ST 411/511 Lab 1
Introduction to R and RStudio

Instructions: Work through the following activity, either on your own, or along with the lab TA. Do the computational parts of Homework 1 shortly after finishing the lab activity.

Objectives for this Lab

- Introduce RStudio
- Install and load R packages.
- Produce some simple graphs (histogram, boxplot, dotplot).
- Calculate sample statistics (mean, median, standard deviation, etc.).
- Perform a t-test.
- (optional) Knit an R Markdown document.

1. If you are doing the lab on your own computer, install R and RStudio before you start. Instructions are here. If you prefer not to install R and RStudio on your computer, you can access them on the Virtual Computer Lab. Instructions are here. Another option is RStudio Cloud (`https://rstudio.cloud/`). Both the Virtual Computer Lab and RStudio Cloud require an internet connection.

2. Download the file Lab1.R from the course Canvas site (Files>Lab Materials). This file contains all the R commands used in this lab. Note the directory to which you download it.

3. Start up RStudio. RStudio is a workspace environment for R.

4. The default RStudio desktop layout shows three panes. Clockwise from left, they are

   - Console This is where RStudio communicates with R. You should see some start up information and a "prompt" `>`.
   - Environment/History
     - *Environment* lists all the variables in R's workspace.
     - *History* keeps track of all the R commands issued in the command window.
   - Files/Plots/Packages/Help/Viewer
     - *Files* shows the contents of the current directory.
     - *Plots* will display the plots and graphs as you generate them.
     - *Packages* lists the R packages installed and allows you to install new packages.
     - *Viewer* allows you to view local web content. We'll probably leave this one alone.

5. Open Lab1.R. In RStudio, select File>Open File... Navigate to the directory where you saved the file and click Open. You should now have a source editor pane open at the top left, displaying Lab1.R.

6. Load the `Sleuth3` package so you have access to the data sets, and load the `ggplot2` package so you can use the `qplot()` command in item 7(a) below to make graphs.

(a) You'll need to install the packages, unless they're already installed on the computer you're using.

(i) From the Packages pane, click the Install button to open the Install Packages dialog box.

(ii) Type "Sleuth3, ggplot2" in the Packages line (case sensitive, "sleuth3" or "GGplot3" won't work). Click "Install" at the bottom of the dialog box.

(iii) If you're running RStudio on the Virtual Computer Lab, the above won't work. If you need to install packages, you'll do this by using the `install.packages()` command in the Console window. Here are the commands, but you don't need to use them for these two packages because they should already be installed.

```
> install.packages("Sleuth3")
> install.packages("ggplot2")
```

(b) Load the Sleuth3 package into the R session's library. You'll have to do this every time you start up RStudio if you want to use the data in the *Sleuth's* case studies and exercises.

```
> library(Sleuth3) # Load the Sleuth3 package
> library(ggplot2) # and the ggplot2 package
```

To run this command, you can either type it after the prompt (`>`) in the Console pane, or place your cursor anywhere in the first line of Lab1.r and click the Run button at the top right of the source editor pane.

All the text in a line after a `#` symbol is called a "comment" and ignored by R.

Note: You'll have to "load" the package to the library again next time you start up RStudio, but once it's "installed" you don't have to do that again. Installing a package downloads it to your computer. Loading it in R tells the current R session about the functions and data it contains.

(c) To get information about the Sleuth3 package, use the `help()` function.

```
> help(package=Sleuth3)
```

We will work with the salary data of case study 1.1.2. This is `case0102` in the Sleuth3 package. In the help window, click on the case0102 link for information and references about this data set. The case study data sets in the Sleuth3 package are named "case####" where #### refers to the chapter and case study number. Data for the exercises are named "ex####".

(d) You can look at the data by typing the name of the data frame:

```
> case0102
```

The Console pane lists the contents of the "data frame," R's term for how it represents a table of data. The data frame contains two columns called `Salary` and `Sex`. The 93 rows are numbered sequentially.

(e) Alternatively, you can open the data frame in a separate tab in the Source window using `View()`.

```
> View(case0102)
```

7. Draw histograms of the salary data.

   (a) Produce a histogram of all salaries. We will use `qplot()` ("quickplot") from the `ggplot2` package, so load that package into the library.

