BTRY 6020 Lab 1: Doing an Analysis from Scratch

January 30, 2017

1. Introduction

In BTRY6010 R Markdown was used in order to emphasize the necessity of reproducibility in statistical analysis. At its core, R Markdown provides one with a way to heavily comment code in order to ensure that there is no ambiguity in the methods being applied.

You may be surprised at the vast difference in readibility from giving someone 100 lines of codes to giving someone 100 lines of code with output and full sentences of explanation in between!

2. Goals for this Lab

This lab will serve two purposes: first it will reacquaint you with doing an analysis in R Markdown, and second it will provide a reference for many R Markdown basics. This will be a review for many of you (which is good), but the goal is to just make sure everyone is starting off this course on the same page.

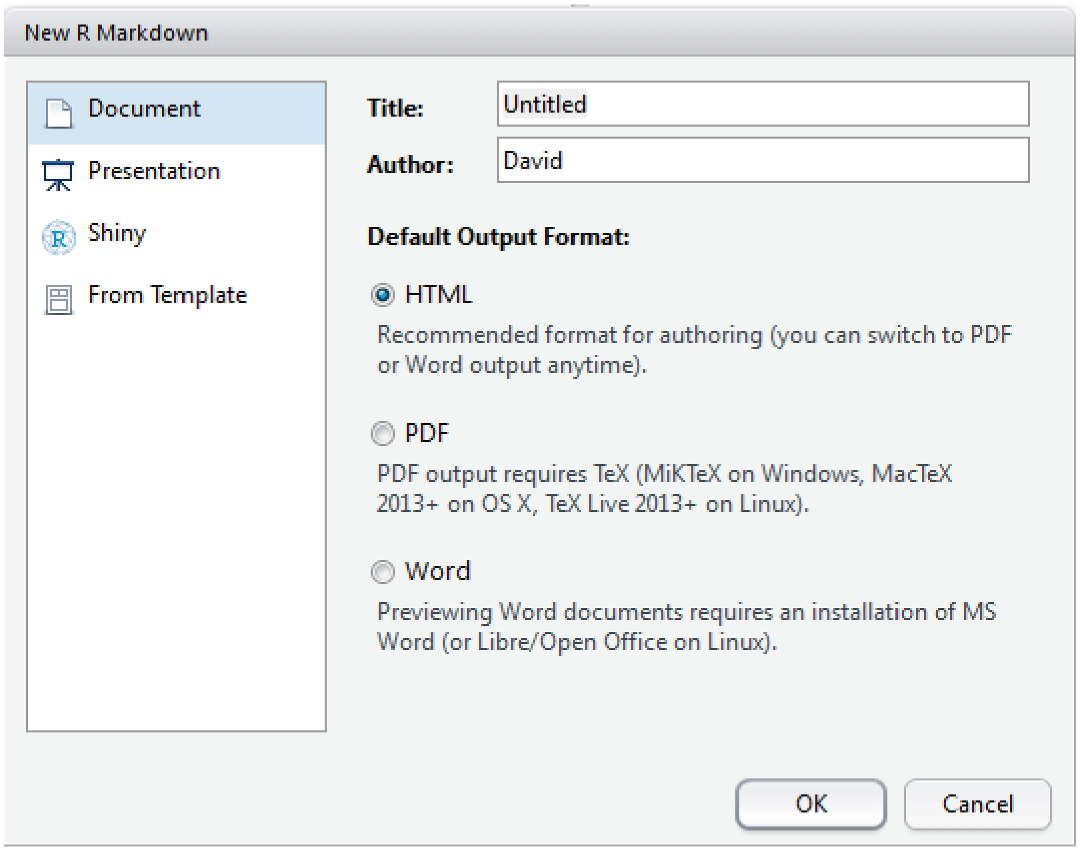
The basics that we will cover are:

1. How to create an R Markdown file to work with
2. A few synatx choices for typesetting in R Markdown
3. Creating a code chunk
4. R coding basics and loading Libraries
5. The difference between the Console Environment and the R Markdown Environment
6. Loading in Data
7. Doing an Linear Regression Analysis

3. Creating your R Markdown file

At the beginning of every homework, you will need to create an R Markdown file by yourself. There are only a few steps to doing this, but we will go over the steps together now.

