

The purpose of this handout is to help get you set up with R, the software we will use in this class. The first two sections will help you get the proper software installed on your computer. The remaining sections show some of the basics of R. The last section ("Background") is entirely optional, and is provided as an FYI. It gives a quick overview of some R facts.

1 RStudio Desktop and RStudio Web

There are two ways that you can use RStudio:

1. (recommended) Through the RStudio web server. This involves requesting an RStudio web account, which means you simply give me your Gmail address and I take care of the rest.
2. Through a local installation on your own computer. This involves installing both R and RStudio onto your computer.

I briefly describe each below.

1.1 RStudio Web

One of the main advantages of the web-based version is its portability. You can work in RStudio on one computer, then log off and go to a different computer. If you log into RStudio on the new computer, you will be able to resume your RStudio session and continue working. Moreover, you don't need to install any additional software on your computer.

If you are interested in the web-based version, you need to give me your Gmail address. I will then get you set-up with an account.

1.2 RStudio Desktop

Remember that to use a locally installed version of RStudio, you first need to install R before installing RStudio.

Installing R on your laptop

- Go to <http://www.r-project.org/>.
- In the menu to the left, under "Download," click on the link "CRAN."
- Select a mirror site (pick a site near New York, for faster download times).
- In the first boxed section titled "Download and Install R," click on the operating system for your computer.
 - *Windows*. Click on the link "base". Then click on the link "Download R 2.14.1 for Windows". Wait for download and then follow instructions that appear.
 - *Macintosh*. Click on the link "R-2.14.1.pkg". Wait for download. Click "Continue" or "Agree" at every step. No need to customize.

Installing RStudio on your laptop

Once you have installed R on your computer, technically it is ready for you to use (double-click on the R icon). However, we will access R via RStudio, the slightly more user-friendly environment for R.

To download: Go to <http://rstudio.org/> and click on the button "Download RStudio."

2 First, a few basics

- Orient yourself to the different windows in RStudio:
 - Console window. This is where you type commands.
 - Text editor window. This is where you can edit your script files, as well as \LaTeX and Sweave files.
 - Workspace window. This shows all objects that you have created. (More on 'objects' later.
 - History window. This shows a history of commands you've typed directly into the Console window.
 - Files window. File management window similar to Windows Explorer or Mac Finder.
 - Plots window. This is where plots that you create will appear.
 - Packages window. List of R packages. More on 'packages' later.
 - Help window. Self-explanatory.
- R works in a question-and-answer style. In general, you enter a command at the command prompt ('>') and press Enter, and R returns an answer: either the result you requested, an error/warning message, a + prompt which is a signal for you to complete your input, or a > prompt which means it did what you asked but didn't need to return you anything (e.g. see *Assignments and Naming Variables* later).
- *Comments*. Anything preceded by the pound ('#') sign is considered a comment, and will be ignored by R. This is useful if you want to write notes/reminders to yourself on why you typed a particular command (i.e. "documentation").
- *Recalling previous commands*. To recall previous commands, hit the up-arrow key.
- *Case-sensitive*. R is case-sensitive so that x is different from X.

3 R as a calculator

The simplest task of R is to act as a calculator. What happens when you type the following? Is this what you expected?

1. `3+2`

2. `sqrt(4)`

- Note: `sqrt()` is an example of a "function" and 2 is its "argument"

3. `sqrT(4)`

- Why did you get the response (an error message) you did? *Hint:* Compare this input with the one above and remember that R is case-sensitive.

4. Hit the up-arrow key once. What happens?

5. Hit the up-arrow key twice. What happens?

6. Keep hitting the up-arrow key...

7. `log(64)`

- Note: `log()` is another example of a "function" and 64 is its "argument"

8. `#log(64)`

9. `sqrt(-1)`

- Note: NaN stands for "Not a Number." Note that R will evaluate this expression for you, but also give you a warning message.

10. `3+`

- Note: This is an example of an *incomplete* input. When this happens, R gives you the `+ prompt`, telling you that it is waiting for you to complete the input. In this case, type something that will complete the input.

4 Creating a script file (very important)

Before we get too far along, let's pause and talk about script files. An R script file is simply a text file containing a series of commands to be executed by R. A good script file will generally also include comments that document the purpose of the commands.

One advantage of R is that it is command-based, which in turn allows us to keep a script file, which in turn facilitates reproducible research. Let's create a script file for this lab session:

1. File -> New -> R Script
2. Save the file (as `.txt` or `.R`; it doesn't matter which, but I usually use `.R` extensions) in a place you will remember, such as a folder dedicated to this class.
3. Make a header at the top of your script that includes: the purpose of your script, the author, the date the script was created, the date the script was last modified, the location where your script is saved, any data files that are used, and any other notes. (Note: a header is not a necessary part of your script, but should be included as good practice). Here is an *example* header:

```
#####
# R Tutorial
# Author: Ming-Wen An
# Date Created: 1/18/2012
```

```
# Date Last Modified: 1/18/2012
# File Location: ~/Dropbox/teaching/R
# Purpose: -Practice creating a script file
#          -Familiarize self with basic R commands
# Notes: [Include any additional notes here]
#####
```

4. Now you're ready to edit the script file. There are 2 ways of doing this:

- (a) Type the commands directly in the Console window, as you have been doing with the calculator commands above. When you are finished with your session, go to the History window, select all the commands that worked (i.e. didn't give error messages), and click on the "To Source" button. This will copy and paste those commands to your script file. Then be sure to edit your script file by adding comments for each of your commands.
- (b) Type the commands (and comments) in the script file as you go. In order to execute a command from the Source window, place your cursor on the line with the command to be executed, and type Command-Enter (on a Mac).

