Rmarkdown

20 April, 2017

Intro

This is an R Markdown document. Markdown 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.

Cheatsheet can be found http://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf

Reference guide can be found https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf

Setup

For today's lecture you will need to open up an Rmarkdown document. Go to File -> New File -> R Markdown and create an PDF document with your name. Title it "R Markdown Practice."

At the top of the Rmd script you should see a section that looks like the following:

---

title: "Rmarkdown"
author: "Your Name"
date: '20 April, 2017'
output: html_document

Output options?

Basics

Rmarkdown has two modes: “text and”code"

Default settings for text mode are black text on a white background.

In the text mode of a markdown document you can type text as you would in a word document and the output document simply returns exactly what you put in, tpyos and all.

In Class Exercise 1

- Introduction
- Data
  - Sources
  - Cleaning Methods
- Analysis
  - Charts
  - Regression Results
  - Discussion
- Conclusion

1. Numbered Lists are fun too!
   1.2 They make it easy to organize things.

LaTeX Basics

LaTeXis a powerful typesetting language used throughout academia and the professional world.

One of the most important features of LaTeXfor our purposes will be the use of math mode.

- Inline equations occur within the text smoothly.
- Display equations occur separate from the text.

Why should you care about this? Regression equations!

Inline Equations

Inline math mode is accessed using a single $ sign surrounding each end of the equation.

- If I wanted to show: $y = mx + b$
- I would type: $y = mx + b$
- LaTeXsupports the greek alphabet so you can type $ \beta $ β or $ \gamma $ γ

Latex Basics

- \sum is a built-in symbol to LaTeX. And gives $\sum$
  - There are many others such as \infty which gives ∞
- The ^ sign gives superscript and the brackets take the numbers to display in the superscript. So x^{2} gives x^2 and x^{x^2} gives x^{x^2}
  - The brackets can accept multiple numbers, so you could theoretically type x^{2,3} which gives $x^{2,3}$ which makes no sense.
  - The same sintax applys to subscripting. To show element A in row i and column j in a matrix you would type A_{i,j} and show $A_{i,j}$

Latex

- Where is appropriate to use inline equations?

Display Equations

In addition to inline math mode, equations in Latex can also be shown in display mode. In this mode the expressions are centered and in a separate line from the main text.

$$\sum_{i=1}^{n} X_i$$

. This mode is activated by using two $ signs surrounding the formula in place of one.

Reproduce the following equation. You may need to google/ask a TA for some support

$$\text{Outcome}_{i,t} = \beta_{i,t} + \gamma_{i,t}\text{Dependent Variable}_{i,t} + \alpha_{i,t}\text{Control Variable}_{i,t}^{\theta}$$

Code Mode

- Embedded R code is where the power of Rmarkdown really comes through.
- Up until now we have not really seen the difference between a markdown file and regular Latex.
- The ability to embed R code, code that can be evaluated in-line with the text, is what makes Rmarkdown so powerful.
- Similar to text, code chunks can be added either inline or chunk.

Inline Code

- Code made is activated via the ‘ character.
- A single ‘ followed by text and closed by another ‘ creates bolded text with a different font.
- This is useful for displaying function names such as `sum()` in a way that readers can understand to be code.
- Inline code can also be evaluated by adding a marker denoting the language to run immediately following the ‘ mark.
  - I.e. ‘ r sum(c(3,4,5))‘ evaluates to 12. We could specify other engines instead of R if desired.
- Let's create the following text: the sum of 1 to 100 is 5050

Code Chunks

- Time to create a code chunk!
- To create a new code chunk press Ctrl+Alt+I or type in three ‘ and specify that you want the R engine in braces like on a new line like so:

“‘{r}

All options for the code chunk go between the braces.
* If you inserted the code chunk with Ctrl+Alt+I it will already have a closing in it.
* If you started the code chunk manually you must close it with three more ‘ on an empty line. So your code chunk should look like the following:

“‘ {r options}
Code
more code
“‘

Within each chunk you can write code like in a normal R script

Basic R chunk

For this lecture we will explore the economics dataset include with the ggplot2 package.

Before we continue make sure you have the following libraries:

```
library(knitr)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
```

Changing Chunk Options

These options affect whether the code is evaluated or not, whether the code itself is shown in the output document as well as many other controls surrounding the output.

