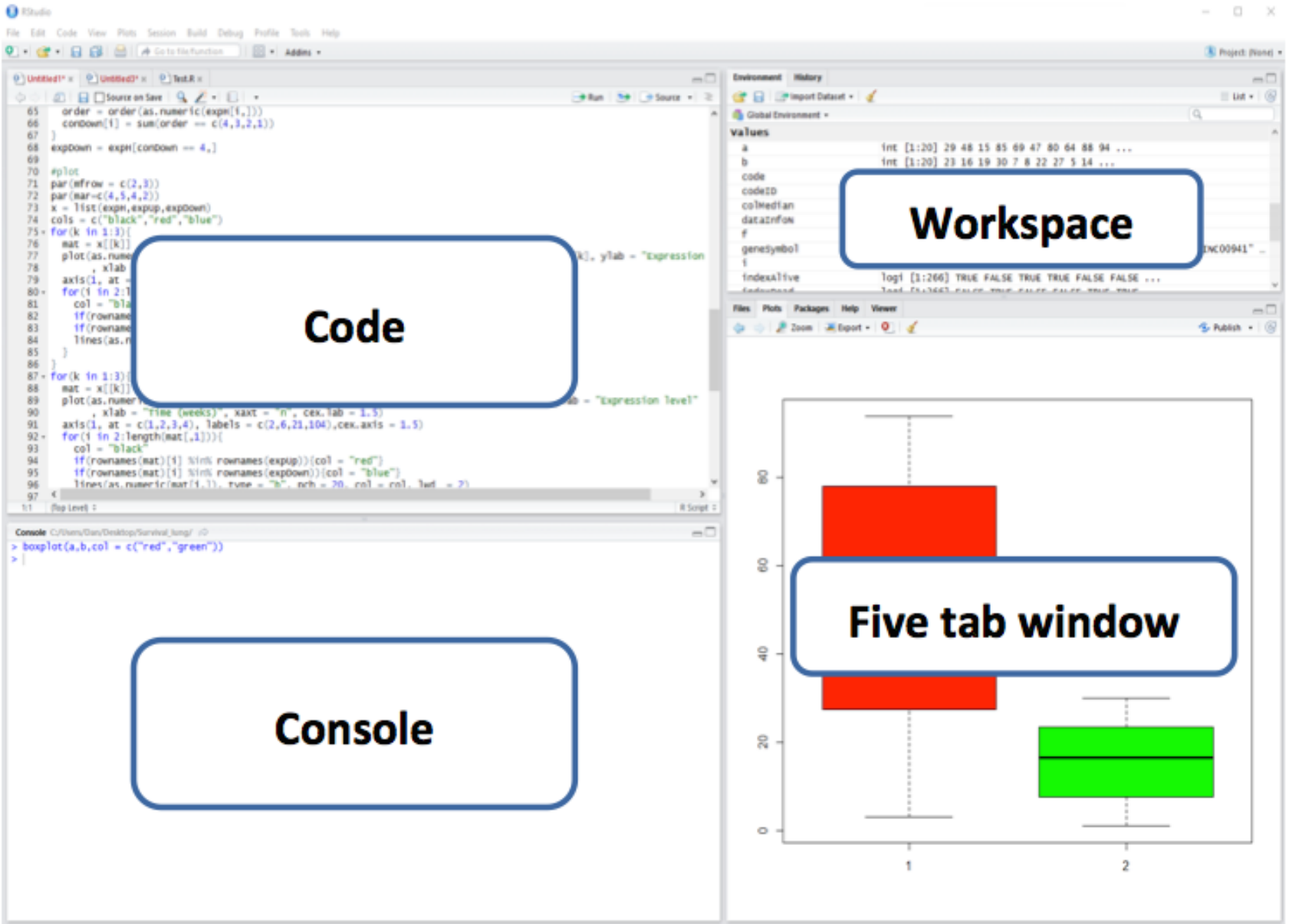


RStudio

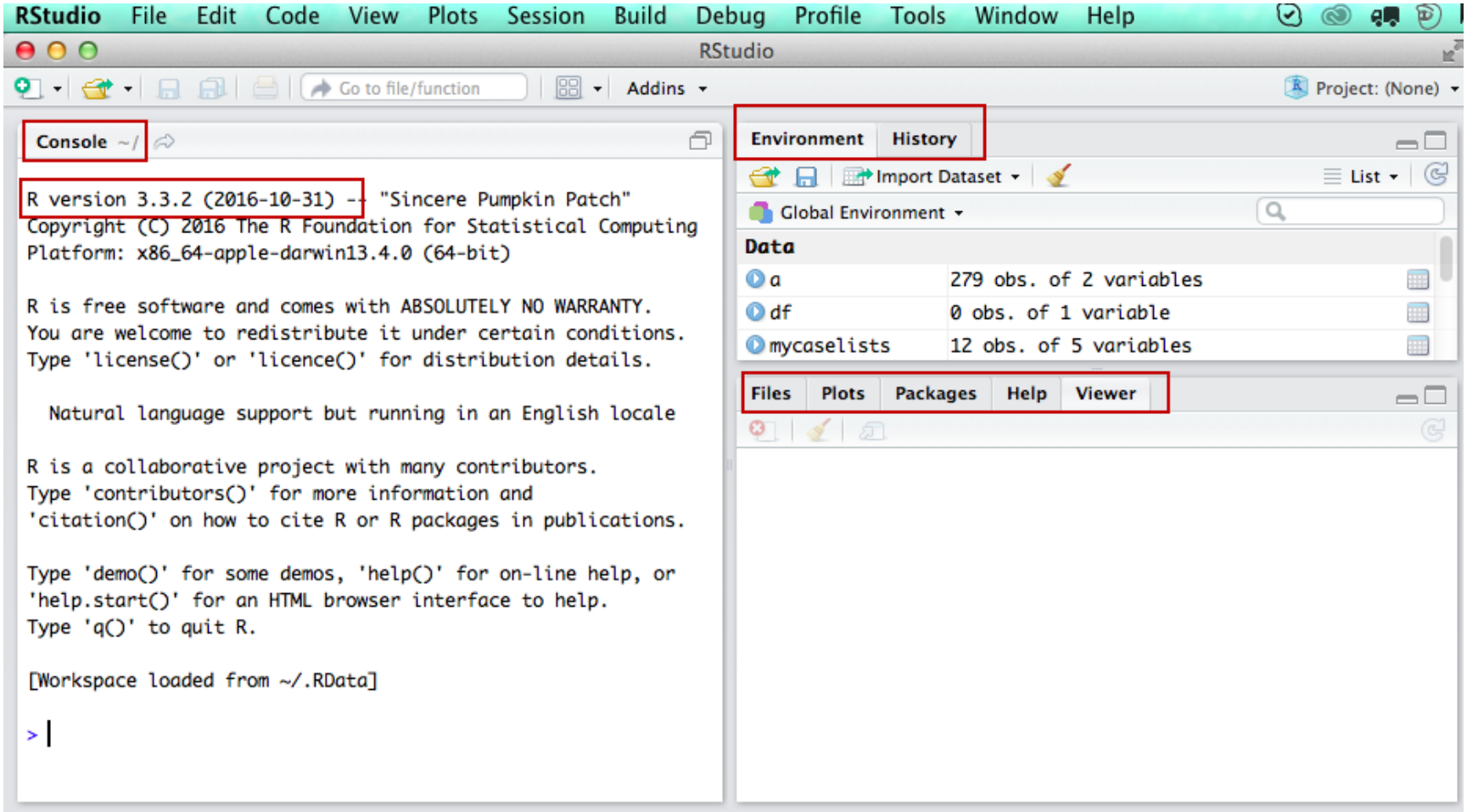
R Studio has with 4 Windows on the main screen