I usually go with the latter approach, but you can try either one and use whichever is more comfortable for you. It may be easier to start with the former approach, especially if you are new to using R.

5 Assignments and Naming Variables

You will find yourself wanting to repeatedly type the same value. To avoid having to do this, you can save the value under some name (technically, we're saving the value to an "object.") To do this, we can use the equal sign. For example, suppose for some reason we want to save the value 5 into the object `x`.

```
> x=5
```

Notice that nothing really exciting happens on our screen, but in the background, R has now stored the value of 5 into the object called `x`. Now type the following and record what happens:

1. `x`
2. `x+10`

Note on Naming Conventions. Here we chose `x` to be the name of our variable. However, we could have chosen almost any name that's a combination of letters, numbers, and the period (`.`). There are a few conditions on the name: no spaces, cannot start with a number nor a period followed by a number, and are case-sensitive (e.g. `X` is not the same as `x`). There are also some other "reserved" names such as `c`, `q`, `t`, `C`, `D`, `F`, `I`, `T` and others. Some of these letters should look familiar to you; that should clue you in on why they are reserved names and should not be names that you use for your variables.

6 Vectors

In statistics, we often don't deal with single numbers at a time. Rather, we deal with many numbers at a time. For example, we might be interested in the SAT Math scores for all applicants to Vassar. The collection of these scores can be represented as a *vector* of numbers. Type the following into R and record what happens:

1. `1:10`
2. `c(1,2,3,4,5,6,7,8,9,10)`
3. `c(2,4,5,2,12,4,2,3,1)`
4. `seq(1,10,by=1)`
5. `seq(1,10,by=2)`
6. `seq(1,10,length=2)`
7. Now, apply what you learned in "Assignments and Naming Variables" by creating a vector named `myvector` with the sequence of values 5, 6, 7, ..., 15.

Each number in the vector is called an "element" of the vector. Sometimes we are only interested in a subset of the elements. We can view this subset using the `[]` (subset) command. Type each of the following and record what happens:

1. `myvector[1]`
2. `myvector[-1]`
3. `myvector[c(1,4)]`
4. `myvector[2:5]`

We can do calculations with the vector. Type each of the following and record what happens:

1. `sum(myvector)`
2. `mean(myvector)`
3. `sd(myvector)`
4. `summary(myvector)`
5. `length(myvector)`
6. `sort(c(2,5,3,20,19,2,1,10))`

7 Other Data Structures

We've seen how to work with single scalar numbers and vectors. You might be asking yourself, What about matrices and other data structures? Now is a good time to pause and consider this question. R stores everything as "objects." Each object is of a particular structure, e.g. vector, matrix, factor, list, and data frame. Briefly,

- A *vector* is a string of either numeric or character elements. A single scalar number is a vector of length 1. Examples of vectors:
 - `1:9`
 - `c("Math", "Economics", "Biology", "Psychology")`
- A *matrix* is an $n \times m$ array of either numeric or character elements. We usually work with matrices of numeric elements. Examples:
 1. `matrix(1:9, nrow = 3, byrow = T)`
 2. `matrix(1:9, nrow = 3, byrow = F)`
 - Compare this with the previous command. Note how specifying `byrow=T` or `byrow=F` affects the matrix.
 3. `matrix(c("A", "B", "C", "D", "E", "F"), nrow = 3)`
- A *factor* serves as a convenient way to work with categorical variables. More on this later.
- A *data frame* is a matrix-like structure in which the columns can be of different types (e.g. numerical, character, and categorical/factors).

8 Reading in data

Often our data come in the form of a spreadsheet (e.g. Excel/.xls, comma-separated values/.csv, text/.txt). Fortunately R can accept a variety of data file types. In general, try to avoid working with Excel files. You can always "save as" an Excel file as a .csv file. In R, there is a series of commands for reading in data from different file types. These are of the form `read.XX` where XX is replaced by the file type:

- `read.csv()`
- `read.table()`
- In the Help window, file the help file for `read.csv()`. You'll see documentation there for the other `read.XX` commands.

