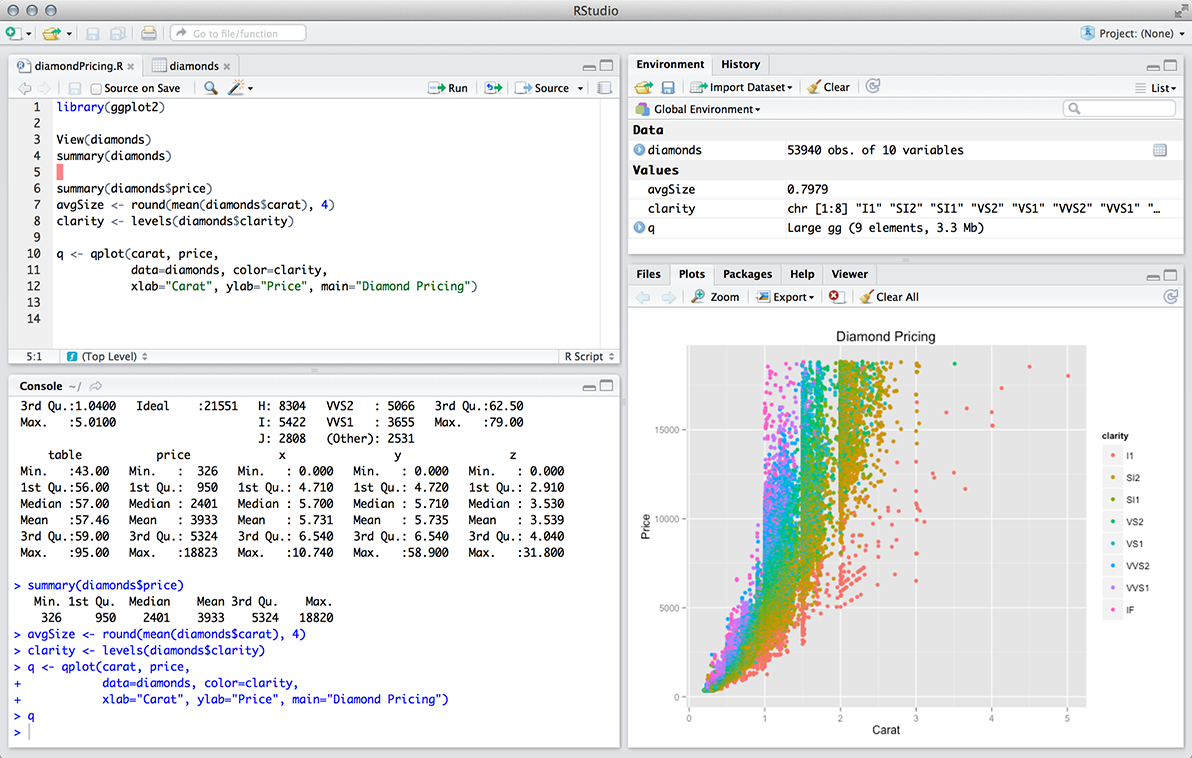
Introduction to Statistics for Psychologists

Claremont McKenna College

Professor Cook

1. R is a powerful language and environment for statistical computing and graphics. The main advantages of R is the fact that R is free. It is quite similar to other programming packages such as MatLab, but more user-friendly than programming languages such as C++ or Fortran. You can use R as it is, but we prefer to use R in combination with the RStudio interface, which has an organized layout and several extra options
2. **RStudio Interface.** RStudio is an R integrated development environment (IDE) for R, which is more user friendly environment than the R GUI. You may notice upon opening RStudio that the interface consists of several window panes.



1. Bottom left window: **The Console Window** (also called the command window)

Here you can type simple commands after the “>” prompt and the press the ENTER (Windows) or RETURN (Mac) key to execute your command. This is the most important window, because this is where R actually runs the script and displays nongraphical results and analyses.

1. Top left window: **The Editor Window** (also called the script window)

Collections of commands (scripts) can be edited and saved. When you do not see this window, you can obtain it by selecting File → New → R script. Just typing a command in the editor window will not run the script; it must be run in the command window. If you want to run a line from the script window (or the entire script), you can click Run or press CTRL+ENTER (MAC: COMMAND + ENTER(RETURN)) to send it to the command window. For this class, we will use this window often when creating script files for analyzing data because doing so will allow us to keep a record of what we completed. Creating scripts is essential for reproducing your results for homework and other projects.

1. Top right window: **The Workspace Window**

In the workspace window you can see which data and values R has in its memory. You can view and edit the values by clicking on them. The history window shows what has been typed before.

