Introduction to R
2021-22

R Essentials

Summary
In this illustration, you will launch R-Studio, familiarize yourself with its windows and features, open a file where you will store your work, execute some commands, and, finally, save your work.

This illustration Assumes that You Have Installed R and R-Studio

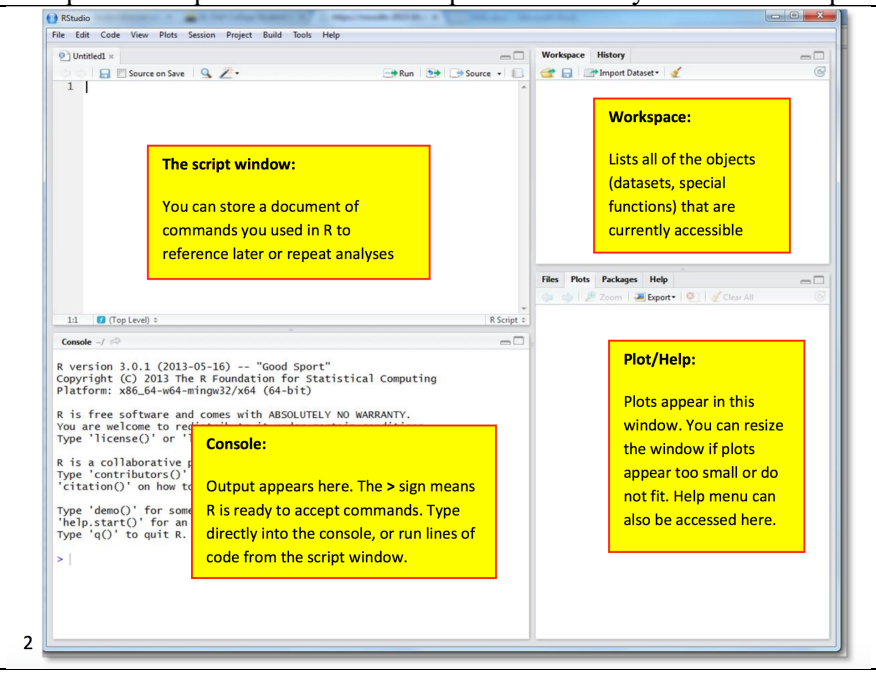
1. Launch R Studio and Have a Look Around

From your dock (Mac Users) or your taskbar (Windows User)
Click on the R Studio icon
Tip: Be sure to click on the R Studio icon, not the R icon.

Do click on R Studio	Do NOT click on R
	

1a. Panes in R Studio

You will then see EITHER 3 panes or 4 panes. Here are all 4 panes. You may not see the script window.



Source: *https://pages.stolaf.edu/boehm/files/2015/02/RStudioInfoFor272.pdf*

1b. Console Pane – Introduction

Welcome to the assignment operator (You could also use the equal sign but for good reasons, don't)

< - "Assign from the right to the left"

a < - 4 "For example - Assign to this thing "a" on the left this thing 4 that is on the right

```
#   Create a vector object
#  Use c() to create a variable (R calls this a vector object) - unsaved
c(1,2, 4, 8, 12, 13, 15)
## [1]  1  2  4  8 12 13 15

# Better is to use the assignment operator <- to save the vector object with the name v1
v1 <- c(1,2, 4, 8, 12, 13, 15)
# Bummer.  R Studio doesn't give you the result.
# You have to tell R Studio to display
# To view the contents of an object, simply type the name of the object
v1
## [1]  1  2  4  8 12 13 15

# Use class() to identify the type of object
class(v1)
## [1] "numeric"

# Create a random sample from a normal distribution
# Use rnorm(samplesize,mean,standarddeviation) to draw a sample of size 1000
y <- rnorm(1000,100,15)
# Convert this thing y that is a vector to a thing that is a a dataframe
# Use data.frame( ) to save your random sample in an object that R calls a data frame.
ydata <- data.frame(y)

# addition – Show the result but do not save it
4+6
## [1] 10

# Subtraction – Show the result but do not save it
4-6
## [1] -2

# Basic math in two steps:  (1) create the object y that is the solution (2) display the object y
y <- 4+6
y
## [1] 10

# Basic math in 2 steps connected by a semi-colon:  (1) create ;  (2) display
x<-5+8; x
## [1] 13

# Basic math in one step but now using parentheses to force R Studio to display
(x<-5+8)
## [1] 13

# Basic math with some annotation using paste( ) to produce reader friendly output
z<-8+16; paste("z = 8+16 = ",z)
## [1] "z = 8+16 = 24"
```

```
#    Some Useful tips for using the console
#    (1)  UP arrow to retrieve previous commands (so you can edit and re-use)
#    (2)  control-l   (that is letter el) to clear the console window (your history will not be lost)
```

1c. Script/Source Pane – Introduction to R Script

This pane is typically located at the upper left. Think of this utility as a "text editor". Here is where you write lines of code that will just sit there until you decide to do something with it.

