Lab 1: Getting Started with RStudio and Descriptive Statistics

Overall Goals

R is an open source tool for statistical computing and graphics, supported by the R Foundation for Statistical Computing. RStudio is a Graphical User Interface for R that we will be using from our servers. In this lab we will learn how to:

- start RStudio and familiarize ourselves with the different screen areas in RStudio
- load a built-in dataset and access its help file
- obtain a tabular view of a dataset and access basic information about the data from that view (number of cases, sorting by column, searching)
- create simple graphs and numerical summaries for variables
- assign values to names for future use

Specific R commands introduced: data, help, View, favstats , histogram, %>%, filter

Start RStudio and Get Your Bearings

To start RStudio from any computer on the campus network:

- Open your browser to rstudio.hanover.edu[1].
- Use your Hanover College login and password.
- After a short interval, you should be seeing the basic RStudio interface.

Notice the *Console* pane (left side of the application window). This is where you type commands and see results.

You interact with RStudio in two basic ways:

- type the command directly in the Console pane and hit <enter>.
- use the menus and buttons in various panes to perform some standard actions.

We have prepared a package that puts together a number of useful commands and datasets. You should **always start your RStudio session by typing the following in the Console**, in order to load this package. Type this into the console now:

```
library(hanoverbase)
```

Ignore the resulting output warnings.

In addition to the Console pane, the RStudio user interface has many other components:

- The *Environment* pane shows all the active objects in your current session.
- The *Files* pane shows all the files and folders in your default workspace.
- The *Plots* pane shows all your graphs. You can click the left-right arrows in the pane to scroll through the various plots you have created. You can also click the Zoom button for a larger view of the graph in its own window.
- The *Packages* pane lists packages or add-ons needed to run certain processes.

[1] http://rstudio.hanover.edu

- The *Help* pane gives you access to R documentation for additional info on a command. The information provided by some documentation pages may be somewhat overwhelming and confusing. The commands list page[2] on the website offers a more easily digestible amount of information, more suitable for beginners.
  Help tip: In the Console, use a prepended `?` to pull up the R documentation for any R command. For instance, typing `?print` in the console will bring up the documentation for the `print` command.

Working with the Console

The RStudio Console includes a variety of features intended to make working with R more productive and straightforward. Here we describe two of the most popular features.

Tab Completion

In the Console, you might begin typing a command and then be unsure if you're spelling it correctly. If you pause for a second, RStudio will show you a list of all commands which begin in that way. You can press <esc> to cancel that list of options, and you can use the arrows keys and hit <tab> to choose a completion.

Command History

As you work with R, you'll often want to re-execute a command which you previously entered. The RStudio Console supports the ability to recall previous commands using the arrow keys. You can scroll through the history in both directions by using the up/down keys while in the Console pane. A common workflow is to incrementally build a task by starting with a simple command, then using the up arrow to recall it and extend it by adding more steps.

You can see a list of all previously entered commands in the History pane, situated at the upper right of the window. Double-clicking any of these commands will paste it into the Console.

Basic R Syntax and Commands

Scientific Notation

To specify a large number such as 3.2 million, we generally use scientific notation. The multiplier goes first, then an `e` to separate the multipler from the exponent, then the actual exponent (desired power of 10). For example, since 2 million is 3.2×10^6, we need `3.2e6` to express the idea of 3.2 million.

Scientific notation is also helpful for indicating small values, such as 0.007 (7 thousands), using negative exponents: `7e-3`. Try these for yourself in the console.

[2] ../commands.html

Variable Assignments

Variable assignments allow us to store the results of complicated commands so they can easily be recalled in the future. To assign a value to a name (or "variable"), use <– or =. When you make an assignment, you do not see a result right away, but a value has been stored for future use.

Try the following and observe the results:

```
x <- 5
y <- 3
x + y
x <- 10
z <- x * y
z
```

If you have written the above correctly, the *Environment* pane will show the current values of x, y and z.

The Piping Command

You will see the "pipe" operator, %>%, in some of the commands below. The pipe operator takes the left-hand side (LHS) of the pipe and uses it as the first argument of the function on the right-hand side (RHS) of the pipe. As a very basic example, 1:10 is the sequence of integers from 1 to 10 inclusive. What is the sum of this sequence? What is the median?