1. Bottom right window: **The Files / Plots / Packages / Help Window.**

Here you can open files, view plots (also previous plots), install and load packages, or to use the help function.

1. **Libraries.** R can perform many statistical and data procedures that are not available in the SPSS version that CMC licenses. In R, statistical procedures are organized in packages or libraries written by open-source contributors. With the standard installation, most common packages are installed, however, many more packages available on the [CRAN website](https://cran.r-project.org/) (Comprehensive R Archive Network). We will install additional packages for this class. If you want to install and use a package (for example, the package called “psych”) you could type in the command pane:

#The following command installs the package on your computer

install.packages("psych")

install.packages("moments")

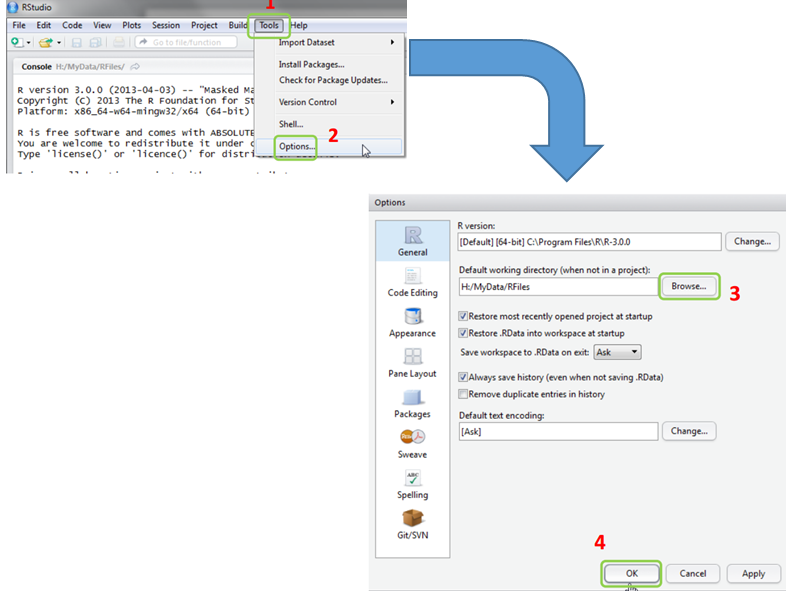
# The following command loads the package for use in the script

library("psych")

library("moments")

There is another way to install libraries, but you will still need to load the library using the library() function. Because the R language is case sensitive, it discriminates between uppercase and lowercase letters in the names of the objects. You must be careful to For example, capitalizing L for the function -- Library("psych") -- will not work.

1. **Using RStudio**
2. **Installing Packages in RStudio.** 
   1. Rather than typing all that code, RStudio makes adding packages easy.
   2. Tools 🡪 Install Packages 🡪 Type in the package name 🡪 Click Install (more on this below)
3. **Setting Default Working Directory.** Every time you open RStudio, the program starts in a default directory. You can change the default directory to a location where you have your data files so you do not have to specify the location every time you open RStudio. To set the default working directory go to Tools->Options



Then set the default directory.

1. **Getting and Setting A Working Directory.** A working directory is the directory/folder on your computer in which you are currently working. When you ask R to open a certain file, it will look in the working directory for this file, and when you tell R to save a data file or figure, it will save it in the working directory. Before you start working, please set your working directory to where all your data and script files are or should be stored.

Windows example:

getwd() # get the absolute path to the current working dir

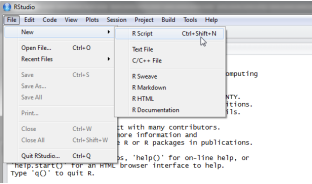
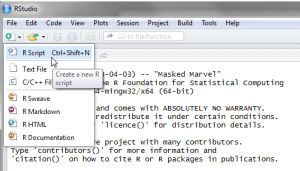
setwd("C:/Users/your\_user\_name/Desktop/CMC109")#set the path location

Mac example:

getwd() # get the absolute path to the current working dir

setwd("/Users/your\_user\_name/Desktop/CMC109/") #set the path location

1. **Creating R Script Files in RStudio.**
   1. To create a new R script you can either go to File -> New -> R Script, or click on the icon with the “+” sign and select “R Script”. The shortcut CTRL+SHIIFT+N (Windows) or COMMAND+SHIFT+N (Mac) will also create a new R Script (.R) for you.