Specifying chunk options is done after a comma within the braces of your chunk declaration like so:

```
```{r chunk name, option1 = TRUE, option2 = FALSE, etc}
Code More Code ```
```
```

Chunk Option Overview

Chunk options should be comma separated. Here are some that I frequently use:

1. echo - controls display of code
   - If echo = TRUE the code in the code chunk will be displayed
   - If echo = FALSE the code in the chunk will not be displayed (and the code will still evaluate)
   - Similar to Eval can take a vector argument as well.
2. error - controls error messages
   - If error = TRUE error messages are displayed and code continues to run
   - If error = FALSE error messages are not displayed, code does not continue to run

More Chunk Options

3. Include
   - If include = TRUE (default) the chunk will be run and included in the final document
   - If include = FALSE the chunk will be run but not included in the final document
     - E.g. plots/tables generated in the chunk are still created and accessible in other chunks
4. Eval
   - If eval = FALSE knitr will not run the code in the code chunk
   - If eval = TRUE (default) the code will be run
   - This is useful for debugging as you can turn eval to FALSE in a chunk and play around with your code but not raise issues compiling your document.
   - This can also take a vector to select which expression(s) to evaluate. e.g. `eval = c(2,4,5)` or `eval = -(4:5)`

In Class Exercise

- Let's explore some r chunk options
- How do you suppress the output?
- How do you display the code without running it?
- How do you run the code without displaying it?

In Class Exercise

In your Rmarkdown document create a code chunk and copy in the following code, then knit your document.

```
data(economics)
## First we should take a look at our data
head(economics)
```

```
## # A tibble: 6 x 6
##         date   pce    pop psavert uempmed unemploy
##       <date> <dbl>  <int>   <dbl>   <dbl>    <int>
## 1 1967-07-01 507.4 198712    12.5     4.5     2944
## 2 1967-08-01 510.5 198911    12.5     4.7     2945
## 3 1967-09-01 516.3 199113    11.7     4.6     2958
## 4 1967-10-01 512.9 199311    12.5     4.9     3143
## 5 1967-11-01 518.1 199498    12.5     4.7     3066
## 6 1967-12-01 525.8 199657    12.1     4.8     3018
```

Setting Chunk Options For The Entire Document

Chunk options can also be set at the top of the markdown document using the `knitr::opts_chunk$set()` function.

This function can be called anywhere in the document but it is a good idea to set it at the beginning since chances set will only affect code downstream.

For example, if we wanted to set the default for the echo value to TRUE so that by default all of our code block would be rendered onto the final document and include to be true so that our code would always be run we could do the following in an R chunk:

```
knitr::opts_chunk$set(echo = TRUE, include = TRUE)
```

Calling Graphs

Just as we were able to show table output inline with our text above, we can also include graphics in our final output as well.

- For ggplot objects this simply requires creating the objects and then calling them.
- Additionally, we can set chunk options either specifically or globally that change the size of our output plots.

Example Plot

```
plot.data <- economics %>%
    filter(date >= as.Date("2006-01-01")) %>%
    select(date, uempmed)

plot1 <- ggplot(plot.data, aes(x = date, y = uempmed)) +
    geom_line() +
    labs(title = "US unemployment (monthly)",
         x = "Date",
         y = "Percent")
```

Plot

Figure 1: Example caption

You may notice that this figure is smaller than the default, centered, and has a caption underneath.

