Tutorial 1: Introduction to R

Johannes Karreth

RPOS 517, Day 1

Instructions: Read pp. 1–8 at home and work through pp. 9–13 in our first class meeting.

In RPOS 517, we will primarily rely on R, a highly versatile open source software package. This tutorial shows you:

- how to install R and RStudio
- how to use RStudio to let R do work for you
- how to create a data analysis notebook in RStudio

Software installation

The most recent version of R for all operating systems is always located at `http://www.r-project.org/index.html`. Go directly to `http://lib.stat.cmu.edu/R/CRAN/`, and download the R version for your operating system. Then, install R.

To operate R, we will rely on writing R scripts. We will write these scripts in RStudio. Download RStudio from `http://www.rstudio.org`. Then, install it on your computer.

Lastly, install LaTeX in order to compile PDF files from within RStudio. To do this, follow the instructions under `http://www.jkarreth.net/latex.html`, "Installation". You won't have to use LaTeX directly or learn how to write LaTeX code in this class.

At-home assignment: Install R, RStudio, and LaTeX, then continue reading this tutorial.

Opening RStudio

Upon opening the first time, RStudio will look like Figure 1.

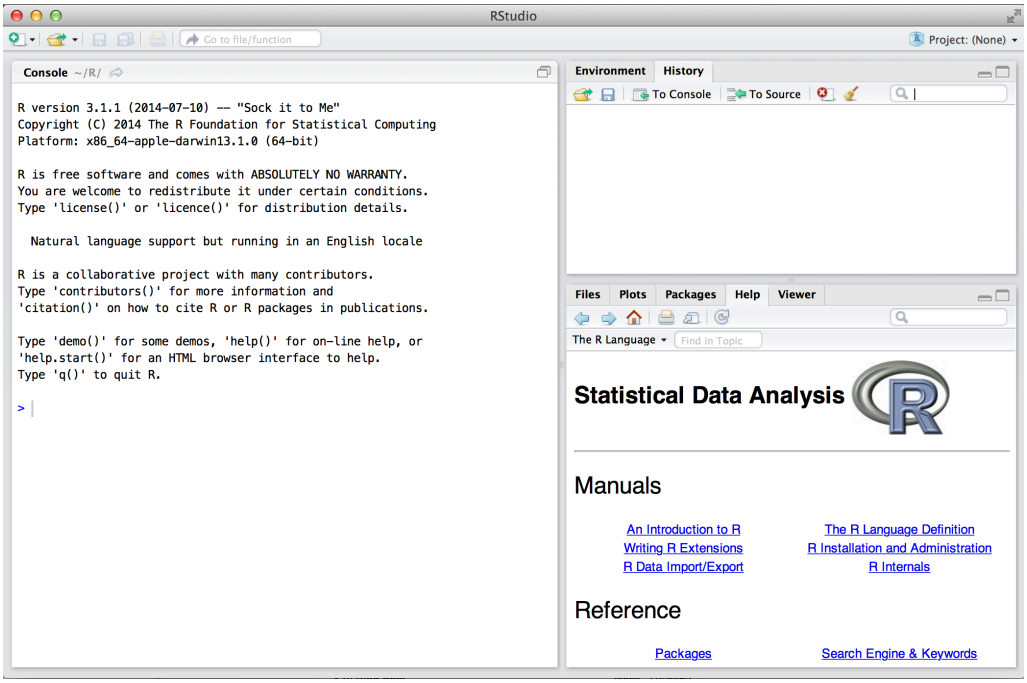

Figure 1: RStudio.

The window on the left is named "Console". The point next to the blue "larger than" sign > is the "command line". You can tell R to perform actions by typing commands into this command line. We will rarely do this and operate R through script files instead.

Typing R commands

In the following pages, I walk you through some basic R commands. Grey boxes show these commands, followed by R's response.

