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.

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.

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

- The *Packages* pane lists packages or add-ons needed to run certain processes.
- 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.

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.

```
library(hanoverbase)
```

Ignore the resulting output warnings.

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.

[2] ../commands.html

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 1 million is `10^6`, we need `e6` to express the idea of 1 million.

Scientific notation is also helpful for indicating small values, such as 0.000007 (7 millionths), using negative exponents.

Examples (try these yourself):

```
-8.7e9   #    -8.7 billion
1e-1     #     0.1
7e-6     #     0.000007
```

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 will show the current values of `x`, `y` and `z`.

Formulas

We will use "formulas" in our graphing commands, to specify the variable(s) to display and any conditioning variables. Here are some examples (these will make more sense in future sections):

- to display variable x alone we would use the formula: `~x`
- to display the numeric variable x for each level of factor A: `~x | A`
- to display the relationship between numeric variables y and x (treating y as the response): `y~x`

More explanation of formulas can be found at the commands list page[3].

[3] ../commands.md#the-formula-interface

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:

```
1:10                  # the sequence 1 to 10
1:10 %>% sum()        # sum of the sequence 1 to 10
sum(1:10)             # The same thing
1:10 %>% median()     # median of the sequence 1 to 10
```

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.

To load the data set, type:

```
data(counties)
```

In the Environment pane (upper right), you should now see a counties entry under Data. Click on that entry to view the data in its own pane. This the same as using the View command in the Console:

```
View(counties)
```

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 headings (click again to change the sort direction).

Warning: The NA (not available) answers will always sort at the bottom, whether you sort ascending or descending.

Use scrolling and sorting to answer 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 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.

```
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?