1. Go to File > New File > R markdown. The following will show up



Title: Untitled

Author: David

Default Output Format:

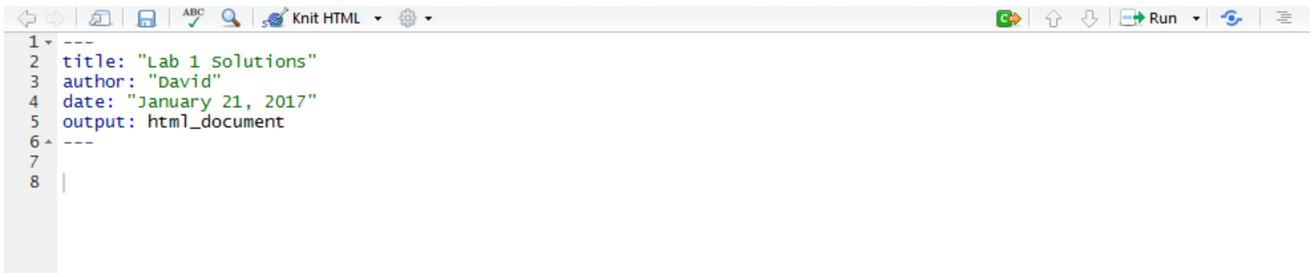
[✓ HTML]
Recommended format for authoring (you can switch to PDF or Word output anytime).

PDF
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

Word
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

2. Change the Title to: “Lab 1 Solutions” and the Author to your name. It doesn’t matter what you select the Output as – this can be changed later at any time.

3. In the new file that opens up, erase all the code that shows up after line 7. This is your empty rmd document and should look like the following



```
1 ---
2 title: "Lab 1 Solutions"
3 author: "David"
4 date: "January 21, 2017"
5 output: html_document
6 ---
7
8
```

We’ll use this file throughout the lab to try out a few things, and complete the analysis posed at the end of this document.

4. Typesetting in R markdown

Here are just a few basic formatting tools that are used in an R Markdown document:

- `#` : begins a new section. If you do `##` then it will begin a subsection
- `*` : asterisks around text will italicize the text
- `**` : double asterisks around text will bold the text
- `$` : dollar signs around text will put the text in math mode to allow for math symbols. E.g. {X_n} N(, ^2/n) becomes $\bar{X}_n \sim N(\mu, \sigma^2/n)$. We will provide most of the math mode formatting that you will need to use.

- ``: Single quotes around text (located at the top left of your keyboard) will make text look like code. This is useful when citing an R function (e.g. `qnorm()`). The key looks like this

In your lab solutions file, create a new section called “Central Limit Theorem”. Write a statement about the *Central Limit Theorem* that has the following attributes:

1. States what the Central Limit Theorem is
2. The words “Central Limit Theorem” are italicized
3. A choice of words are bolded (e.g. The central limit theorem is **extremely** useful.)
4. Using math mode, a mathematical formulation of the Central Limit Theorem.

This should provide you with most (if not all) of the syntax you’ll need while typesetting in R Markdown.

5. Creating a Code Chunk

Now let’s get to actually including code in our R Markdown documents! This is what separates R Markdown from your usual Word document.

There are two ways to include a code chunk. The first is to click the “chunk” button at the top of your R Markdown script window. The button looks like: .

You can also simply type on a new line: ```{r}

Then when you are done writing `R` code, type ``` to finish the code chunk.

In your lab solutions file, create a new section called “Testing R Code”, and create a code chunk.

6. R Code (Chunk) Basics and Loading Libraries

In this section we go over how you want to present a code chunk in your R Markdown file.

In the way that we use `R` , we will be loading in data and then running functions on the data to complete our analysis.

One reason that `R` has been so heavily adopted by researchers is that the number of functions available to apply to our data is almost endless. These functions are located in packages, and these packages contain state of the art statistical methodology.

In this section we are going to be applying the `boxcox` function to our data (don’t worry – you won’t be tested on this transformation). The Box-Cox transformation raises your data to some power (e.g. $\lambda = 2$ squares each observation) and in practice can be used to complete a linear regression test, even if your data is only normally distributed *after some power transformation*. The `boxcox` function will show us which lambda values bring our data closest to normality.

So, let’s use this function! Use the following steps to edit the “Testing R Code” section in your lab solutions file to try out this function.