Plot Round 2

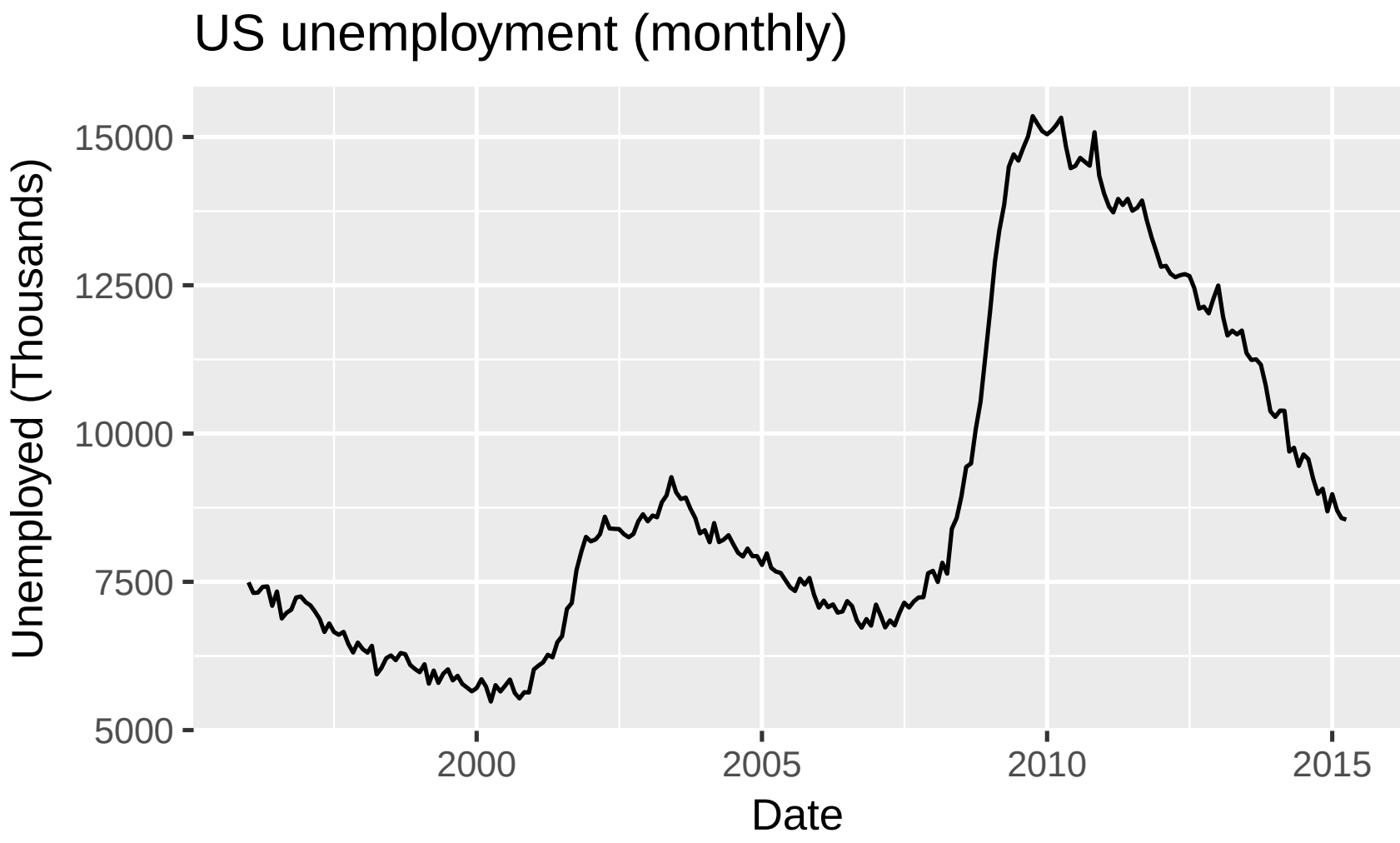

Figure 2: Time Series of US Unemployment

In Class Exercise

Copy some of your code from your project and them make a plot using Rmarkdown

Inserting tables

Just like we can insert graphs into the markdown document we can also insert tables

- The knitr package comes with the `kable` function that we can use to call a table that is more formatted than standard R output.

```
kable(head(economics, 3), "pandoc",
      col.names = c("Date", "Pce", "Pop",
                    "Psavert", "uempmed", "unemploy"),
      align = "c")
```

Date	Pce	Pop	Psavert	uempmed	unemploy
1967-07-01	507.4	198712	12.5	4.5	2944
1967-08-01	510.5	198911	12.5	4.7	2945
1967-09-01	516.3	199113	11.7	4.6	2958

Inserting tables

```
library(pander)

economics_reg <- lm(psavert ~ pce + pop + uempmed + unemplo

panderOptions("digits", 2)
pander(economics_reg)
```

Table 2: Fitting linear model: psavert ~ pce + pop + uempmed + unemploy

	Estimate	Std. Error	t value	Pr(>|t|)
pce	0.00084	0.00011	7.6	1.4e-13
pop	-0.00018	1e-05	-17	1.1e-53
uempmed	0.11	0.033	3.4	0.00075
unemploy	0.00029	4.5e-05	6.5	1.6e-10
(Intercept)	46	2.1	22	9.2e-79

Beamer Slides

Beamer slides are the pdf version of PowerPoint. This slide show is Beamer.

To start a Beamer slide presentation go to File -> New File -> RMarkdown -> Presentation -> PDF (Beamer)

The syntax for making a slideshow is the same as we have covered in this lecture where we made a longer-style document. The main difference is that when making a slideshow, instead of creating a large header, using a single # will insert a new slide with whatever title you decide to give it.