Post01-John-Nipp

John Nipp

October 30, 2017

R Markdown: in History and in Practice!

Check it

Introduction

Professor Sanchez gave us an overview of the duties of being a data analyst/scientist (DS). He mentioned that DSs use close to 70% of their time in data preparation. This is one of the biggest reasons for the course being a requirement for stats majors, and is what gives this course utilitarian value. In fact, if you have done your research, you'll know that R is an extremely important language for most job positions titled with "Analyst", and that it is used across many different related disciplines. I took this class because most jobs I am looking to apply for have proficiency in R as either a preferred or required skill.

But… why is there such an empahsis on rmd files? Since the beginning of the course, almost every assignment has had a heavy emphasis in the use of R-Markdown. Professor Sanchez is extremely strict on us turning in md files! This leads me to the following questions. If Professor puts a strong emphasis on the practicality of this course, then:

1. Where did R Markdown come from?
2. What are the functions of RMarkdown and how does it work?
3. How is RMarkdown used in the field?

This post is the culmination of research found searching for the answers to these questions. Oh and if he were to ask this post falls under the category "reporting tools".

History behind R Markdown

R

We learned in the first week of class that R is a free implementation of the language S. An implementation is a system for executing computer programs. Why is this necessary? Because the computer uses "machine code" which typically is difficult to read or interact with by a person. An implentation is creates a simpler syntax for a person to use that has the same functions as the "lower level language". In other words, the implementation translates what the person is coding to the language the computer can read to execute the program.

Normal R programs are called R-scripts. If you are trying to create a program to be executed, R-scripts may serve their purpose. The major issue with this is that R is often used in analysis & reports. If you were to use R-scripts alone, you would have to create a seperate text or word document and show the output! This violates the philosophy of R, which is to give us the ability to do interactive data analysis.

```
# Comments alone can't be used to create an HTML
# This means that if you expect to make a report using R-Scripts... without any
# rendering they aren't going to come out particularly attractive.
```

We can create reports through RScripts, but it is of course using an rmarkdown package, and it's a lot less efficient then writing the report straight through an rmd file.

Markdown

Markdown was created in 2004 by John Gruber. His goal was "to write using an easy-to-read, easy-to-write plain text format, and optionally convert it to structurally valid XHTML (or HTML)". Markdown is a light-weight markup language.

Light-weight markup languages are intended to make it easy for someone to write into file with very easy syntax.

Let's say I'm using python and I want to print a haiku:

```
print("      God created cake")
print("    Tastes like total ecestacy")
print("      No need for drugs kids")
```

```
##       God created cake
##     Tastes like total ecestacy
##       No need for drugs kids
```

As you can see, I had to use a specific function to print this. In a light-weight markup language, printing is nearly as easy as typing it out on a word document. Python is not a markup language.

