Lab 1: R Basics and Working with Data

Stats 10: Introduction to Statistical Reasoning

Objectives

1. Demonstrate basic R Skills (create vectors, perform vector operations, install packages)
2. Use basic statistical functions (mean, min, max, median, sd)
3. Visualize data (dot plot, histogram, box plot, contingency tables, scatter plot)

Software

1. Download the latest version of R at
   If you are using Mac: https://cran.r-project.org/bin/macosx/
   If you are using Windows: https://cran.r-project.org/bin/windows/base/
2. After R is installed, download the latest version of RStudio (the free one on the left): https://www.rstudio.com/products/rstudio/download/
3. If you are not able to / do not want to download the software on your desktop, you can access to our department's server running RStudio Server software. Rstudio Server can be found at http://rstudio.stat.ucla.edu/

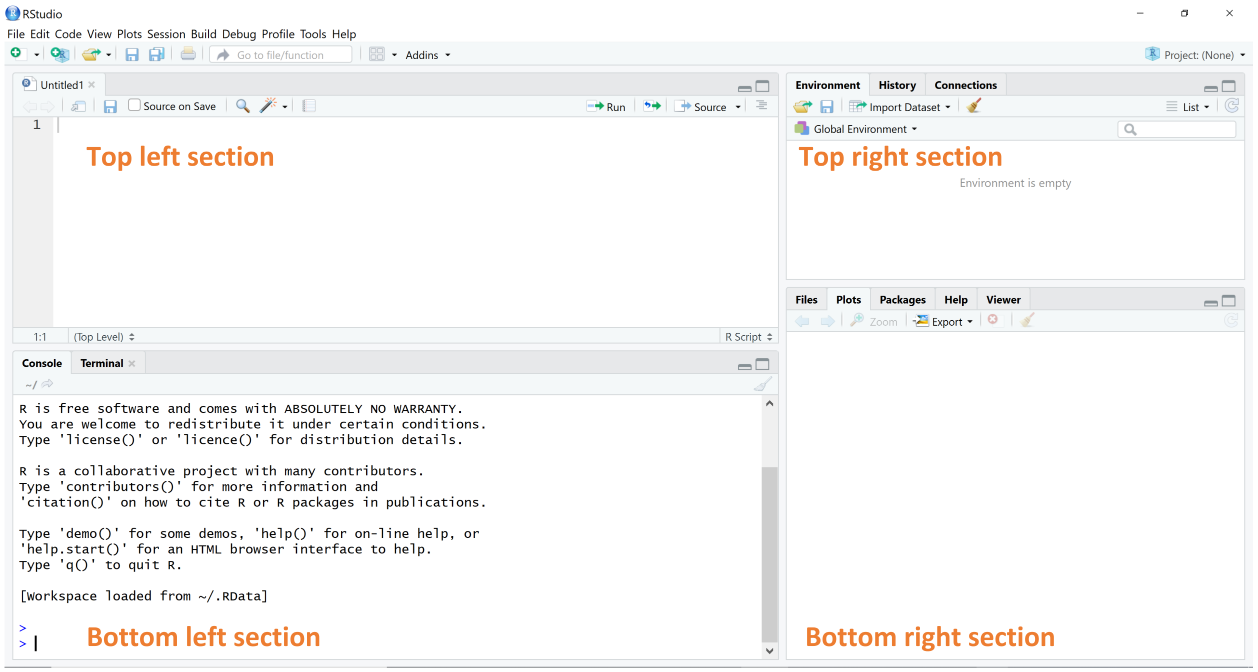
A Primer on labs

In Stats 10, the lab courses are designed to help you explore the capabilities of computer programming to assist you in statistical analysis. This quarter we will be using R, a high-level statistical programming language. We will cover several basic topics in R to help you carry out analysis that you learn in lecture. R is a popular program in universities because it is free and open-source, which means anyone can access the program's source code and suggest improvements.

