Example

Contents

List of Figures

List of Tables

Introduction

This is an example of an RMarkdown document that can be authored using `compileRmd`.
The html version was compiled using the command: `compileRmd()`.
The word version was compiled using: `compileRmd(format='word')`.
The pdf version was compiled using: `compileRmd(format='tex', texEngine='pdflatex', bibEngine='bibtex')`.

Motivation

I use RMarkdown for most of my writing.

It is missing some features, such as the ability to number figures (or tables) and then cross-referencing them later. So, I thought I'd try coming up with a way of adding this feature.

At the same time, I was also interested in getting a better handle on regular expressions, and this problem gave me an excuse to play with them. Conventional wisdom would suggest this was a bad idea:

> Some people, when confronted with a problem, think "I know, I'll use regular expressions."
> Now they have two problems.
> \- jwz

I've tried to avoid using long, intricate regexs because:

1. I didn't want the source code to look like it was written by Q*bert

2. I'm not *that* good at regex to begin with

Basics

YAML Headers

The YAML header contains information which RMarkdown and Pandoc use for producing the output. It is usually the first thing in an Rmarkdown document.

An example of a basic YAML header for tex (pdf) output is below:

```
---
output:
  pdf_document:
    fig_caption: yes
    keep_tex: yes
bibliography: [`varBib`]
csl: `varCSL`
---
```

The bare minimum file structure for 'compileRmd' is an RMarkdown .Rmd file which contains a YAML header for either an html, word or pdf document. However, this is not the most practical way because the YAML header specifies the output format. So, if you want to output a pdf and docx file, you'd have to change the YAML header. An alternative is to split the YAML header into an external file, and save it as either `yaml_md.txt`, `yaml_html.txt`, `yaml_htmlB.txt`, `yaml_word.txt`, or `yaml_tex.txt`. When we're outputting to a format, such as html, we can make `compileRmd` grab the html header without having to do anything ourselves. If the above naming scheme is used, then `compileRmd` will automatically read the header file. Otherwise, you will have to specify the file using `compileRmd(yaml='path/to/yaml_header.txt')`

Another advantage of having YAMl headers as separate files is that each output format can have its own settings without conflicting with each other. Beyond this, additional files such as templates or .bib files can also be included in the YAML header.

Look through the YAML files (`yaml_*.txt`) which are in the same folder as this .Rmd file. You'll see that the .bib file has been included in the YAMl header, and there are also some knitr settings. For example, the html YAML has a style.css and has PNG output for figures, whereas the tex YAML has pdf output for figures.

Additionally, some parts of the YAMl header contain placeholders such as `varBib` and `varCSL`, which are placeholders for the .bib and .csl files respectively. The input arguments (e.g. `compileRmd(bibFile='path/to/file.bib', cslFile='path/to/file.csl')`) will overwrite the placeholder. Alternatively, the user can enter the appropriate files into the YAML header directly. In this example's headers, the .bib file is specified in the YAML header, but the csl file is not.

Syntax

Markdown's syntax is straightforward so it should be easy to pick up by just reading the source document.

Perhaps the only thing that might leave newcomers confused is how to force a line break.
Simply hitting the
enter key will not cause a
new line to begin.
The previous sentence is broken into lines (it might qualify as a haiku?) yet the 2nd last sentence is not broken. The reasoning is subtle: you need two blank spaces before breaking the line to create a new line.

Citations

Here, look, it's a citation to a fictional paper [1].

For html or word output, pandoc's citation syntax can also be used. This becomes a bit troublesome if the desired output format is *not* LaTeX because it will hardcode the .tex output by default. There's a special document for details.

Equations

An in-line equation can be dropped, $f(x) = \sqrt{(ax-b)^2}$, anywhere. The syntax is identical to LaTeX, and $c\alpha$n bϵ useδ for symbols as well.

Some equations are important and deserve their own block. For example, the following equation keeps me up at night:

$$P(job|degree) = \frac{P(degree|job)P(job)}{P(degree)}$$

More often than not, equations have to be referenced in the text. One way to do this is by numbering them. For example, equation 1 is the well known cDonald's theorm.

$$n^2 + 9 + 9 \tag{1}$$

Figures and Images

Have a look at figure 1. It contains a short caption which will not show up in the figure caption, but will show up in the list of figures (pdf output only).