To begin, though, see how R responds to commands. Type a simple mathematical operation:

You do not need to type anything yourself at this point; just read the following pages until you get to the first assignment.

```
1 + 1
```

```
## [1] 2
```

```
2 * 3
```

```
## [1] 6
```

```
10 / 3
```

```
## [1] 3.333333
```

Error messages

R will return error messages when a command is incorrect or when it cannot execute a command. Often, these error messages are informative by themselves. You can often get more information by simply searching for an error message on the web. Here, I try to add 1 and the letter a, which does not (yet) make sense:

```
1 + a
```

```
## Error in eval(expr, envir, enclos): object 'a' not found
```

As your coding will become more complex, you may forget to complete a particular command. For example, here I want to add 1 and the product of 2 and 4. But I forget to close the parentheses around the product:

```
1 + (2 * 4
)
```

```
## [1] 9
```

You will notice that the little > on the left changes into a +. This means that R is offering you a new line to finish the original command. If I type a right parenthesis, R returns the result of my operation.

R as an object-based language

R is an object-based programming language. This means that you, the user, create objects and work with them. Objects can be different types. To create an object, first type the object name, then the "assignment character", a leftward arrow <–, then the content of an object. To display an object, simply type the object's name.

The following are the types of objects you need to be familiar with:

A good overview of objects is here: `http://www.statmethods.net/input/datatypes.html`. Read this page and then continue this tutorial.

- Vectors
  - Numeric (numbers)
  - Character (words or letters)
  - Logical (TRUE or FALSE)
- Matrix
- Data frames
- Lists

Below, you find some more specific examples of different types of objects.

- Numbers (called scalars):

```
x <- 1
x

## [1] 1

y <- 2
x + y

## [1] 3

x * y

## [1] 2

x / y

## [1] 0.5

y^2

## [1] 4

log(x)

## [1] 0

exp(x)

## [1] 2.718282
```

- Vectors (a chain of scalars):

```
xvec <- c(1, 2, 3, 4, 5)
xvec

## [1] 1 2 3 4 5

xvec2 <- seq(from = 1, to = 5, by = 1)
xvec2
```

```
## [1] 1 2 3 4 5

yvec <- rep(1, 5)
yvec

## [1] 1 1 1 1 1

zvec <- xvec + yvec
zvec

## [1] 2 3 4 5 6
```

- Matrices (a group of numbers):

```
mat1 <- matrix(data = c(1, 2, 3, 4, 5, 6), nrow = 3, byrow = TRUE)
mat1

##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
## [3,]    5    6

mat2 <- matrix(data = seq(from = 6, to = 3.5, by = -0.5),
               nrow = 2, byrow = T)
mat2

##      [,1] [,2] [,3]
## [1,]  6.0  5.5  5.0
## [2,]  4.5  4.0  3.5

mat1 %*% mat2

##      [,1] [,2] [,3]
## [1,]   15 13.5   12
## [2,]   36 32.5   29
## [3,]   57 51.5   46
```

- Data frames (equivalent to data sets, a collection of vectors):

```
Assignments <- c(100, 91, 77, 85, 42)
Exercises <- c(80, 69, 68, 81, 92)
Midterm1 <- c(70, 82, 89, 95, 84)
Name <- c("Student 1", "Student 2", "Student 3", "Student 4",
          "Student 5")
Roster <- data.frame(Name, Assignments, Exercises, Midterm1)
Roster

##        Name Assignments Exercises Midterm1
## 1 Student 1         100        80       70
## 2 Student 2          91        69       82
```

```
## 3 Student 3          77        68       89
## 4 Student 4          85        81       95
## 5 Student 5          42        92       84
```

- Lists (combinations of different types of objects):

```
Class <- c("RPOS 517")
myList <- list(Class, Roster)
myList

## [[1]]
## [1] "RPOS 517"
##
## [[2]]
##        Name Assignments Exercises Midterm1
## 1 Student 1         100        80       70
## 2 Student 2          91        69       82
## 3 Student 3          77        68       89
## 4 Student 4          85        81       95
## 5 Student 5          42        92       84
```