What might you do with saved written lines of code? Answer:

- You might highlight it and then do a thing to execute it; OR
- You might save your written lines of code in to what is called a ".R" R script file (analogy: ".docx" file)

How to execute lines of code that are in your R Script file? Answer:

- METHOD 1: Highlight the code. Then do <control>-return OR
- METHOD 2: Highlight the code. Then, at top, click on the little run button.

Recommendation for how to make good use of R Script

Use R Script when you want to keep a record of your code (perhaps just for archival, but perhaps for re-use)

- step 1: Launch R Studio
- step 2: Open a new R Script file using FILE > NEW > R Script
- step 3: Immediately save it under a name that makes sense to you
- step 4: Begin your R Script file with a header of some sort (this will be a series of comments)
- step 5: Consider a set of intial commands such as setwd()
- step 6: Organize your R Script file into smaller "modules" of commands (e.g., input data, clean data, descriptives, etc.)

1d. Introduction to Packages

Like it or Not
R is all about Packages

Yes, you can do many things with your basic ("base R") installation – such as math and data manipulations – but, ultimately, you will be working with "add-on" packages.

Working with "add-on" packages requires TWO steps:

Step 1: Install the package *(one time)*
Step 2: Load (or require) the installed package to your RStudio session *(every session)*

1.d.1 Introduction

- Why do I need to know about packages? Answer: Packages are collections of commands (functions) and datasets. And the truth is, you will be working with LOTS OF THESE!
- When you installed R and RStudio, some packages came pre-installed so you already have them, eg: base and stats
- Going forward, trust me. You will want to install and use other packages.
- This is why you need to know about packages.

1.d.2 How to Install a Package

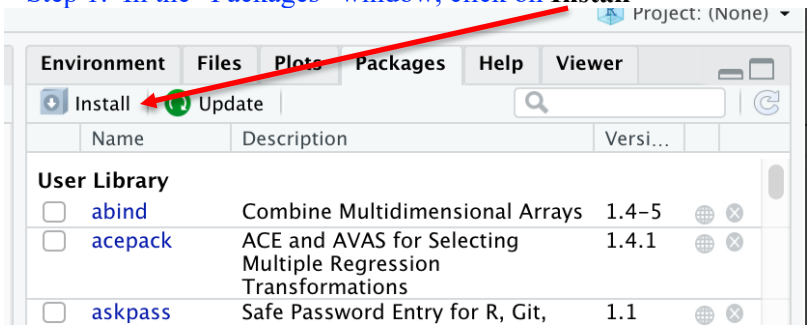
There's more than one way to download and install a package …

IMPORTANT
Do not put package installation commands into an R Script or R Markdown

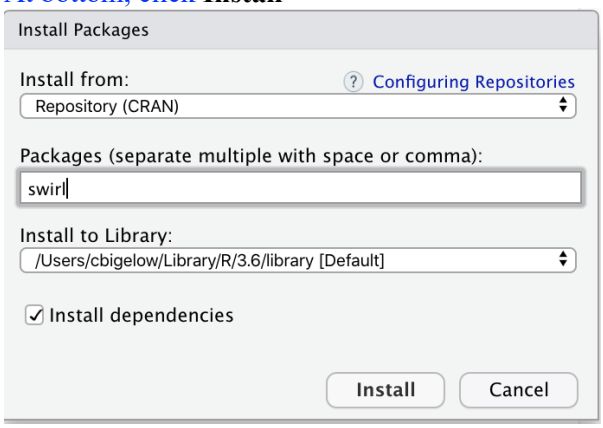
Why? You might think it would be fine to install a package from an R-script file (we'll come to this later) or from a R-markdown file (we'll also come to this later) but these are ***saved*** collections of commands which cannot be executed as a "batch" job.