1. First, the `boxcox` function is in the `MASS` package. Install this package by typing `install.packages("MASS")` into the console. **Note: if we ask you to use a function that's located in a package, we will tell you what package needs to be installed. This step only needs to be completed once on your computer.**
2. Now, to allow us to use the functions in the `MASS` package, in the code chunk we created in the "Testing R Code" section add the line `library(MASS)` . **Note: to load any other package, replace "MASS" with the name of the package, and complete steps 1 and 2**.
3. In the code chunk, add a new line that says `#create normal data` . The `#` in the code chunk tell R Markdown that this is a comment, and our comment is that we will create some normal data
4. On the next line, run the code `data = rnorm(100, mean = 10, sd = 1)` . This will create normal data with mean 10 and variance 1. In practice we will usually use real data, but either way, you always need a line of code that creates data for you.
5. Next, add a comment saying we will be doing a Box-Cox transformation, and then add the line of code `boxcox(data~1, lambda = seq(-5, 5, 1/10))`
6. Now, *before the code chunk* add a short description in full sentences of what we're doing (e.g. generating some data and trying out the Box-Cox transform)

At this point it should be clear what your coding intentions are in the "Testing R Code" section of your document. Now you can knit the file (by clicking the at the top of the R Markdown file), and we will write some conclusions after the code chunk about the Box-Cox plot that shows up.

6.1 Main Points for doing an R analysis

You don't need to worry about the Box-Cox transformation, when we do provide a new analysis you'll be tested on we'll spend much longer on it! The main things to note is that we did the following:

- described our analysis before the code chunk
- loaded the needed package(s)
- loaded our data
- ran the appropriate function on our data
- concluded our analysis after the code chunk

6.2 Main Points for loading a package

(As per step 1 and 2) to load a package, first type `install.packages("PACKAGENAME")` in the console where `PACKAGENAME` is replaced by the package you wish to use. Then type `library(PACKAGENAME)` in the *environment* you're working in to access those functions.

7. The R Markdown and R Console Environments

The `R` environments are where packages are loaded, and variables are saved. If you run a new line that calls a variable or a function, it's important that a previous line *in the same environment* has created that variable or opened the package for that function.

The R Markdown document you create will give code that fully creates the environment for the statistical analysis you describe in the document.

The `R` console provides a useful place to try out code to prepare for your R Markdown environment. If you have to knit your document everytime you want to see if some R code works, then that can be very time consuming.

The process I recommend is:

1. Try some code in the console
2. If the code is doing what you like, add it to a code chunk in your Rmarkdown document
3. Repeat steps 1 & 2 until you're ready to test your R Markdown document
4. Knit the R Markdown document to make sure all your code has been transferred over correctly.

Note that for step 3, the number of lines of code you add to your R Markdown document is up to you, however if there is an error, then the R Markdown will just tell you the code chunk where the error is happening. Therefore, the less often you test, the harder it will be to find your error.

We will see some examples of this process when we go over the linear regression example. Just remember: **the R Markdown environment (i.e. any code in your R markdown file) is totally separate from your R Console environment (i.e. any code ran in your R Console).**

8. Loading in Data

We will look at data that compares the chirps per second for the striped ground cricket, to the temperature in degrees farhenhit.

The data set is the excel file `slr02.xlsx`. In order to load this data set, from the tabs at the top of the screen we can do: File > Import Dataset > From Excel and select the data set. **Note: If you are using an older version of R Studio, the Import Dataset option is located under the Tools tab.**

After selecting the file (and inspecting the preview generated to make sure the file is being loaded in correctly), a few lines of code will appear in your console. Copy the line of code that uses the `read_excel` or `read.csv` or `read.table` function into an R chunk. *Do not copy the line of code that uses the view function*. Also, if you use the `read_excel` function, note that the `readxl` library was loaded into your console. Make sure to copy this into the R Markdown environment as well. This is all shown in the next code chunk.

```
#Loading in the data
library(readxl)
slr02 <- read_excel("~/Dropbox/BTRY6020/2017/lab1/slr02.xlsx")
```

In practice data can be endlessly complicated to prepare for analysis (in fMRI analysis the dataset can take multiple days to process), however the Import Dataset option in `R` is fairly robust.

9. A Linear Regression Example

Now, in your Lab Solutions file, create a section called “Linear Regression Example”.

In this section, do the following steps to complete the analysis:

1. Describe the data and what we can learn from analyzing this data.
2. Add a code chunk that loads the data in your R Markdown environment and uses the `head` function to preview that data
3. Plot *var1* against *var2* and describe why a linear regression may be appropriate for modelling this data.
4. Add a code chunk that fits the a linear regression to the data using the `lm` function. Use the `names` function on the `lm variable` created to see what variables are available to you.
5. Describe some diagnostic tests you will use, and add a code chunk that makes diagnostic plots.
6. Make conclusions based on your diagnostic plots.
7. If appropriate (based on your diagnostic plots), include a code chunk that uses the `summary` function on your `lm variable` to look at how your model was fitted.
8. Make conclusions about your analysis.