R packages

Many useful and important functions in R are provided via packages that need to be installed separately. A package is a small add-on program for R that you can add to R for some additional functionality. You can do this by using the Package Installer in the RStudio menu (Tools > Package Installer), or by typing the command `install.packages` into the R command line. So, to install the package "foreign", type:

```
install.packages("foreign")
```

in the R command line. You only need to do this once. Once the package is installed, you don't need to run this command again.
Next, every time you use R and want to use a function from a package, you need to load the packages you want to use. To do this, add the following line to your R script.

```
library(foreign)
```

This line has to be in your script file in order for you to use any function from the package "foreign." Otherwise, you will receive an error message, such as `Error: could not find function "read.dta"`. When you build a script file, you should put these commands at the beginning of the script file, always before you use a particular function from a package. We will explore this in examples later.

R help

Within R, you can access the help files for any command that exists by typing `?commandname` or, for a list of the commands within a package, by typing `help(package = "packagename")`. So, for instance:

```
?mean
help(package = "foreign")
```

Workflows and conventions

In this course, we will maintain a consistent approach to working with R. It will make your life much, much easier—with regards to collaboration, replication, and general efficiency. A few really important points that you should consider as you start using R:

- Never type commands into the R command line. Always use a script file in RStudio.
- Always put your work in a **Project**: RStudio will handle the folder structure for you.
- Comment your script files! Comments are indicated by the # sign. Anything that follows after a # sign in the same line will be ignored by R. Compare what happens with the following two lines:

```
1 + a

## Error in eval(expr, envir, enclos): object 'a' not found

# 1 + a
# Comments are useful to remember what I was doing
```

- Save your script files in a project-specific working directory (via the Project structure in RStudio).
- Use a consistent style when writing code. A good place to start is Google's style guide: `http://google-styleguide.googlecode.com/svn/trunk/google-r-style.html`.
- Do not use the `attach()` command.

Useful resources

As R has become one of the most popular programs for statistical computing, the number of resources in print and online has increased dramatically. Searching for terms like "introduction to R software" will return a huge number of results. In this course, we will rely on two main resources:

QR Kabacoff, R. Quick-R. Available at `statmethods.net`. This website offers well-explained computer code to complete most, if not all, of the data analysis tasks we work on in this course.

RC Fox, J. and Weisberg, S. (2011). *An R Companion to Applied Regression, Second Edition.* Sage, Thousand Oaks. This companion to *Applied Regression Analysis* introduces the R software we use in this course. If you would like a lengthier treatment of the methods explained on `statmethods.net` (and more), I recommend this book. You do not need to purchase the book to be successful in this course, but you might find it and its companion website at `tinyurl.com/carbook` useful.

Working with data sets

In your data analysis tasks, you will not type up your data by hand, but use data sets that were created in other formats. A data set is a spreadsheet that contains multiple columns and rows.

mydata

5 observations of 8 variables

	Name	Assignments	Exercises	Midterm1	Midterm2	Replication	Proposal	Review
1	Student 1	100	80	70	83	74	100	61
2	Student 2	91	69	82	95	92	86	59
3	Student 3	77	68	89	90	91	73	83
4	Student 4	85	81	95	90	81	63	81
5	Student 5	42	92	84	93	100	91	100

Figure 2: A dataset, viewed in RStudio.

Columns are called variables and rows are usually observations. We will talk more about data sets and how to handle them throughout this course.

You can easily import such data sets into R. Here are the most common options. Most of them use the foreign package, hence you need to load that package before using these commands:

```
library(foreign)
```

Note that for each command, many options (in R language: arguments) are available; you will most likely need to work with these options at some time, for instance when your source dataset (e.g., in Stata) has value labels. Check the help files for the respective command in that case.

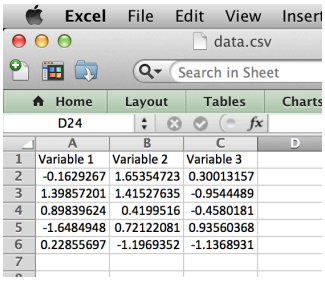

Figure 3: A comma-separated value file in Excel.