The dataset we'll work with is available at <http://mathserver.vassar.edu/Faculty/An/data/Movies.csv>, which is from DeVeaux, Velleman, and Bock *Intro Stats*. Let's read the data into R, saving it under the object name `data`.

```
> data = read.csv("http://mathserver.vassar.edu/Faculty/An/data/Movies.csv")
```

Note, in some cases, you might have the data file stored on your own computer (instead of on the web). Then in place of the URL address, you would specify the file location in quotes, e.g.

```
data=read.csv("~/Dropbox/teaching/data/Movies.csv")
```

Or you can complete it in 2 steps; first step is to *set your working directory* and the second step is to read in the data file:

```
setwd("~/Dropbox/teaching/data/Movies.csv")  
data=read.csv("Movies.csv")
```

9 Looking at your data

The first step for any analysis is to LOOK AT YOUR DATA. There are many ways to do this in R. Type the following commands and record what happens:

1. `data`
2. `head(data)`
3. `summary(data)`
4. `str(data)`
5. `names(data)`
6. `nrow(data)`
7. `ncol(data)`
8. `dim(data)`

We can request summaries of particular variables in the dataset by using the `$` syntax. In general, we use the form *Name of your data object*`$`*Name of your variable*. For example, record what happens when you type:

```
mean(data$Runtime)
```

We can also look the distribution of movie run times by creating a histogram:

```
> hist(data$Runtime)
```

This should open a pop-up window with a histogram. More on this later. One advantage of R is that it can produce very nice-looking graphs. The default graph create here isn't so exciting, but with a little more experience with R, we can change that.

10 Saving graphs

One way to save a graph (to include in a report, for example):

1. In the Plots window, use the Left/Right arrow icons to browse through the plots until the graph you wish to save appears.
2. Click on Export and select the option you wish to use.
 - If you're preparing a report in Word, you may want to "copy to clipboard" and then paste the graph into your Word file.
 - If you're preparing a report in \LaTeX , then you may want to choose "Save as PDF".
 - If you wish to keep a file of your graph, then choose either "Save as PDF" or "Save as image."

Another way to save graphs is by enclosing your commands for plotting with the "postscript" or "pdf" command. This is a more automatic way as it avoids the pointing & clicking required in the first approach, and as such is better for reproducible research.

```
> postscript(file = "myhistogram.ps")
> hist(data$Runtime)
> dev.off()
```

You can also save to a PDF file using

```
> pdf(file = "myhistogram.pdf")
> hist(data$Runtime)
> dev.off()
```

11 Explore!

This is only the tip of the R iceberg. Don't be afraid to explore and be creative; build upon these basics. Read the help files or other resources listed below to find out more about particular commands. This way, you'll grow in confidence and comfort with R.

12 Help! Where can I get help for R?

- To obtain help on any of the commands, you can either access R-Help from the menu or by typing at the command prompt:

```
> ?hist
```

This will bring up a separate help window on the command you typed.

- If you do not know the exact name of the command, you can do this instead:

```
> help.search("histogram")
```


- Google your question. There are a *lot* of R resources on the web. *Hint:* Include the letter “R” in your key word searches.
- R Forum on Moodle. Use it! Post neat tricks you discover. Copy and paste trouble-some code *and the accompanying error messages* in their entirety. Offer your solutions to these errors. Learn from others’ coding problems.
- Handouts. I will also provide handouts throughout the semester as needed.
- R Office Hour. Though you can ask me R questions in any office hour, I will designate one office hour each week to be an “R Office Hour” and take R-related questions.
- Me. E-mail me with specific details of what you wish to execute in R and I will help you figure out the necessary commands.
- *Introduction to Statistics with R*, by Dalgaard.
Note: This is available electronically for free via the Vassar Library’s subscription to the Springer e-books.
- The World Wide Web
 - *An Introduction to R* available at <http://cran.r-project.org/doc/manuals/R-intro.html>.
 - * FYI: There are other manuals (some on more advanced topics) located at: <http://www.r-project.org/>. Click on the Manuals link on the left menu bar, for a list of R manuals.
 - <http://pj.freefaculty.org/R/Rtips.html>
 - <http://cran.r-project.org/doc/contrib/usingR.pdf>

Hint: In RStudio (desktop and web versions), if you start typing the first few letters of a command and forget the remainder, hit the Tab key. A list of possible completions plus the first few lines of the help file will appear.

13 Background (Optional FYI)

Background (Optional FYI)

Some S facts

- S is a language and system for organizing, visualizing, and analyzing data.
- S has been a project of statistics research at Bell Labs since 1976.
- The language has evolved through several major versions to become the most widely used environment for research in data analysis and statistics.
- In 1998, S became the first statistical system to receive the Software System Award, the top software award from the ACM.

Products and projects based on S

- The S-Plus language is based on the S software from Bell Labs. S-Plus products are distributed by the Insightful Corporation, which has an exclusive license to distribute software based on Bell Labs' S.
- The R language is an open-source system distributed under the GPL license. It is a separate project based on the S language, with a number of differences to Splus. **R is what we will be using in this class.**

Some R facts

- R is an environment for data analysis and visualization.
- R is an open source implementation of the S language (S-Plus is a commercial implementation of the S language).
- The current version of R (December 2011) is 2.14.1.

The R software

- R is mainly written in C.
- R is available for many platforms:
 - Unix of many flavors, including Linux, Solaris, FreeBSD, AIX.
 - Windows 95 and later.
 - MacOS X.
- Binaries and source code are available from www.r-project.org.
- R "talks" to data bases, programming languages, and other statistical packages.
- R should be source code compatible with most of the Splus code written.