- 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

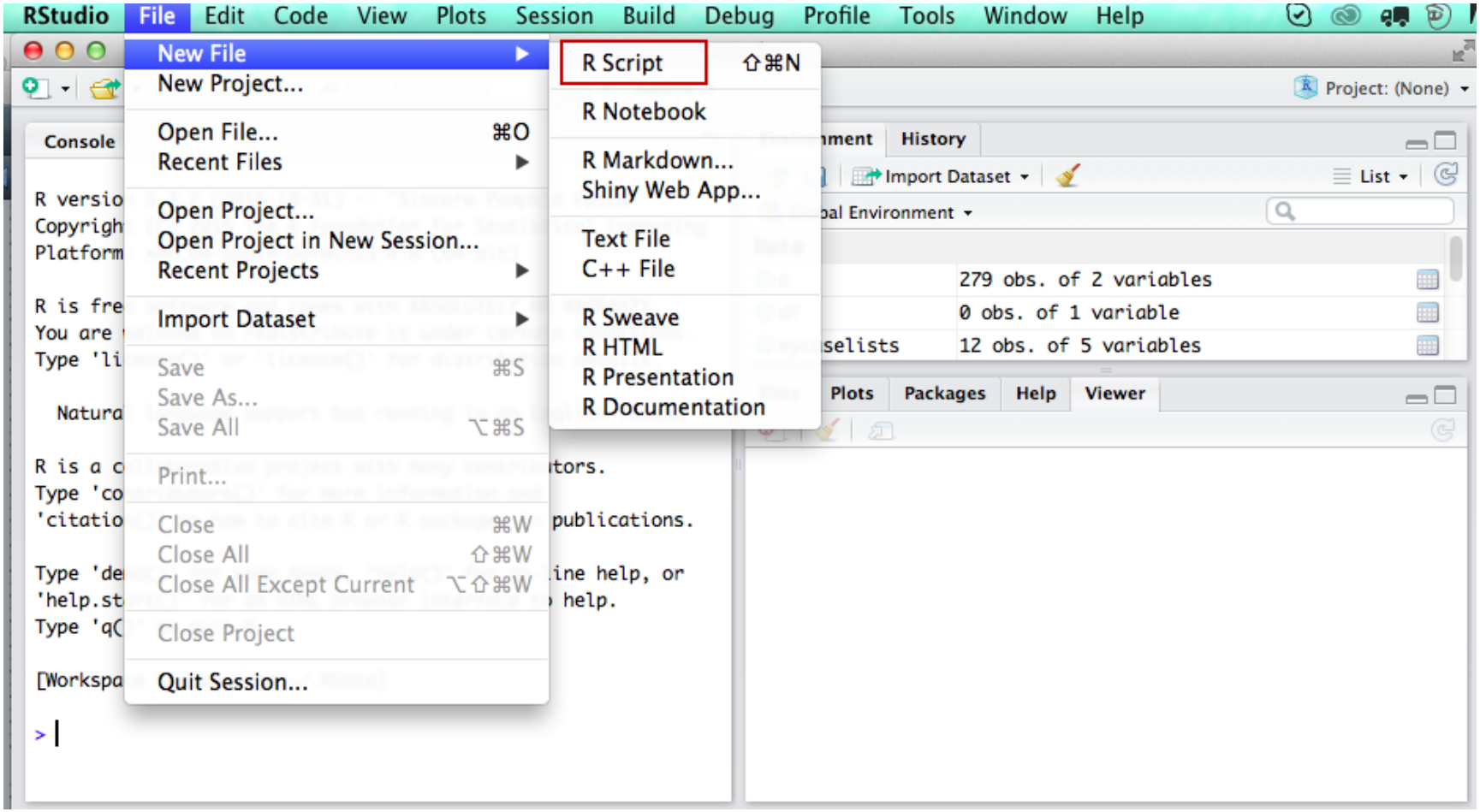
RStdio windows



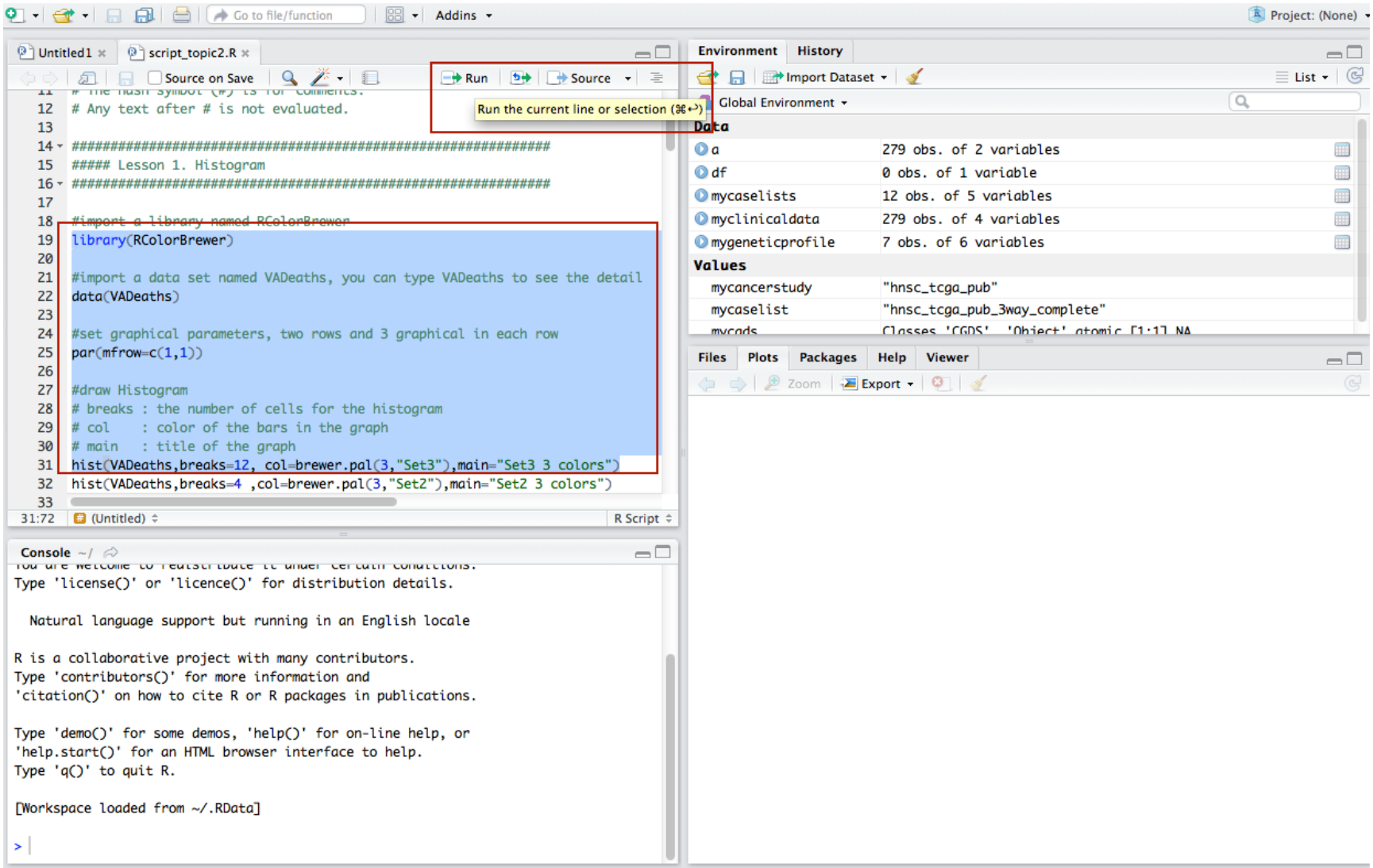
RStdio


RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
RStudio
Go to file/function
Addins
Project: (None)
Console ~/
R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (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.
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]
>
Environment History
Import Dataset
List
Global Environment
Data
a 279 obs. of 2 variables
df 0 obs. of 1 variable
mycaselists 12 obs. of 5 variables
Files Plots Packages Help Viewer