- **CSV files:** You will often enter data in Excel. Save your data in Excel as a comma-separated values file with the extension .csv, then read it into R:

```
mydata_csv <- read.csv("http://www.jkarreth.net/files/data.csv",
                       header = TRUE)
mydata_csv

##            y         x1         x2
## 1 -0.1629267  1.6535472  0.3001316
## 2  1.3985720  1.4152763 -0.9544489
## 3  0.8983962  0.4199516 -0.4580181
## 4 -1.6484948  0.7212208  0.9356037
## 5  0.2285570 -1.1969352 -1.1368931
```

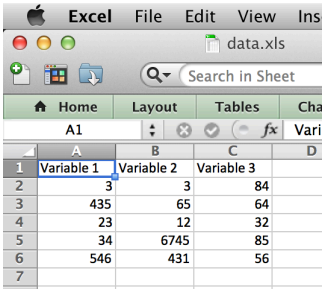

Figure 4: An Excel spreadsheet.

- **Excel files:** You can also read Excel files into R. For this, you need to first load the gdata package. Make sure your Excel spreadsheet is saved in the "97-2004 Workbook" format.

```
library(gdata)
mydata_excel <- read.xls("http://www.jkarreth.net/files/data.xls")
mydata_excel

##   Variable.1 Variable.2 Variable.3
## 1          3          3         84
## 2        435         65         64
## 3         23         12         32
## 4         34       6745         85
## 5        546        431         56
```

- **SPSS files:** If you have an SPSS data file, you can do this:

```
# mydata.spss <- read.spss("http://www.jkarreth.net/files/data.sav",
# use.value.labels = TRUE)
```

- **Stata files:** If you have a Stata data file, you can do this:

```
mydata_stata <- read.dta("http://www.jkarreth.net/files/data.dta",
                         convert.dates = TRUE, convert.factors = TRUE)
mydata_stata
```

```
##            y         x1         x2
## 1 -0.1629267  1.6535472  0.3001316
## 2  1.3985720  1.4152763 -0.9544489
## 3  0.8983962  0.4199516 -0.4580181
## 4 -1.6484948  0.7212208  0.9356037
## 5  0.2285570 -1.1969352 -1.1368931
```

Write and run a script

Practice the things you have just learned by opening and working through the script file `RPOS517_Day1_A1.R`. You can execute a single line by hitting Command + Return while the cursor is at the beginning of a line.

In-class assignment

Start a project in RStudio

From this point on, every script you ever write in RStudio will be part of a specific project. Projects can be assignments for this class, a paper project, a dissertation, and so on. The advantage of using projects in RStudio is that RStudio will automatically create a project folder for you, helping you keep your files organized and ensuring you find everything you need when you revisit your work later.

To start a project, follow these steps:

1. Open RStudio
    - if RStudio is already open, make sure you save any existing script files, then close RStudio, and open it again.
2. Click File > New Project
3. Click through the 3 windows to specify your project (see screenshots on the right)
4. In your new project, open a new RStudio script (File > New File > R Script)
5. When you save this script, RStudio will automatically suggest saving it in your project directory.

In-class assignment: Try this for yourself, creating a minimal script file and saving it in your project directory.

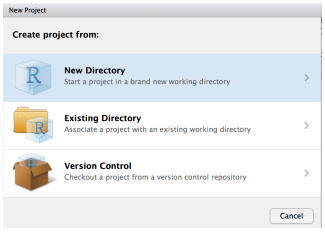

Figure 5: Creating a new directory.

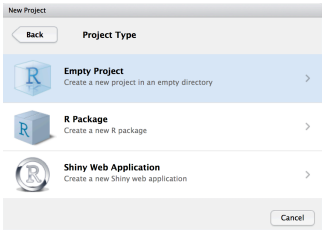

Figure 6: Creating an empty project.

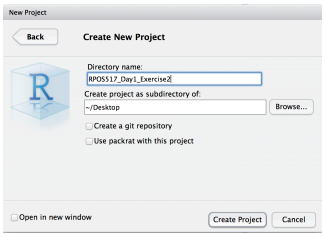

Figure 7: Providing a project name.