Section 1 – R and RStudio Basics

Instructions

When you open RStudio, you'll notice the window is partitioned into four sections:



```
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.

[Workspace loaded from ~/.RData]

>
> |
```

- **The top left section** is your working code file. This is where you should type and run your commands. It serves two main purposes:
  - You can edit and save this section just like a word document. In fact, let's save right now. Go to file>save, and save this in a convenient location on your computer.
  - You will run lines of code by highlighting the code chunk you want to run, and clicking the run button (shortcut: ctrl+enter for PC, cmd+enter for Mac)
- **The bottom left section** is the "console". This is essentially the R engine that runs your commands and produces output. You should not need to type anything in this section, but you will need to copy and paste output for exercises.
- **The top right section** defines your environment. The environment contains any objects you create and data you import. You won't use it much but it's a quick way to see what data and objects you have to work with.
- **The bottom right section** contains three important tabs in this course:
  - **Plots:** when you create plots, they will show up here. You should copy and paste these to answer some of the lab exercises.
  - **Packages:** R contains packages that you can install and run to use cool features and functions that the basic R install did not give you.
  - **Help:** If you are unsure about what a function does, try searching for it in the help tab. You will find information about the function arguments and examples.

Working with packages

To work with packages, we must do two things:

1. Install the package by using the packages tab mentioned above
2. Type and run library(“package_name”) to tell R to load the package we installed

Reading Data into RStudio

R needs to be told which folder to look in to retrieve data. To read in data, first download it into a convenient folder on your machine. Then, try one of these methods:

1. At the top of RStudio, go to Session> set working directory> choose directory. Then select the folder the data is saved in, and click “Select Folder”. Then, create a new data object by using the syntax object_name <- read.csv(file = "filename.csv")
2. Use the syntax object_name <- read.csv(file.choose()) and use the file explorer to manually select the downloaded data.

Vectors – Examples

Try running the commands below. The first creates a numeric vector object called *numbers* that contains the numbers 1 to 5. The second creates a character vector object called *schools* that contains the names of two top-tier colleges, and the name of a terrible school for spoiled children. Note that when you create a vector of names, you must put quotes around each element. The c() function is used to create both vectors. Try typing c() in the help menu to get a better sense of what this function does.

```
numbers <- c(1, 2, 3, 4, 5)
```

```
schools <- c("UCLA", "UC Berkeley", "USC")
```

Note that when you run these commands, there is no output. That is because whenever you define an object, there is not output. However, once you have made the object, you can print the contents of an object by typing the name of the object. Example: type numbers

If the vector is numeric, we can do math on it. For example, try typing numbers * 2

We can also use square brackets to create subsets of our data. For example, if you type schools[2] you will retrieve only the second element from the schools vector.

Object classes

Vectors, matrices, and data frames are three common object classes you will work with. Matrices and data frames can be thought of as tables of data. However, matrices contain only numeric data. Data frames can contain several types of data. The file you will work with in this lab is a sample of information about babies born in North Carolina. It is considered a data frame because it contains numeric information about each baby, as well as various pieces of categorical data such as the race of the parents, and whether the mom has a smoking habit.

Section 2 – Summarizing Data (one variable)

Useful functions

The functions summary(), mean(), sd(), max(), and min() all produce helpful results to help us understand how quantitative data is distributed. These functions will take in a numeric vector inside the parentheses. As an example, max(NCbirths$weight) will give us the maximum weight of all the babies in out sample.

We can also use the tally() function in the mosaic package to help us summarize categorical data. Tally requires a categorical vector inside the parentheses. For example, after you have installed and loaded the mosaic package, try typing tally(NCbirths$Racemom).

Section 3 – Visualizing Data (one quantitative variable)

Useful functions

The histogram, dot plot, and box plot are useful tools for visualizing quantitative data. The functions you can use in R are histogram(), dotPlot(), and boxplot(). Browse the help screen for these functions to get a sense of what these functions require.

Section 4 – Visualizing Data (two categorical)

Useful functions

With two categorical variables we can again create two-way tables using the tally() function. For example, try tally(~Habit | MomPriorCond, data = NCbirths, format = "proportion")

Section 5 – Visualizing Data (two quantitative)

Useful functions

To visualize a potential relationship between two quantitative variables, we typically use scatter plots. The syntax is plot(variable1 ~ variable2)

Also, for most plotting functions we can add some arguments to make the plots look more presentable.

- The *col* argument changes the color of the data.
- *cex* changes the size of the points
- *pch* should be an integer from 1 to 25 changes the style of the points
- *xlab* and *ylab* define the labels for the x and y axes
- *main* gives us the title for the plot

As an example, run and compare the following lines of code:

```
plot(NCbirths$weight ~ NCbirths$Gained)
```

```
plot(NCbirths$weight ~ NCbirths$Gained, col = "red", cex = 1.5, pch = 3,
    xlab = "Weight gained during pregnancy", ylab = "Baby weight (oz.)",
    main = "Baby weight vs. pregnancy weight gain")
```

Most of these arguments work for histograms, box plots, and dot plots too!