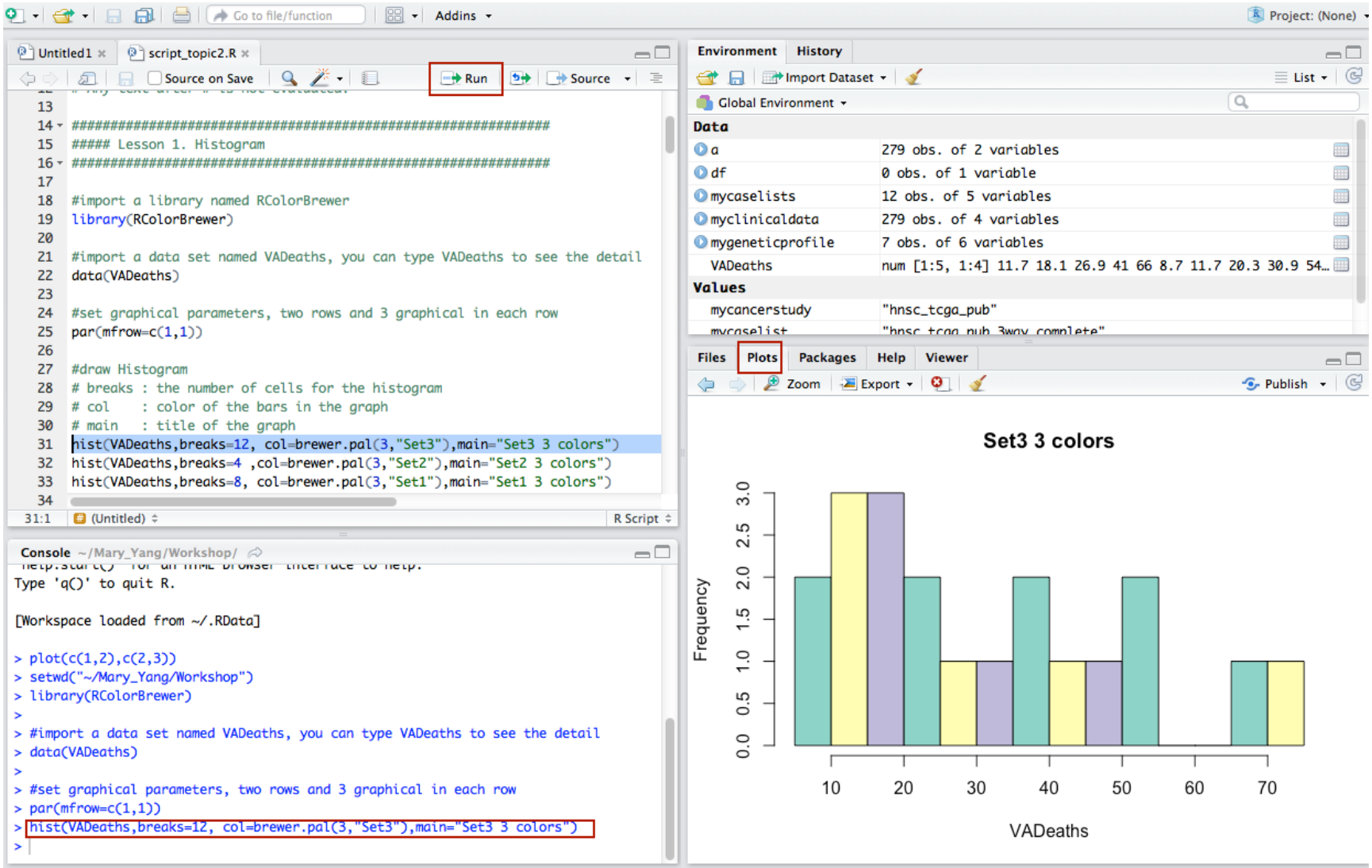
RStdio: create a new R script



RStdio: run R commands



RStdio: run a R command



R

- 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 Import

- 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

Data Import

- 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

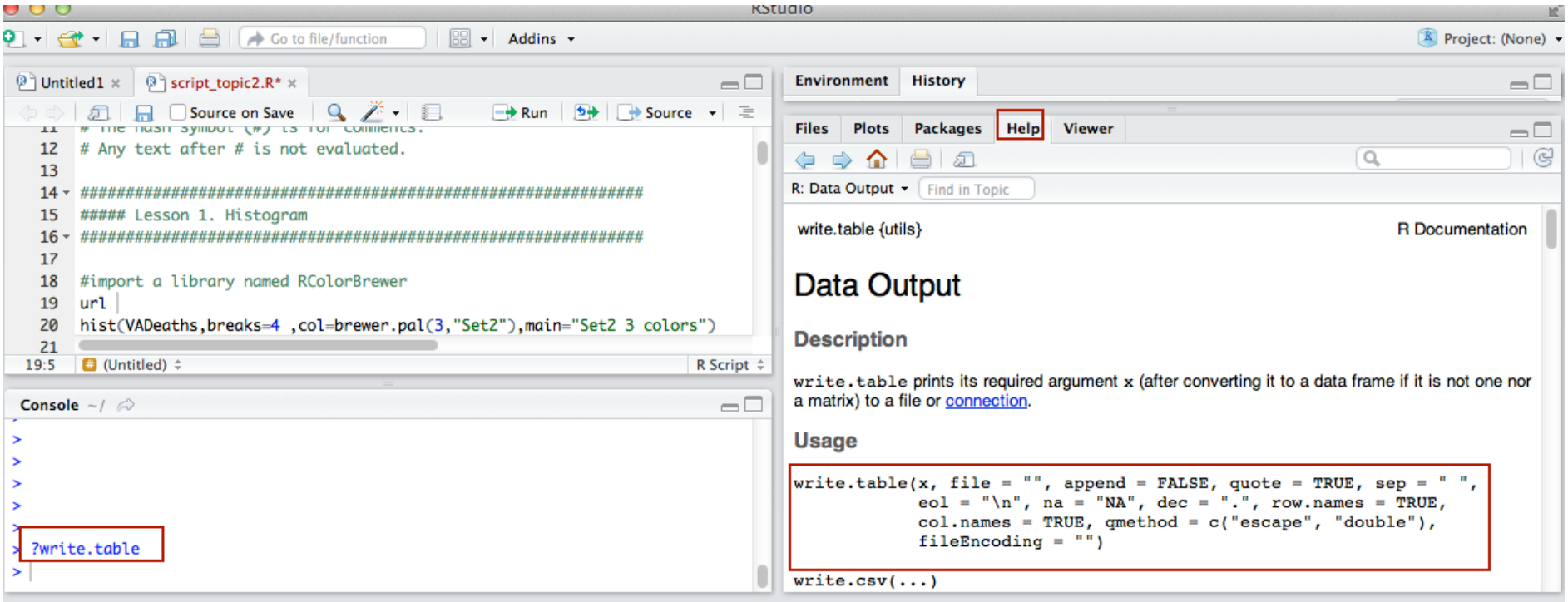
```