Start a data analysis notebook in RStudio

You will document all data analysis work in this course (and hopefully beyond!) using R Markdown to write a data analysis notebook. R Markdown is a simple way of writing text and R code in one document. It has four major advantages:

- It is saved as a text file and can be opened on any computer with any text editor, regardless of whether you have R and RStudio installed or not
- It combines your data analysis commands with the output you produce (such as tables or plots)
- It allows you to write up text (such as explanations of what you are doing, or even a whole research paper!) in the same document
- It produces output that is already properly formatted in Word or PDF

This means that you do not need to spend much (if any at all) time to switch between writing up an assignment or a data analysis report and the actual data analysis.
To begin a notebook, follow these steps:

For more information on R Markdown, please read through this website: `http://rmarkdown.rstudio.com`

1. Open your RStudio project that you created previously
2. Open a new file (File > New File)
3. Click "R Markdown"
4. Select "Word" as output format
5. Begin working on your file
6. Convert your notebook to Word by clicking the "Knit Word" button on the top.

In this course, you have to turn in your assignments as R data analysis notebooks. Practice this using the file `rpos517_assignment_template.Rmd` as a model. You can download this file from the course website. Open the file and compile it first as a Word, then as a PDF document.

In-class assignment

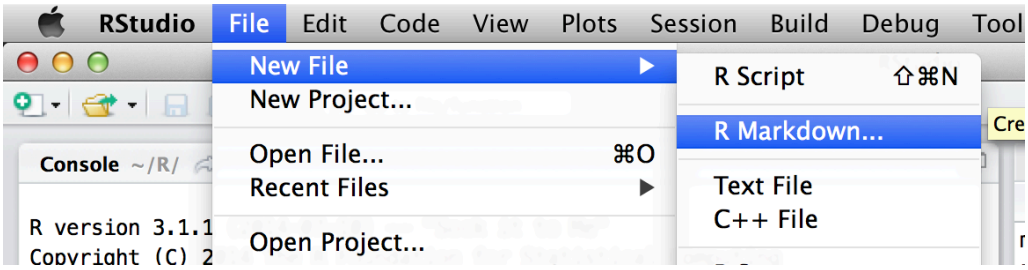

Figure 8: Opening a new file in RStudio.