METHOD I: Menu Driven (Easy and recommended)

Step 1: In the "Packages" window, click on **Install**



Step 2: In the "Install Packages" window
Example: I want to install the package called swirl
In Install from: **default (Repository CRAN) is fine**
In Packages (separate multiple with space or comma:) **type in name of package (e.g. swirl)**
In Install to Library: **leave as is**
Check box for "Install dependencies": **check this**
At bottom, click **Install**



Step 3: Be patient. Wait. Then take a look at your console window. You should see:

At bottom you should see the prompt ">" .

METHOD II: Command Driven (From the console window)

How to Install a Package from the Console Window

install.packages("nameofpackageinquotes")

Example: install.packages("foreign")
HACK: Don't forget the period in install.packages
HACK: The name of the package MUST be enclosed in double quotes
HACK: Always install packages in the console window
HACK: Never install a package within a R Markdown file (more on this later)

1.d.3 How to Attach a Package to Your Session

How to Attach a Package to Your R Studio Session

library(nameofpackageNOTinquotes)

Example: library(foreign)

1.d.4 Some Useful help commands related to packages

- To see what packages you have installed now, from console: library()
- To see what packages are attached to your session, from console: search()
- To get location of library containing packages, from console: .libPaths()
- To get help, from console: help(package="foreign")
  NOTE: The help will appear in the help/plot window at bottom right

1.d.5 How to Upgrade a Package

- Tip: Be sure to visit https://www.datacamp.com/community/tutorials/r-packages-guide
- To see what packages are old and need updating: old.packages()
- To update ALL packages: update.packages()
- To update JUST ONE package is a 2 step procedure:
  STEP 1: Remove the old package. Example: remove.packages("vioplot")
  STEP 2: Re-install the same package. Example: install.packages("vioplot")

2. Produce Numerical Descriptives

In Unit 1 (Numerical Summaries), we will learn several types of numerical descriptves (1 variable continuous, 1 variable discrete, 2 variables, etc). In this introduction, I am just showing you one example. Stay tuned. You will see lots more!

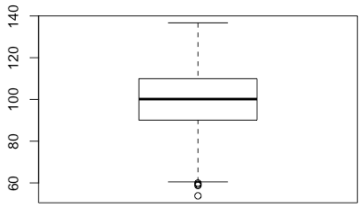
```
# Preliminary – Create some data to play with!
# Here, lets draw a random sample of observations from a Normal distribution (more on normals later)
# Use rnorm(samplesize,mean,standarddeviation) to draw a sample of size 1000
# NOTE:  YOUR sample of 1000 maybe different from mine
y <- rnorm(1000,100,15)
# Use data.frame( ) to save your random sample in an object that R calls a data frame.
ydata <- data.frame(y)

# ONCE EACH SESSION:  Use library( ) to attach the package stargazer
# This assumes that the package is already installed
library(stargazer)

# Plain - Use command stargazer() in package {stargazer} to produce descriptive statistics of every variable
stargazer(ydata, type="text")

##
## ================================================
## Statistic   N    Mean  St. Dev.  Min     Max
## ------------------------------------------------
## y         1,000 99.602  14.571  53.803 136.696
## ------------------------------------------------

# Fancy - Use command stargazer() to produce statistics of your choosing
stargazer(ydata,type="text", summary.stat=c("n", "mean", "sd", "min", "p25", "median", "p75", "max"))

##
## ============================================================================
## Statistic   N    Mean  St. Dev.  Min   Pctl(25) Median  Pctl(75)   Max
## ----------------------------------------------------------------------------
## y         1,000 99.602  14.571  53.803  90.138  100.162 109.934  136.696
## ----------------------------------------------------------------------------
```

3. Produce a Graph

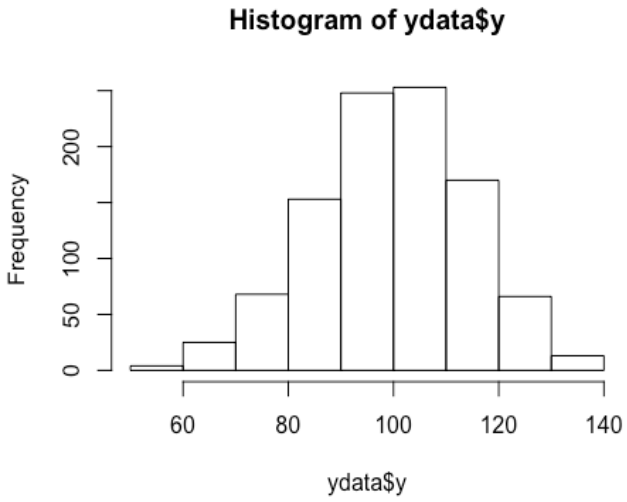
In Unit 2 (Data Visualization), as in Unit 1, we will learn several types of ways to graph data. We will alse learn that in R, there are lots of packages that do data visualizations. Of these, we will be using the package ggplot2 (you'll love it)

In this introduction, however, I am showing you how to do a simple graph using graphing commands that are in the base package.