write.table(data, file = "path/file_name.txt", row.names =
    FALSE, col.names = TRUE)
```

- 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.

help

More options

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



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

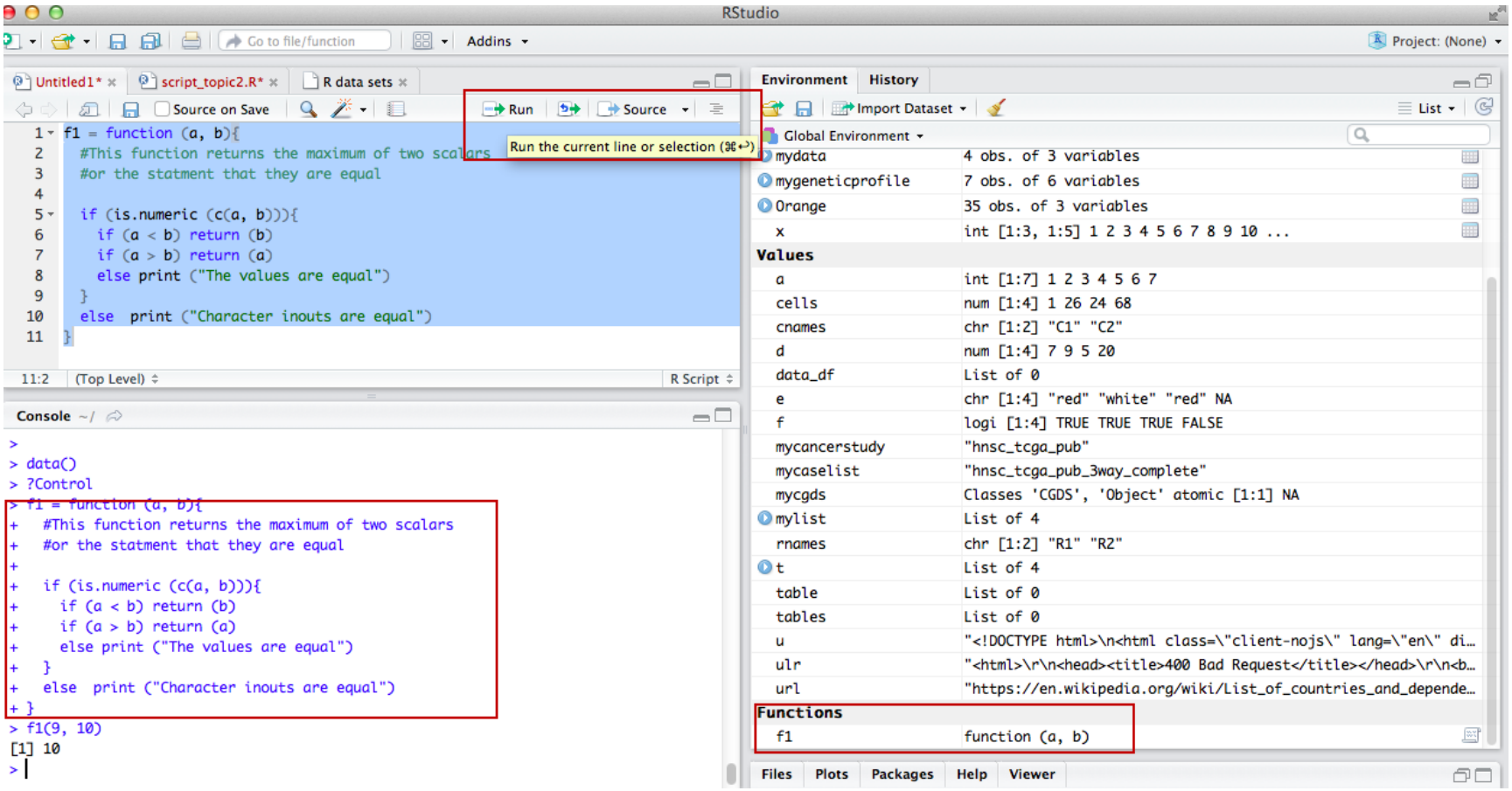
User-defined function

- 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.