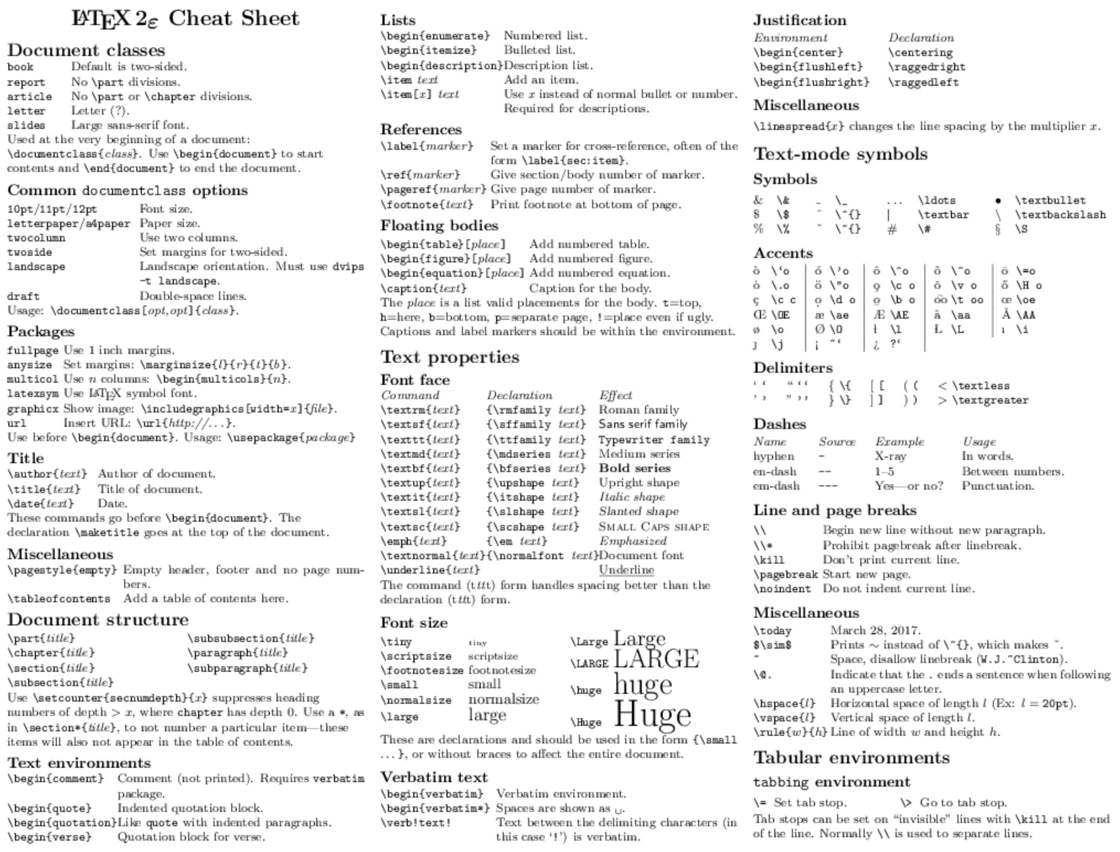
Computational Notebooks

A computational notebook is a file that combines text features with executable code. In the late 1980s, Mathematica introduced it's front end. In 2001, IPython was released (Jupyter Notebook is it's new name).

In 2002, Sweave, a popular function used to create notebooks in R was released. Sweave allowed users to combine graphics, text, and executable code. Sweave combined LaTeX, another markup language, with embedded R code. The common output file is PDF.

The issue with LaTeX, is that the markup language is a lot more complex then Markdown. Another common annoyance is that their were syntax adaptions when writing R in these documents, meaning, you couldn't use normal R syntax. This is because Sweave uses text string parsers, meaning that using a comma can be misinterpreted by Sweave to seperate values when it is used as a literal. However, it did fulfill the purpose of making research more transparent (more about this later).

I really wanted to show examples of code with LaTeX, but based on my research, it requires a different file format. So I'll show you the comparison between use of Markdown and LaTeX through "cheatsheets".



$\LaTeX\ 2_\varepsilon$ Cheat Sheet

Document classes

`book`	Default is two-sided.
`report`	No `\part` divisions.
`article`	No `\part` or `\chapter` divisions.
`letter`	Letter (?).
`slides`	Large sans-serif font.

Used at the very beginning of a document: `\documentclass{`*class*`}`. Use `\begin{document}` to start contents and `\end{document}` to end the document.

Common documentclass options

`10pt/11pt/12pt`	Font size.
`letterpaper/a4paper`	Paper size.
`twocolumn`	Use two columns.
`twoside`	Set margins for two-sided.
`landscape`	Landscape orientation. Must use `dvips -t landscape`.
`draft`	Double-space lines.

Usage: `\documentclass[`*opt,opt*`]{`*class*`}`.

Packages

`fullpage`	Use 1 inch margins.
`anysize`	Set margins: `\marginsize{`*l*`}{`*r*`}{`*t*`}{`*b*`}`.
`multicol`	Use *n* columns: `\begin{multicols}{`*n*`}`.
`latexsym`	Use LaTeX symbol font.
`graphicx`	Show image: `\includegraphics[width=`*x*`]{`*file*`}`.
`url`	Insert URL: `\url{`*http://...*`}`.