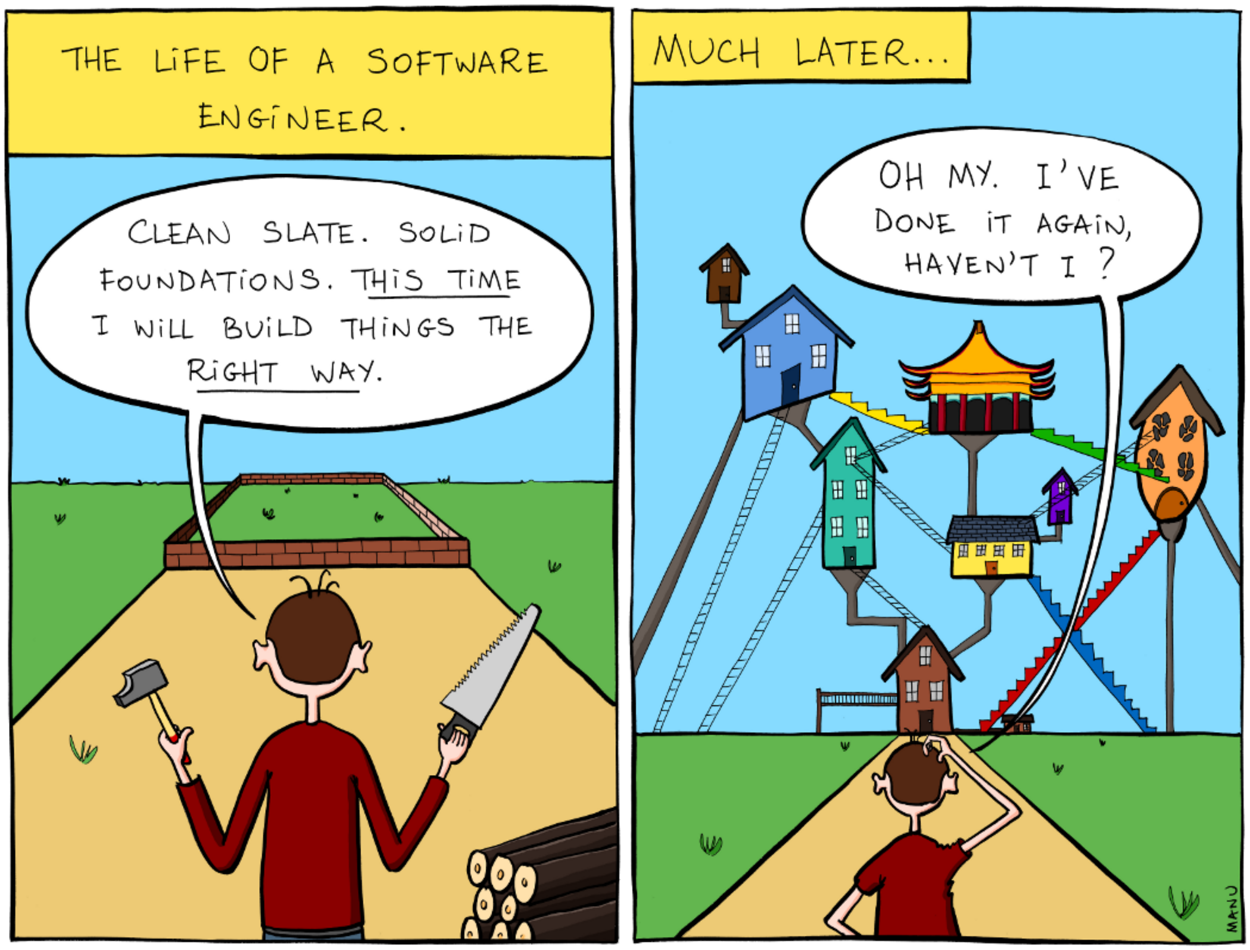
THE LIFE OF A SOFTWARE ENGINEER.
CLEAN SLATE. SOLID FOUNDATIONS. THIS TIME I WILL BUILD THINGS THE RIGHT WAY.
MUCH LATER...
OH MY. I'VE DONE IT AGAIN, HAVEN'T I ?
MANU

Figure 1: Is this what people mean when they say 'building code violations'? Artist:Manu Cornet

Tables

I have a habit of avoiding tables because they're usually a pain to make - for example, tables in LaTeX require a summoning ritual which I can never get quite right on the first try. Thankfully, tables are pretty easy to make in Markdown. Check out table 1 below.

Table 1: A fictional table. Chairs not included.

x	y
1	1
2	4
3	9
4	16

The next chapter will show an even easier way of making tables.

Dynamic Results

It is possible to embed code in a RMarkdown file and have it execute when the output is produced. This chapter provides an example of that.

Tables

I have a csv file saved in the `./exampleData/` folder which contains a summary of the New York vs Washington series of the 2015 Stanley Cup Playoffs. The CSV only contains the goals scored by each team. We can calculate the goal difference, and the win/loss through R, and display the results. Check out table 2 which shows the results.

Table 2: Summary of New York Rangers vs Washington Capitals series of the 2015 Stanley Cup Playoffs, from the Rangers POV

Game	Goals For	Goals Against	Goal Diff	Win - Loss
1	1	2	-1	0 - 1
2	3	2	+1	1 - 1
3	0	1	-1	1 - 2
4	1	2	-1	1 - 3
5	2	1	+1	2 - 3
6	4	3	+1	3 - 3
7	2	1	+1	4 - 3

Analysis

Let's perform some simple analysis on the data in table 2.