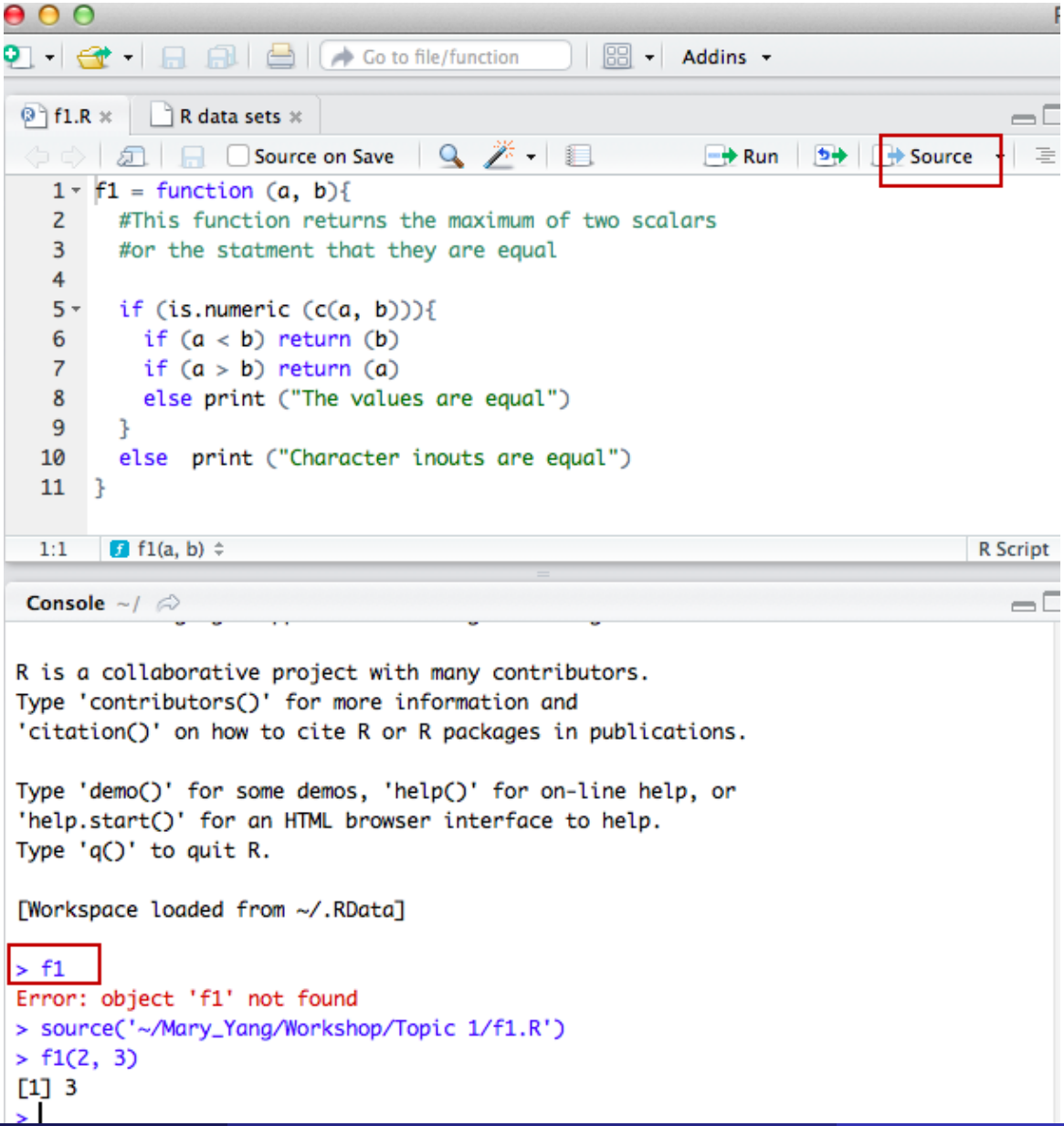
A user-written function



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



R Script

- 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

R Script

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

Built-In Function: character function

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

Built-In Function: statistical function

Function	Description
`mean (x)`	`mean of object x`
`sd (x)`	`standard deviation of object (x)`
`median (x)`	`median`
`range (x)`	`range`
`sum(x)`	`sum`
`min (x)`	`minimum`
`max(x)`	`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()
```

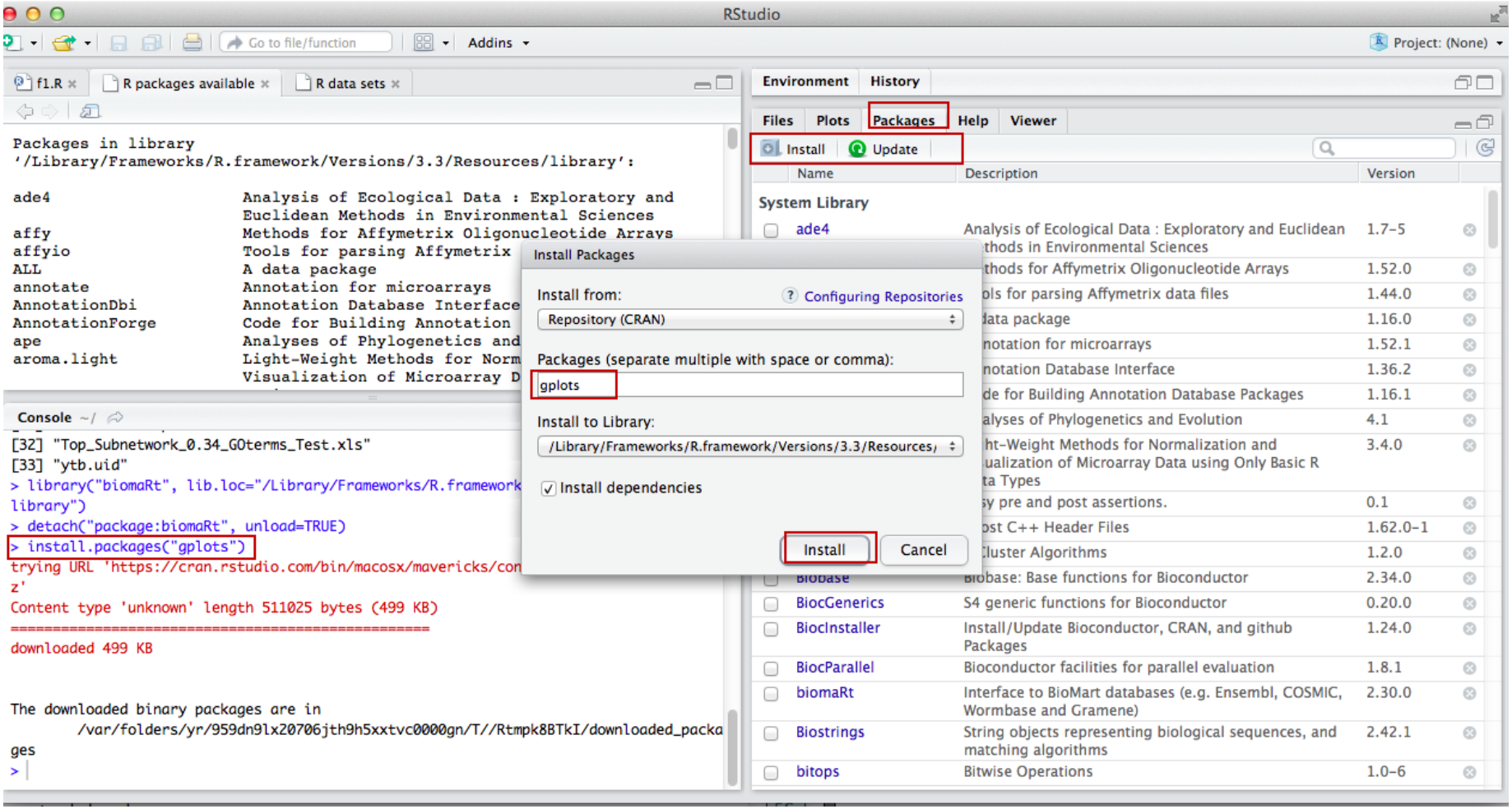
- To install a new package, you can either click on Packages-install package(s), or type commands into the console window:

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

- To update a package called gplots

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

R Package


RStudio
Go to file/function
Addins
Project: (None)
f1.R
R packages available
R data sets
Packages in library
'/Library/Frameworks/R.framework/Versions/3.3/Resources/library':
ade4 Analysis of Ecological Data : Exploratory and Euclidean Methods in Environmental Sciences