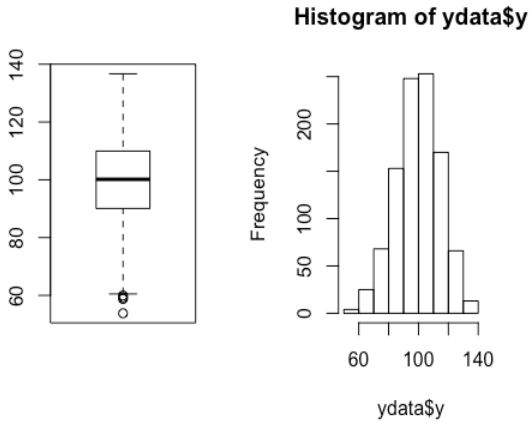
```
# BOXPLOT - Illustration of a box plot using basic installation of R (no package required)
# NOTE – You do not have the data to reproduce this plot on your own (sorry!)
# Command is boxplot(DATAFRAMENAME$VARIABLENAME) to make a box plot
boxplot(ydata$y)
```




```
# HISTOGRAM - Illustration of a histogram using basic installation of R (no package required)
# NOTE – You do not have the data to reproduce this plot on your own (sorry!)
# Command is hist(DATAFRAMENAME$VARIABLENAME) to make a histogram

hist(ydata$y)
```



```
# TWO PANEL FIGURE  - Create a TWO PANEL graph (3 steps, but not hard)

# Step 1 - Use par() to define panel as having 1 row and 2 columns
par(mfrow=c(1,2))

# Step 2 - Produce the two panel graph
boxplot(ydata$y)
hist(ydata$y)
```




```
# Step 3 - Return the graph setting to 1 row and 1 column (important!)
par(mfrow=c(1,1))
```

R Essentials
Summary

Preliminary – Set Your Working Directory

Command	Example	Description/Notes
getwd( )	getwd( )	Get current working directory
setwd( )	setwd("/Users/cbigelow/Desktop")	Set current working directory *What could go wrong*: - you forgot to enclose in quotes

Create R Objects

Command	Example	Description/Notes
c( )	x <- c(1,3,5)	Create vector called x NOTES: - elements separated by commas
objectname	x	View elements of x
class( )	class(x)	Identify type of object; eg – "numeric"
data.frame( )	ydata <- data.frame(y)	Create data frame called ydata using information in y
rnorm(sample size,mean,sd)	y <- rnorm(1000,100,15)	Creates a vector y that is a random sample from distribution: Normal sample size: 1000 mean: 100 standard deviation: 15

Create R Objects

Command	Example	Answer
+	3+7	10
-	3-7	-4
*	3*7	21
/	3/7	0.42857
^ or **	7^3	$7^3 = 7*7*7 = 343$
sqrt	sqrt(3)	1.732051
log	log(2)	Natural log: ln(2) = 0.693
log10	log10(2)	Log base 10: $\log_{10}(2) = 0.30103$
exp	exp(2)	$e^2 = 2.718282^2 = 7.389056$

- continued -

R Essentials
Summary - continued

Descriptive Statistics Using Package *stargazer*

Command	Example	Description/Notes
stargazer( ) **Here: produces default statistics**	stargazer(ydata, type="text")	Produce n, mean, standard deviation, min, and max
stargazer( ) **Here: user specifies statistics desired**	stargazer(ydata type="text", summary.stat=c("n", "mean", "sd", "min", "p25", "median", "p75", "max"))	Produce n, mean, standard deviation, minimum, 25th percentile, median, 75th percentile and maximum

Produce a graph

Command	Example	Description/Notes
par(mfrow=c(**# rows, # columns**))	par(mfrow=c(1,1)) par(mfrow=c(1,2))	Define graphic design to be ONE panel: 1 row, 1 column Define graphic design to have TWO panels in one: 1 row, 2 columns.
boxplot( )	boxplot(ydata$y)	Box plot of variable y that is in data frame ydata
hist( )	hist(ydata$y)	Histogram of variable y that is in data frame ydata