The New York Rangers scored a total of 13 goals, with an average of 1.9 $\pm$ 1.3 goals per game. The highest scoring game for the Rangers was game 6 where they scored 4 goals. This was a hard-fought series, but ultimately won by the **New York Rangers**.

If you're reading the output document, then the previous paragraph probably looks normal (asside from the obnoxious bold at the end). In the .Rmd source, you can see that there is embedded R code. For tidiness, I did the calculations inside an R code block, and then called the results in-line. The advantage with this is that these results will automatically change if the input data was changed.

For example, suppose that the csv data was incorrect and that game 7 actually had 5 goals scored in favour of New York and 8 goals scored by Washington. This means that table 2 has to be changed. Furthermore, the results of our analysis (the total goals, average goals per game, highest scoring game, and the eventual winner) will all have to be changed. That's a lot of work... but not for us - all we have to do is update the CSV file.

To test this, I modified the csv file and saved it as `./exampleData/NYRvsWSH-alt.csv`. Change the 'Load Data' line in the R code above to read this csv file and re-compile this document to see the results.

Figures

Figures can also be created at runtime and embedded. Take a look at figure 2. This figure will also change if the csv data changes.

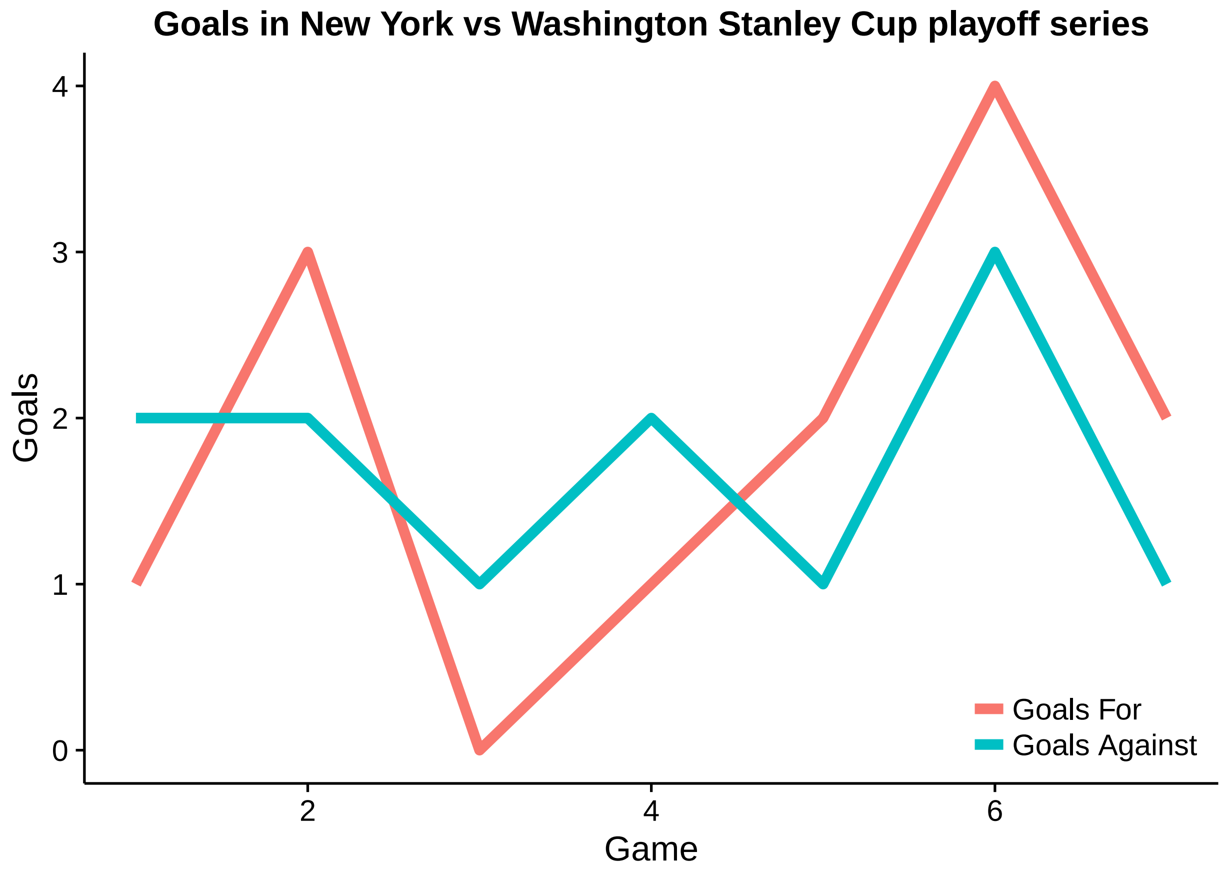

Figure 2: Goals in the New York vs Washington series.

References

[1] Rosencrantz and Guildenstern. Exploring the Law of Averages with Coin Flips. *Poisson Noise And Stochastics*, 2, August 1966.