* 1. To edit the script file type commands after the line numbers. Press ENTER(RETURN) to move to the next command line. Saved files can be constantly edited and resaved allowing you to run an analysis multiple times without writing the script again.
  2. To save the script file go to File-> Save and type the name of your saved R Script, or click on the floppy disk icon to save the file. The shortcut CTRL+S (Windows) or COMMAND + S (Mac) will also save the R Script (.R) for you
  3. Running Code in Script Files.
     1. To run one line of code in the script file just leave the cursor anywhere on the line where the command is and press CTRL+ENTER (Windows) or COMMAND+ENTER(RETURN)(Mac) or click on the ‘Run’ icon above. The Output will appear in the console below the script.
     2. To run a whole section of code in the script file just highlight the lines you wish to run and press CTRL+ENTER (Windows) or COMMAND+ENTER(RETURN)(Mac) or click on the ‘Run’ icon above. The Output will appear in the console below the script.

1. **Using the R Command/Console.** To get you started, enter the following command in the console after the “>” prompt.

#Read the csv file

cdc <- read.csv("cdc.csv”)

That command will bring in data from a random sample of 20,000 people from the Behavioral Risk Factor Surveillance System survey conducted in 2000. Although there are over 200 questions or variables in this dataset, this data set only includes 9 variables.

* **genhlth**: A categorical variable indicating general health, with categories excellent, very good, good, fair, and poor.
* **exerany**: A categorical variable, 1 if the respondent exercised in the past month and 0 otherwise.
* **hlthplan**: A categorical variable, 1 if the respondent has some form of health coverage and 0 otherwise.
* **smoke100**: A categorical variable, 1 if the respondent has smoked at least 100 cigarettes in their entire life and 0 otherwise.
* **height**: A numerical variable, respondent's height in inches.
* **weight**: A numerical variable, respondent's weight in pounds.
* **wtdesire**: A numerical variable, respondent's desired weight in pounds.
* **age**: A numerical variable, respondent's age in years.
* **gender**: A numerical variable, respondent's gender.

In the R console enter:

cdc

This command will show the data in the R console. Note that the row numbers in the first column are not part of the data. You can think of the row numbers as the index that you see on the left side of an Excel spreadsheet; those numbers are there to help guide you. In this case, the index corresponds to the number of rows of data are in the file. In fact, the comparison to a spreadsheet will be generally helpful.

Below are some commands in R to examine the data:

|  |  |  |
| --- | --- | --- |
| Command | General Code | Example |
| **Examining a Data Frame** | | |
| Is Object a Data Frame | is.data.frame(*object*) | as.data.frame(cdc) |
| Coerce into Data Frame | as.data.frame(*object*) | as.data.frame(cdc) |
|  |  |  |
| See Column Names | names(*dataframe)* | names(cdc) |
| Structure of R Object | str(*dataframe*) | str(cdc) |
| Dimensions of R Object | dim(*dataframe*) | dim(carbon copied) |
| First Several Cases in DF | head(*dataframe*) | head(cdc) |
| Last Several Cases in DF | tail(*dataframe*) | tail(cdc) |
| View Data in a Tab | View(*dataframe*) | View(cdc) |
|  |  |  |
| **Examining Data** | | |
| Mean | mean(*dataframe$variablename*) | mean(cdc$weight) |
| Median | median(*dataframe$variablename*) | median(cdc$weight) |
| Variance | var(*dataframe$variablename*) | var(cdc$weight) |
| Standard Deviation | sd(*dataframe$variablename*) | sd(cdc$weight) |
| Obtain Summary | summary(*dataframe$variablename*) | summary(cdc$weight) |
| Skewness | skewness(*dataframe$variablename*) | skewness(cdc$weight) |
| Kurtosis | kurtosis(*dataframe$variablename*) | kurtosis(cdc$weight) |
|  |  |  |
| **Help** | | |
| Help | help(*command\_name*) or ?*commandname* | ?mean |

View(cdc) # View the data file

is.data.frame(cdc) # Determine if the data is a data frame object

str(cdc) # Let’s you know the type of the data (e.g., numeric, integer, Factor, etc.)

dim(cdc) # dimensions of the data frame; 20,000 observations; 9 columns

names(cdc) # What are the column/variable names in the data frame?

head(cdc)

tail(cdc)

One issue with the above is when the column in data frame (aka. *vector*) …..has NA values. Functions default na.rm = FALSE, so this returns a NA

x <- c(4, 3, 5, 0, 7, 8, NA)

mean(x)

mean(x, na.rm = TRUE)

median(x, na.rm = TRUE)

var(x, na.rm = TRUE)

sd(x, na.rm = TRUE)

library(moments)

skewness(x, na.rm = TRUE)

kurtosis(x, na.rm = TRUE)

The best method to learn R is practice and it is highly encouraged to mess around with the variables and various features and commands.

Practice:

Choose a different variable from the cdc data set and:

Mean

Median

Histogram