affy Methods for Affymetrix Oligonucleotide Arrays
affyio Tools for parsing Affymetrix
ALL A data package
annotate Annotation for microarrays
AnnotationDbi Annotation Database Interface
AnnotationForge Code for Building Annotation
ape Analyses of Phylogenetics and
aroma.light Light-Weight Methods for Norm
Visualization of Microarray D
Console ~/
[32] "Top_Subnetwork_0.34_GOterms_Test.xls"
[33] "ytb.uid"
> library("biomaRt", lib.loc="/Library/Frameworks/R.framework
library")
> detach("package:biomaRt", unload=TRUE)
> install.packages("gplots")
trying URL 'https://cran.rstudio.com/bin/macosx/mavericks/con
z'
Content type 'unknown' length 511025 bytes (499 KB)
==
downloaded 499 KB
The downloaded binary packages are in
/var/folders/yr/959dn9lx20706jth9h5xxtvc0000gn/T//Rtmpk8BTkI/downloaded_packa
ges
>
Environment
History
Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Version
System Library
ade4 Analysis of Ecological Data : Exploratory and Euclidean thods in Environmental Sciences 1.7-5
thods for Affymetrix Oligonucleotide Arrays 1.52.0
ols for parsing Affymetrix data files 1.44.0
data package 1.16.0
notation for microarrays 1.52.1
notation Database Interface 1.36.2
de for Building Annotation Database Packages 1.16.1
alyses of Phylogenetics and Evolution 4.1