   ```
   > library(ggplot2)
   > qplot(case0102$Salary, geom="histogram")
   ```

   The command `qplot()` creates a plot using the data in its first "argument." The `geom="histogram"` tells R we want a histogram. RStudio displays plots in the lower right-hand pane.

   The notation `case0102$Salary` refers to the column "Salary" in the data frame "case0102."

   (b) It can be tedious to type the data frame name each time you want to refer to a column when issuing a single R command. The `with()` function provides a shortcut:

   ```
   > with(case0102, qplot(Salary, geom="histogram"))
   ```

   You should get the same histogram as before, except the title and horizontal axis label omits the data frame name.

   (c) With `qplot()` and many but not all R functions, you can use the `data=` argument to alert R to the data frame name.

   ```
   > qplot(Salary, data=case0102, geom="histogram")
   ```

   (d) To make separate histograms for males and females as in Display 1.4, we'll need to extract the observations from males, and draw a histogram, then repeat for females. R complains if we use the `data=` argument here because the number of male salaries is different than the length of the data frame, so we will use `with()`.

   ```
   > with(case0102, qplot(Salary[Sex=="Male"], geom="histogram"))
   > with(case0102, qplot(Salary[Sex=="Female"], geom="histogram"))
   ```

   The syntax `Salary[Sex=="Male"]` extracts the salaries from the rows where Sex is "Male." The double "==" is not a typo. The above code produces two graphs. You can use the back arrow in the Plots pane to see the previous graph.

   These histograms don't look like the ones in Display 1.4 because the breakpoints are different. You can specify the breakpoints with an optional argument to `qplot()`.

   ```
   > with(case0102, qplot(Salary[Sex=="Female"],
   +                       geom="histogram",
   +                       breaks=c(3800,4200,4600,5000,5400,5800,6200,6600)))
   ```

   The `c()` command *concatenates* its arguments into a single unit. Setting `breaks=c(3800,...)` tells R the specific breakpoints to use. You can make several other custom adjustments to a `qplot()` plot. For details, check the help documentation.

   ```
   > help(qplot)
   ```

   R documentation takes some getting used to.

8. Histograms are useful for seeing the shape of the distribution of data, but they are highly dependent on the breakpoints. *Boxplots* are less subjective.

   (a) Use `qplot()` to produce side-by-side boxplots of the salary data as in Display 1.12.

   ```
   > qplot(Sex, Salary, data=case0102, geom="boxplot")
   ```

   The first two arguments to `qplot()` are the grouping variable and the quantitative variable, respectively. Next are the `data=` and `geom=` arguments. Here the geometry is "boxplot." Note that the order of the first two arguments matters, but that the second two can come in either order. For example, the following produces the exact same plot

   ```
   > qplot(Sex, Salary, geom="boxplot", data=case0102)
   ```

   whereas the following produces a rotated plot.

   ```
   > qplot(Salary, Sex, data=case0102, geom="boxplot")
   ```

   (b) You will usually have two or more choices for doing the same thing in R. For example, you can also use `ggplot()` to make a boxplot. The `ggplot()` function has a lot more flexibility than `qplot()`, but it isn't as...quick. Here's a way to make the side-by-side boxplots using `ggplot()`.

   ```
   > ggplot(case0102, aes(x=Sex,y=Salary)) + geom_boxplot()
   ```

   The "gg" in `ggplot()` stands for "grammar of graphics." It's a novel way of making sophisticated, informative graphics. For ST 411/511, `qplot()` will generally be sufficient, but for the moment, check out what this small addition to the command does:

   ```
   > ggplot(case0102, aes(x=Sex,y=Salary)) + geom_boxplot(aes(fill=Sex))
   ```

   We can add a title.

   ```
   > ggplot(case0102, aes(x=Sex,y=Salary)) + geom_boxplot(aes(fill=Sex)) +
   +             ggtitle("Starting Salaries")
   ```

   The three plus symbols have two different functions. The code in Lab1.r has only the plus symbols before and after `geom_boxplot(aes(fill=Sex))`. These tell R that we're adding elements to the plot, and they **must not** come at the beginning of a line. The plus symbol directly below the > prompt is a *continuation* prompt. R is telling us that the line of code continues onto another line.