Use before `\begin{document}`. Usage: `\usepackage{`*package*`}`

Title

`\author{`*text*`}`	Author of document.
`\title{`*text*`}`	Title of document.
`\date{`*text*`}`	Date.

These commands go before `\begin{document}`. The declaration `\maketitle` goes at the top of the document.

Miscellaneous

`\pagestyle{empty}`	Empty header, footer and no page numbers.
`\tableofcontents`	Add a table of contents here.

Document structure

`\part{`*title*`}`	`\subsubsection{`*title*`}`
`\chapter{`*title*`}`	`\paragraph{`*title*`}`
`\section{`*title*`}`	`\subparagraph{`*title*`}`
`\subsection{`*title*`}`	

Use `\setcounter{secnumdepth}{`*x*`}` suppresses heading numbers of depth > *x*, where `chapter` has depth 0. Use a `*`, as in `\section*{`*title*`}`, to not number a particular item—these items will also not appear in the table of contents.

Text environments

`\begin{comment}`	Comment (not printed). Requires `verbatim` package.
`\begin{quote}`	Indented quotation block.
`\begin{quotation}`	Like `quote` with indented paragraphs.
`\begin{verse}`	Quotation block for verse.

Lists

`\begin{enumerate}`	Numbered list.
`\begin{itemize}`	Bulleted list.
`\begin{description}`	Description list.
`\item` *text*	Add an item.
`\item[`*x*`]` *text*	Use *x* instead of normal bullet or number. Required for descriptions.

References

`\label{`*marker*`}`	Set a marker for cross-reference, often of the form `\label{sec:item}`.
`\ref{`*marker*`}`	Give section/body number of marker.
`\pageref{`*marker*`}`	Give page number of marker.
`\footnote{`*text*`}`	Print footnote at bottom of page.

Floating bodies

`\begin{table}[`*place*`]`	Add numbered table.
`\begin{figure}[`*place*`]`	Add numbered figure.
`\begin{equation}[`*place*`]`	Add numbered equation.
`\caption{`*text*`}`	Caption for the body.

The *place* is a list valid placements for the body. `t`=top, `h`=here, `b`=bottom, `p`=separate page, `!`=place even if ugly. Captions and label markers should be within the environment.

Text properties

Font face

Command	*Declaration*	*Effect*
`\textrm{`*text*`}`	`{\rmfamily` *text*`}`	Roman family
`\textsf{`*text*`}`	`{\sffamily` *text*`}`	Sans serif family
`\texttt{`*text*`}`	`{\ttfamily` *text*`}`	`Typewriter family`
`\textmd{`*text*`}`	`{\mdseries` *text*`}`	Medium series
`\textbf{`*text*`}`	`{\bfseries` *text*`}`	**Bold series**
`\textup{`*text*`}`	`{\upshape` *text*`}`	Upright shape
`\textit{`*text*`}`	`{\itshape` *text*`}`	*Italic shape*
`\textsl{`*text*`}`	`{\slshape` *text*`}`	*Slanted shape*
`\textsc{`*text*`}`	`{\scshape` *text*`}`	SMALL CAPS SHAPE
`\emph{`*text*`}`	`{\em` *text*`}`	*Emphasized*
`\textnormal{`*text*`}`	`{\normalfont` *text*`}`	Document font
`\underline{`*text*`}`		Underline

The command (t*tt*t) form handles spacing better than the declaration (t*tt*t) form.

Font size

`\tiny`	tiny	`\Large`	Large
`\scriptsize`	scriptsize	`\LARGE`	LARGE
`\footnotesize`	footnotesize	`\huge`	huge
`\small`	small	`\Huge`	Huge
`\normalsize`	normalsize		
`\large`	large		

These are declarations and should be used in the form `{\small` ...`}`, or without braces to affect the entire document.

Verbatim text

`\begin{verbatim}`	Verbatim environment.
`\begin{verbatim*}`	Spaces are shown as ␣.
`\verb!text!`	Text between the delimiting characters (in this case '`!`') is verbatim.

