Recitation 1: Introduction to R

1 What is R?

R is an open-source and free software package for data analysis and graphical representation. It is very flexible and highly customizable, and its excellent graphical tools make R an ideal environment for data analysis. Although the learning curve is somewhat steep compared to "point and click" software (such as Excel), in many cases, R allows users to quickly get results without having to write a complex program.

2 Downloading and Installing R

R (and RStudio) have already been installed on computers in the recitation rooms. If you prefer to work with your own computer:

- To install the basic R environment, visit `http://cran.r-project.org`. Select your operating system, and only install `base`.
- While the installation comes with a base IDE, we'll be using the more polished and user-friendly RStudio IDE in recitations this semester. To install RStudio, visit `https://www.rstudio.com/products/rstudio/download/`, and download the version appropriate to your operating system.

In either case, finish by opening RStudio.

3 Working with RStudio

RStudio's interface is divided into four main components (for a screenshot, visit `https://www.rstudio.com/wp-content/uploads/2014/04/rstudio-windows.png`):

- The **console window** (bottom left in the screenshot), in which you can type commands and view output
- The **workspace tab** (or **environment tab**, top right), which displays all objects (vectors, variables, data frames) that have been created. This can be toggled with the **history tab**, which displays the last few commands executed in the console.
- The **files tab** (bottom right) displays all files and folders in your current working directory. We'll more frequently work with the **plots** tab, which shows any graphs that you create via the console.
- The **script window** (top left) can be used to execute a sequence of commands simultaneously in the console. Scripts written there can also be saved as `.R` files.

4 The Console Window

Essentially, a very sophisticated calculator. What you type is what you get, as long as it's valid R code. Type commands directly into the console, and execute them using `Enter`.

1. To get a feel for how to use the console, try the following commands
   - `5 + 7`
   - `sqrt(16)*8`
   - `3/log(3)`
   - `2^log10(100)`
   - `x <- 7` (or `x = 7`)

   The last example creates a variable `x`, and assigns it the value 7. Variables are examples of R *objects*.

2. We'll be working with *vector* objects in R this semester. (Think of a vector as an array, or as a collection of numbers, strings, or booleans.) To learn how to work with vectors in R, try the following. You don't need to turn anything in, but it may help to make a note what each command does.
   - `x <- c(1, 3, 2, 10, 5)`
   - `y <- 1:5`
   - `x[1]`
   - `y[2:4]`
   - `x[x>3]`
   - `y`
   - `y + 2`
   - `3*y`
   - `y^2`
   - `y <- 2*y`
   - `y`
   - `a <- c("apple", "banana", "orange")`
   - `b <- c(TRUE, FALSE, TRUE)`

3. A few more examples of vector arithmetic, this time using built-in functions in R. If you're unsure as to what any of the following do, type, e.g., `help(sum)` or `?max`.
   - `x <- c(1, 3, 2, 10, 5); y <- 1:5` (multiple commands executed simultaneously, separated by semicolons)
   - `x + y`
   - `x*y`
   - `sum(x)`
   - `max(x)`
   - `min(x)`
   - `sort(x)`
   - `sort(x, decreasing=T)`
   - `length(x)`

4. We'll sometimes work with logical vectors in R (that is, vectors only containing the values `TRUE` or `FALSE`), as they make certain tasks simpler:

    - `x > 3`
    - `sum(x > 3)`
    - `(1:length(x))[x > 3]`

5 Data Frames

R is especially well-suited for manipulating datasets. These can be stored in *data frame* objects, which you can think of as tables (or as a collection of vectors with names). In this course, you'll typically import datasets into R from `.txt` or `.csv` files. We'll perform some basic manipulations here.

1. Download the file `marathon.csv` from Blackboard. The file contains the winning times for a particular annual marathon, for years ranging from 1971 to 1999.