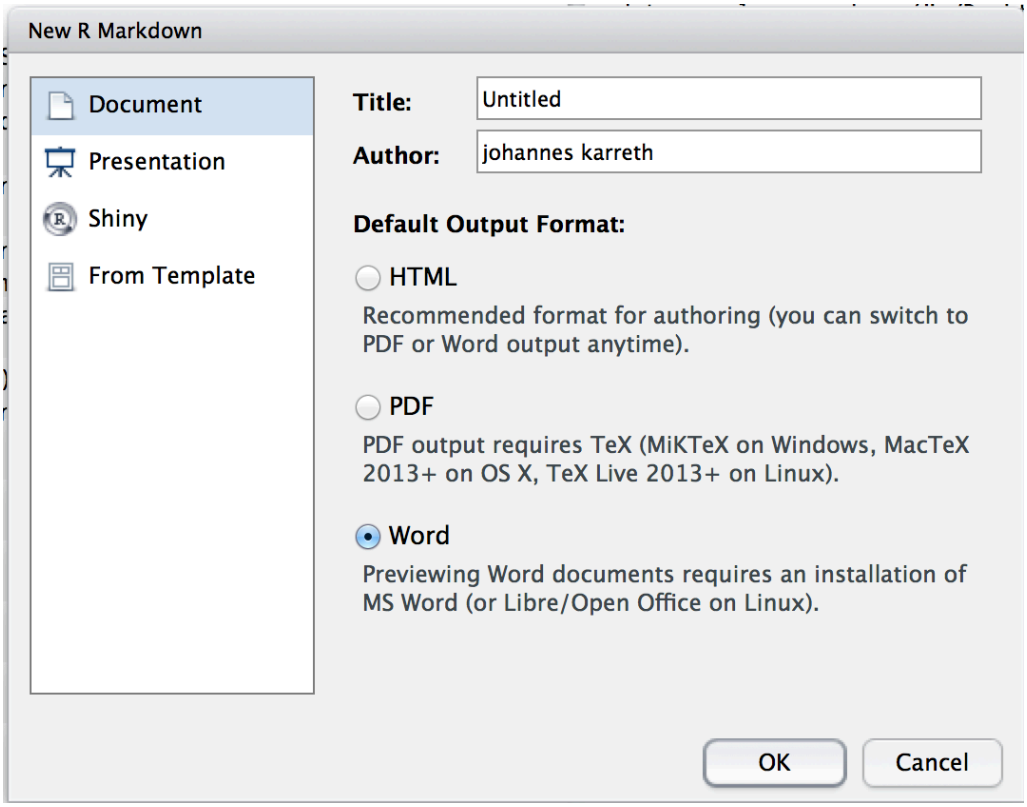

Figure 9: Selecting your desired output format.

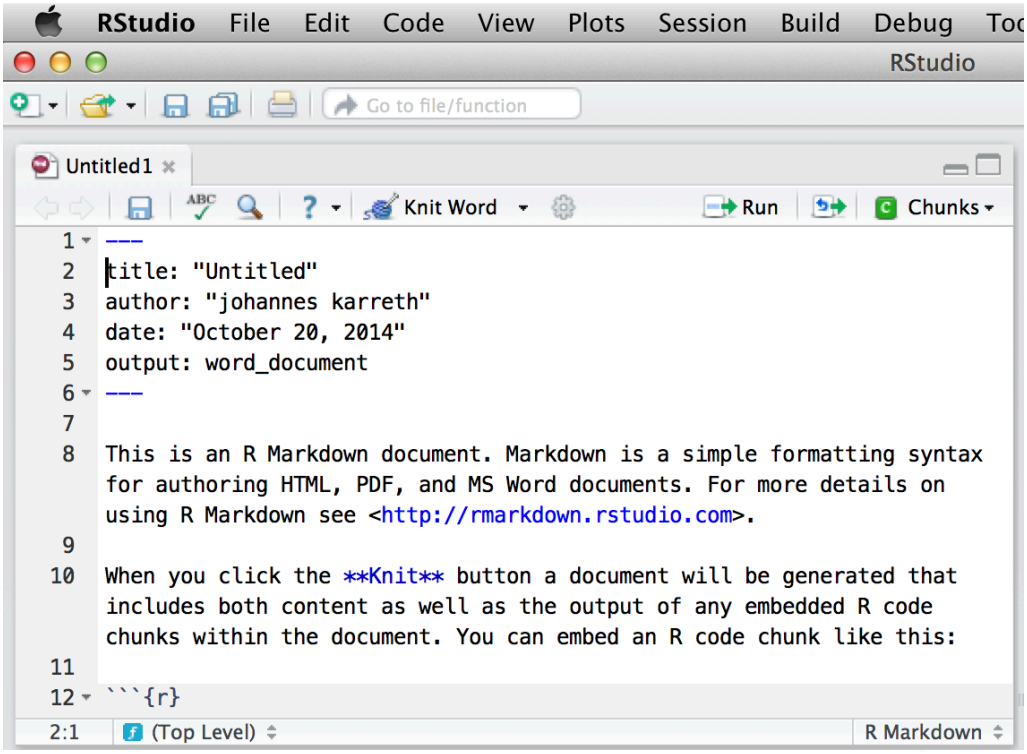

Figure 10: You'll start with this template.