Justification

Environment	*Declaration*
`\begin{center}`	`\centering`
`\begin{flushleft}`	`\raggedright`
`\begin{flushright}`	`\raggedleft`

Miscellaneous

`\linespread{`*x*`}` changes the line spacing by the multiplier *x*.

Text-mode symbols

Symbols

&	`\&`	_	`\_`	...	`\ldots`	•	`\textbullet`
$	`\$`	^	`\^{}`	|	`\textbar`	\	`\textbackslash`
%	`\%`	~	`\~{}`	#	`\#`	§	`\S`

Accents

ò	`` \`o ``	ó	`\'o`	ô	`\^o`	õ	`\~o`	ō	`\=o`
ȯ	`\.o`	ö	`\"o`	ǫ	`\c o`	ǒ	`\v o`	ő	`\H o`
ç	`\c c`	ọ	`\d o`	o̲	`\b o`	o͡o	`\t oo`	œ	`\oe`
Œ	`\OE`	æ	`\ae`	Æ	`\AE`	å	`\aa`	Å	`\AA`
ø	`\o`	Ø	`\O`	ł	`\l`	Ł	`\L`	ı	`\i`
ȷ	`\j`	¡	`` !` ``	¿	`` ?` ``				

Delimiters

'	`` ` ``	"	` `` `	{	`\{`	[	`[`	(	`(`	< `\textless`
'	`'`	"	`''`	}	`\}`	]	`]`	)	`)`	> `\textgreater`

Dashes

Name	*Source*	*Example*	*Usage*
hyphen	`-`	X-ray	In words.
en-dash	`--`	1–5	Between numbers.
em-dash	`---`	Yes—or no?	Punctuation.

Line and page breaks

`\\`	Begin new line without new paragraph.
`\\*`	Prohibit pagebreak after linebreak.
`\kill`	Don't print current line.
`\pagebreak`	Start new page.
`\noindent`	Do not indent current line.

Miscellaneous

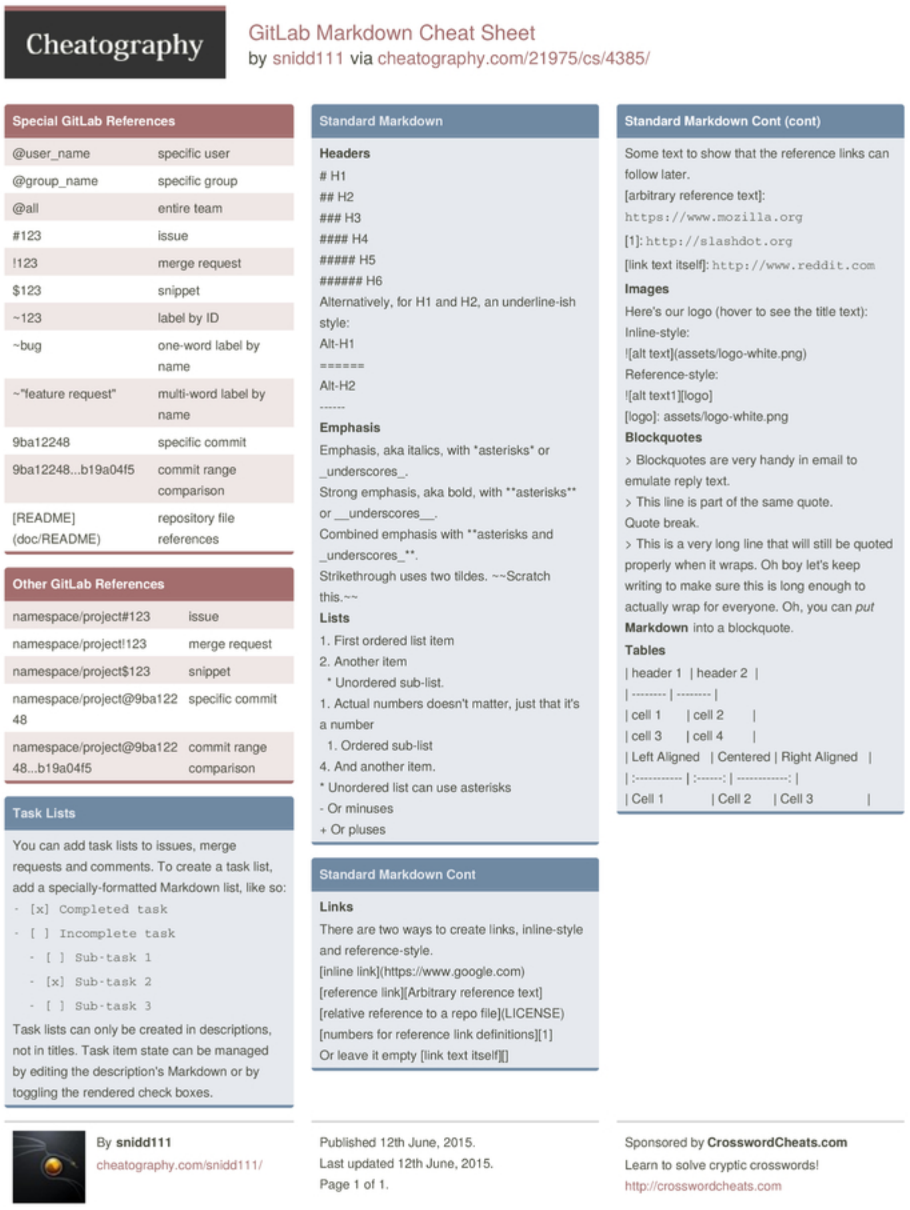
`\today`	March 28, 2017.
`$\sim$`	Prints ∼ instead of `\~{}`, which makes ˜.
`~`	Space, disallow linebreak (`W.J.~Clinton`).
`\@.`	Indicate that the `.` ends a sentence when following an uppercase letter.
`\hspace{`*l*`}`	Horizontal space of length *l* (Ex: l = `20pt`).
`\vspace{`*l*`}`	Vertical space of length *l*.
`\rule{`*w*`}{`*h*`}`	Line of width *w* and height *h*.