   (c) Finally, plain R has a `boxplot()` function that `ggplot()` has largely replaced. Here's how to make side-by-side boxplots with `boxplot()`:

   ```
   > with(case0102, boxplot(Salary~Sex))
   ```

   The variable on the left-hand side of the $\sim$ is read as a dependent ("response") variable, whereas the variable on the right-hand side is an independent variable. In this case, the independent variable is a categorical variable which serves to group the response into separate categories.

9. You can include an R graphic in a Word or LaTeX document. To do this, click the Export menu in the Plots pane. You can save as a PDF, copy to the clipboard, or save as an image. The easiest way to insert into Word is to copy to the clipboard, then paste into an open Word document. Or see the optional item 14. below for an introduction to R Markdown.

10. A stem-and-leaf diagram shows both the distribution of the data and the actual numbers. R's `stem()` function produces these. It won't do back-to-back diagrams as in Display 1.10, so we'll have to do separate commands. Here's the command for the male salaries:

    ```
    > with(case0102, stem(Salary[Sex=="Male"]))
    ```

    Note that the diagram appears in the Console pane, not the Plots pane.

11. To get summary statistics about a set of data, use `summary()`.

    ```
    > with(case0102, summary(Salary[Sex=="Female"]))
    ```

    Many R commands behave differently depending on what type of argument you give them. `summary()` is one of those. The argument `Salary[Sex=="Female"]` is *numeric*. On the other hand, `Sex` is a "factor" variable, which is R's name for a categorical variable. You can verify this using `is.factor()`.

    ```
    > with(case0102, is.factor(Sex))
    ```

    If you ask for a summary of a factor variable, you'll get a display of the unique values ("levels") of the factor and the counts in each group.

    ```
    > with(case0102, summary(Sex))
    ```

12. The summary statistics of a numeric variable include the minimum, maximum, median, mean, and the 1st and 3rd quartiles, but *not* the sample standard deviation. R's `sd()` function will calculate this for you:

    ```
    > with(case0102, sd(Salary[Sex=="Female"]))
    ```

13. Perform the one-sided two-sample t-test reported in the Summary of Statistical Findings at the bottom of page 4 of the textbook. We will spend more time on the t-test in Chapter 2, but this should not be the first t-test you've ever done.

    ```
    > t.test(Salary~Sex, alternative="less", data=case0102, var.equal=TRUE)
    ```

    There's that same "formula" that we used in item 7(d). Setting `alternative="less"` tells R that the alternative hypothesis is that the female salaries are less than the male salaries. The ordering is usually alphabetic, but you can check the output of `summary(Sex)` in 11. to see that "Female" is indeed listed first. The `var.equal=TRUE` is needed to do the t-test based on the assumption that the two populations have equal variance. We will revisit this issue in Chapters 3 and 4.

    Find the p-value and the confidence interval.

14. (Optional) Produce a pdf document using R Markdown and knitr. R Markdown is a slick way to combine R code, graphs, equations, and prose into a single pdf or html document.

    (a) Open a new R Markdown document from the File menu: File>New File>R Markdown...

    RStudio will probably tell you it needs to update or install several packages. Once that's done, you'll see the New R Markdown dialog box.

    (b) Fill in a title and author (or not), and select PDF as the default output, then click OK. This will create an example R Markdown document displayed in the Source window.

    (c) If you don't already have LaTeX (MiKTeX for Windows; MacTeX for Mac), install TinyTex. LaTeX is a typesetting system with the ability to produce nice-looking mathematical formulas. The Virtual Computer Lab does not have LaTeX.

    ```
    > tinytex::install_tinytex()
    ```

    During the installation process, you'll get several error messages. Ignore them. Just click OK, and be patient. After the installation process completes, it will tell you to quit and reopen your R session and IDE. You can ignore this too.

    (d) "Knit" the example R Markdown document by clicking the Knit icon on the Source window's toolbar. RStudio will ask you to save the document, so you should do that. The appropriate file extension for R Markdown files is .rmd.

    (e) After you save the file, R will process it into a pdf, then display the result.

15. Close RStudio and quit R.

    (a) Save any files that you've edited. These files contain the commands needed to reproduce your work. The File menu allows you to save a file.

    (b) Select Quit RStudio... from the File menu to close RStudio. This action automatically quits R as well. You generally don't need to save the workspace.