2. Import your dataset using one of the following options:

    - Use RStudio's `Import Dataset` feature, which can be found near the top of the workspace (or environment) tab. Select `From Text(readr)`, and search for the file on your computer. Check that the dataset is formatted properly (e.g., that RStudio recognizes that the first row contains column names).
    - The long way:
        (a) Set your working directory to be the one containing `marathon.csv`. This can be done via the command `setwd(<pathname>)`, or by using `Session > Set Working Directory > Choose Directory`. If you use the former option, you may need to replace forward slashes (/) with backslashes (\). To see your current working directory, type the command `getwd()`.

        (b) Use one of the following lines to import the dataset into an R data frame:
            – `marathon <- read.table("marathon.csv", sep = ",", header=TRUE)`
            – `marathon <- read.csv("marathon.csv")`

            If you use the former, `sep = ","` specifies that entries within a row are separated by commas, while `header=TRUE` specifies that the first line of the file contains column names.

3. Display the dataset using `View(marathon)`.

4. For brief descriptions of the data frame, try:

    - `summary(marathon)`
    - `head(marathon)`, `head(marathon,10)`
    - `tail(marathon)`, `tail(marathon,10)`
    - `attributes(marathon)`

5. To access individual rows or columns of the data frame, type the following commands into the console window:

- `marathon$Year`
- `marathon["Year"]` (What is the difference between this and the first command?)
- `marathon[, 2]`
- `marathon[2, ]`
- `marathon[c(1,3,5),]`
- `marathon$Year[1]`

6. Construct a plot with `Year` on the x-axis, and `Time` on the y-axis. The syntax for this is `plot(x, y)`, where `x` and `y` are vectors (or, alternatively, columns of a data frame).
7. The above is an example of a *time series* plot. This can instead be constructed via `ts.plot(y)`. Try this.

6 The Script Window

As mentioned earlier, the script window can be used to create R scripts, so that sequences of commands can be executed without retyping them into the console one-by-one. Keep this in mind when working on questions in the next section.

1. If the script window is not already open, do so using `File > New File > R Script`.
2. Execute script files by highlighting the portion of the code you would like to run, and then pressing either `Ctrl + Enter` or `Cmd + Return`.
3. You can save your script for later use.

A few more bits of information about writing scripts:

- R is case-sensitive.
- Variables and file names must be alphanumeric, though exceptions include `.` and `-`.
- Lines of code can either be terminated with `;` or a line break.
- Begin comments with the `#` symbol.

7 A Second Example

Your installation of R comes with a large number of built-in datasets. We'll use `sleep`, which contains data gathered from a study on the efficacy of two types of sleeping drugs.

1. To view the dataset, simply type `sleep` into the console. (No importing necessary.)
2. Boxplots provide a convenient way to visualize data, and can be done using R's `boxplot` function. Specifically, given a vector `x`, `boxplot(x)` constructs a boxplot of the entries in `x`. Construct a boxplot of the increase in the number of hours of sleep resulting from either drug (i.e., the values in the column `extra`).

3. You can filter out rows and columns using commands such as the following:

   - `subset(sleep, group==2)`
   - `subset(sleep, group==2)[ ,1]`
   - `subset(sleep, group==2 & extra > 1.5)`
   - `subset(sleep, group==2 | extra > 1.5)`
   - `group2 <- subset(sleep, group==2)`
   - `group2`

   Separate data from each group data into two data frames, `group1` and `group2`.

4. To get an idea of how the two drugs compare, we'll construct side-by-side boxplots. Do this using `boxplot(group1$extra, group2$extra)`.

5. Histograms can be constructed using R's `hist` function. Construct one of the increase in the number of hours of sleep resulting from either drug.

6. If you want to see a list of R's pre-loaded data sets, you can do so using the `data()` command.

8 More Guidance

To learn more about R:

- Try the interative tutorial at
  `http://tryr.codeschool.com/`

- See the official R tutorial for more details:
  `http://cran.r-project.org/doc/manuals/R-intro.html`

- Check out some statistical applications of R:
  `http://www.r-tutor.com/`

- To create a nice report for homeworks, check out R MarkDown. It is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see
  `http://rmarkdown.rstudio.com/`. An example of putting some basic data analysis we have practiced so far into a pdf using R Markdown can be found on blackboard.

- R Markdown cheatsheet:
  `https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf`

Also remember to use R's `help()` function.