TRY THIS:

```
# Everything in a line following a pound sign (#) is a comment (R ignores it).
# You don't need to type in the comments below.
# We use comments to explain the meaning of the commands.
1:10                # the sequence of all integers from 1 to 10
sum(1:10)           # sum of the sequence 1 to 10
1:10 %>% sum()      # the same thing, using a pipe
median(1:10)        # median of the sequence 1 to 10
1:10 %>% median()   # the same thing, using a pipe
```

Detecting Typing Errors in the Console

Once the hanoverbase packase is loaded with the library(hanoverbase) command, any red messages in the console window indicate errors. The most common error you may make is a "typo". For example, you meant to type View but it came out as view or Vview. TRY THIS and notice the resulting error messages:

```
summmm(1:10)
mEdian(1:5)     # notice how the case (lower/upper) matters!
```

Another type of error, which will **not** give you a red error message, is entering a command which is incomplete. For example, if you enter sum(1:10, and press <enter> without the closing parenthesis, R responds with a plus sign (+) reminding you to finish up the command. Your console will look like this:

```
> sum(1:10
+
```

Notice that the "prompt" changed from the usual > to a plus sign +. R is waiting for you to *continue* the previous line to properly finish the command. Just type the closing parenthesis and hit <enter> (don't repeat the previous line). Alternatively, you can hit the <escape> key to cancel the last command and try again.

Data Investigations

Load Data and Start your Investigation

The U.S. 2010 Census generated a wealth of data. We will see later how to download data from the web and other sources. For now, we will use a built-in dataset, containing information about the U.S. counties. (Always View the data when you load it. Notice that View starts with a *capital* V.)

To load the data set, type:

```
data(counties)
View(counties)
```

If this worked correctly, you now see the data view in the upper-left portion of the application. **Stop and figure out the problem if this is not the case.**

In the Environment pane (upper right), you should now see a counties entry under Data. It should tell you the number of observations (rows) and the number of variables (columns) in the data.

You should also bring up the documentation page for the dataset by running ?counties or help(counties) in the Console.

You can use the search box at the top right of the data view to filter the rows. For instance typing Indiana will show only the rows that have the word Indiana in one of their fields.

Notice that you can **sort** the data file according to any column by clicking the column heading. Click a second time to sort in the opposite direction. For example, try this out (using the pop2010 column) to discover which U.S. county has the smallest population (2010) and which has the largest.

Don't forget that you can use scrolling and sorting to answer some of the following questions.

1. How many counties are there in the U.S.?

2. Name the top 3 counties by population (2010); give the county name, the state name and the 2010 population. For each of these three counties, explain why the county has such a large population. (Use internet search as needed.)

3. List the 3 counties with the least population (2010), including their populations. Use the internet to find an interesting fact about each county.

Histogram and Summary Statistics, One Quantitative Variable

The most popular graph for showing the distribution of a single quantitative variable is the histogram. The following will draw the default histogram for pop2010:

```
histogram(~pop2010, data=counties)
```

You may need to wait several seconds for the results. You can get a bigger version of the graph by clicking the Zoom button in the *Plots* pane.

We can enhance our understanding of the distribution by producing summary statistics. The command favstats will give us the five-number summary and the mean for the distribution:

```
favstats(~pop2010, data=counties)
```

4. What are the mean and median populations for counties? Which one is larger? Does that make sense based on the shape of the distribution?

5. Even though the precise shape of the distribution cannot be fully seen in this histogram, we can definitely note that the distribution is extremely skewed to the right. Explain how this makes sense given what you know about regional population distribution in the U.S.

In the above histogram, almost all of the counties are thrown into a single bin, and not much can be said about the distribution (e.g. number and location of modes). We can filter the data to exclude counties with large population (e.g. require pop2010 <= 2e6). We can use the "piping" command for this to pipe the counties through a filter . The command for this is:

```
histogram(~pop2010, data=counties %>% filter(pop2010 <= 2e6))
```

6. This histogram is still lacking in detail, and we should filter out even more values. Adjust the histogram command to use a lower cutoff point than 2 million. Use your judgment to find a cutoff so that the main pattern of the distribution is clearly shown. How many modes does the distribution have and what are their approximate values (these modes represent "typical" county populations)?

7. The following command shows the percent of residents of each county that are foreign-born, broken down by state. You can enter the whole thing in a single line, or create line breaks with the <enter> key.

```
histogram(~foreign_born|state,
    data=counties %>%
        filter(state %in% c("California", "Indiana", "West Virginia")),
    layout=c(1,3,1), breaks=20)
```

   What are three things we can learn about these states and their counties by looking at this graph?

To end your RStudio session, simply click the red button at the top right and close the window. If you're leaving the lab, log out of Windows.