Tabular environments

`tabbing` environment

`\=` Set tab stop.	`\>` Go to tab stop.

Tab stops can be set on "invisible" lines with `\kill` at the end of the line. Normally `\\` is used to separate lines.



Cheatography

GitLab Markdown Cheat Sheet
by snidd111 via cheatography.com/21975/cs/4385/

Special GitLab References

@user_name	specific user
@group_name	specific group
@all	entire team
#123	issue
!123	merge request
$123	snippet
~123	label by ID
~bug	one-word label by name
~"feature request"	multi-word label by name
9ba12248	specific commit
9ba12248...b19a04f5	commit range comparison
[README] (doc/README)	repository file references

Other GitLab References

namespace/project#123	issue
namespace/project!123	merge request
namespace/project$123	snippet
namespace/project@9ba12248	specific commit
namespace/project@9ba12248...b19a04f5	commit range comparison

Task Lists

You can add task lists to issues, merge requests and comments. To create a task list, add a specially-formatted Markdown list, like so:

```
- [x] Completed task
- [ ] Incomplete task
  - [ ] Sub-task 1
  - [x] Sub-task 2
  - [ ] Sub-task 3
```

Task lists can only be created in descriptions, not in titles. Task item state can be managed by editing the description's Markdown or by toggling the rendered check boxes.

Standard Markdown

Headers

```
# H1
## H2
### H3
#### H4
##### H5
###### H6
```

Alternatively, for H1 and H2, an underline-ish style:

```
Alt-H1
======
Alt-H2
------
```

Emphasis

Emphasis, aka italics, with *asterisks* or _underscores_.
Strong emphasis, aka bold, with **asterisks** or __underscores__.
Combined emphasis with **asterisks and _underscores_**.
Strikethrough uses two tildes. ~~Scratch this.~~

Lists

```
1. First ordered list item
2. Another item
  * Unordered sub-list.
1. Actual numbers doesn't matter, just that it's a number
  1. Ordered sub-list
4. And another item.
* Unordered list can use asterisks
- Or minuses
+ Or pluses
```