ht-Weight Methods for Normalization and ualization of Microarray Data using Only Basic R ta Types 3.4.0
y pre and post assertions. 0.1
ost C++ Header Files 1.62.0-1
luster Algorithms 1.2.0
Biobase: Base functions for Bioconductor 2.34.0
BiocGenerics S4 generic functions for Bioconductor 0.20.0
BiocInstaller Install/Update Bioconductor, CRAN, and github Packages 1.24.0
BiocParallel Bioconductor facilities for parallel evaluation 1.8.1
biomaRt Interface to BioMart databases (e.g. Ensembl, COSMIC, Wormbase and Gramene) 2.30.0
Biostrings String objects representing biological sequences, and matching algorithms 2.42.1
bitops Bitwise Operations 1.0-6
Install Packages
Install from:
Configuring Repositories
Repository (CRAN)
Packages (separate multiple with space or comma):
gplots
Install to Library:
/Library/Frameworks/R.framework/Versions/3.3/Resources/
Install dependencies
Install
Cancel

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.

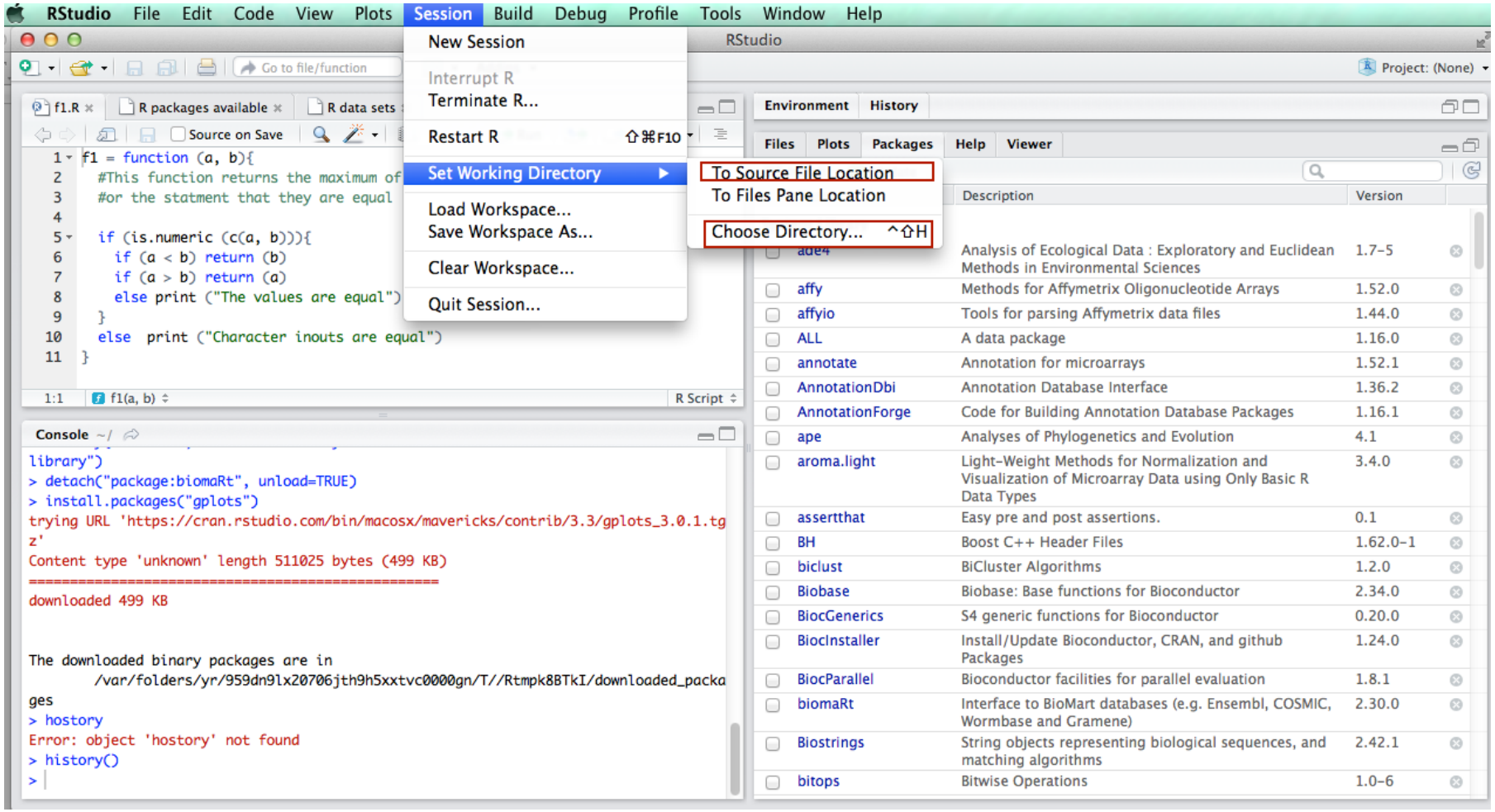
```
#To set a working directory in R:
> setwd("Directory name")

#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

- 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

Bioconductor

- 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"))
```