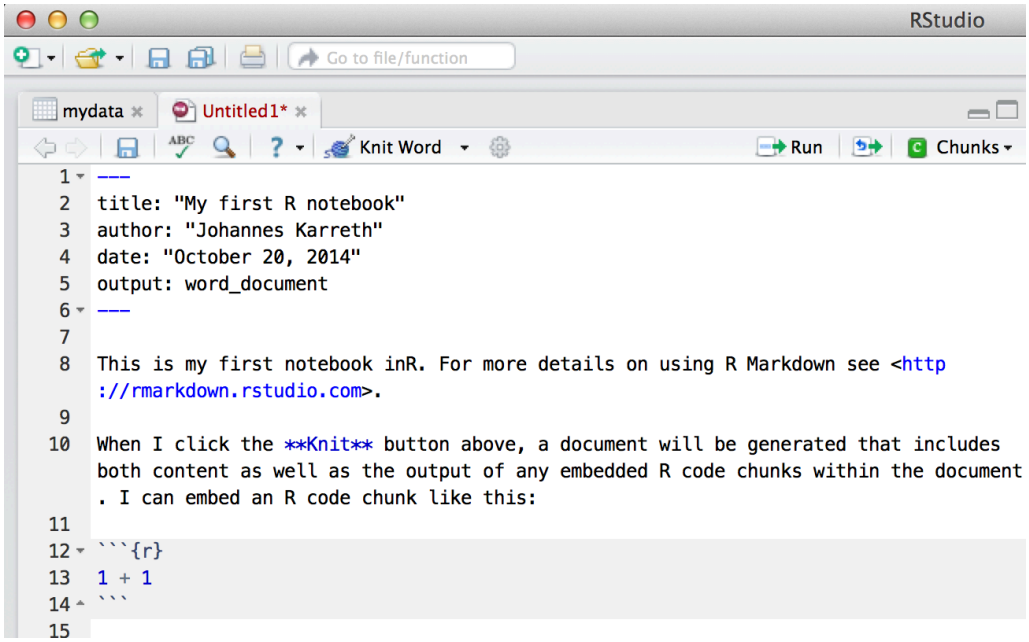

Figure 11: Here, I made some edits.

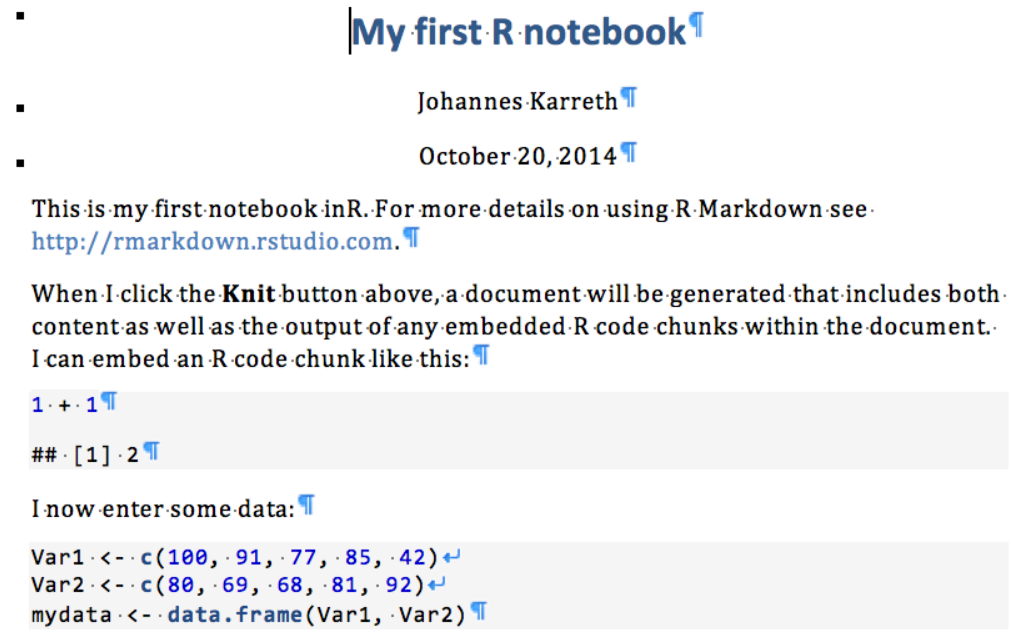

Figure 12: The first few lines of my notebook in Word.