Standard Markdown Cont

Links

There are two ways to create links, inline-style and reference-style.
[inline link](https://www.google.com)
[reference link][Arbitrary reference text]
[relative reference to a repo file](LICENSE)
[numbers for reference link definitions][1]
Or leave it empty [link text itself][]

Standard Markdown Cont (cont)

Some text to show that the reference links can follow later.
[arbitrary reference text]: `https://www.mozilla.org`
[1]: `http://slashdot.org`
[link text itself]: `http://www.reddit.com`

Images

Here's our logo (hover to see the title text):
Inline-style:
![alt text](assets/logo-white.png)
Reference-style:
![alt text1][logo]
[logo]: assets/logo-white.png

Blockquotes

> Blockquotes are very handy in email to emulate reply text.
> This line is part of the same quote.

Quote break.

> This is a very long line that will still be quoted properly when it wraps. Oh boy let's keep writing to make sure this is long enough to actually wrap for everyone. Oh, you can *put* **Markdown** into a blockquote.

Tables

```
| header 1  | header 2  |
| -------- | -------- |
| cell 1     | cell 2      |
| cell 3     | cell 4      |
| Left Aligned   | Centered | Right Aligned   |
| :---------- | :------: | -----------: |
| Cell 1           | Cell 2     | Cell 3            |
```

By **snidd111**
cheatography.com/snidd111/

Published 12th June, 2015.
Last updated 12th June, 2015.
Page 1 of 1.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

As you might have guessed, LaTeX is not a light-weight markup language. That is the major advantage of Markdown.

R Markdown

According to sources, R Markdown was created in 2012 with knitr by Yihui Xie. Knitr is a package that combines functionalities of pandoc, Sweave, cacheSweave, pgfSweave, weaver, R2HTML::RweaveHTML and more. It was created with the intention to solve some long-standing issues with LaTeX and Sweave. A lot of these issues that were resolved were regarding adding more functionalities to R code-chunks (such as allowing more than one figures printed per chunk) and simplifying it's use. Unfortunately, this is only how far research has taken me with the history of R Markdown.

R Markdown is a type of file (rmd) that is a markdown file with embedded, executable r code chunks. When "knitted". It is more appropriate to speak of knitr and corresponding programs directly rather than RMarkdown to understand how it is used under the hood.

KnitR takes the rmd file and executes all of the code chunks, and creates an md file. Then it is handed off to pandoc, which creates the final format of the md file.

Let's get to work!

First one must install knitr directly. One could use the function install.packages(knitr) in order to do this (inline code is omitted since knitr is already installed on my computer). Then, one can use RStudio to create a new rmd file. In this file they can make relative specifications of how they want their "knitted" file to look like.

Markdown

Markdown syntax is extremely important since *it can be used* **in any way** *i* ***you*** like. Since you are a Stat 133 student, I'm going to show you syntax that you might not know about rather than what you can learn in an rmarkdown tutorial. This is my favorite tutorial.

Strike-Through

```
~~you~~ I could use this to show prior assumptions.
```

~~you~~ I could use this to show prior assumptions.

No Space

```
$Nice font mang$
```

$Nicefontmang$

Footnotes

These can be used instead of parentheses in order to provide information without sidetracking the reader.

```
I wish I could have told you about this before you started the assignment[^1]
Start earlier next time.[^2]


[^1]: They look nice in the knitted file.
[^2]: Be an A student.
```

I wish I could have told you about this before you started the assignment.[1] Start earlier next time.[2]

Check the bottom of the file.

Definition Lists

These are useful in research documents.

```
Money
:    Used as a measurement of value between products so to trade them instead of bartering.

Love
:    Something that can't be bought...
```

Money
Used as a measurement of value between products so to trade them instead of bartering.
Love
Something that can't be bought…

Horizontal Dividers

These can be used to divide parts of a report.

```
***

---

___
```

It turns out that in my research, there isn't a lot of markdown syntax beyond what we have already spent time with. But… I did find out that markdown supports direct use of html. Check some of this out.

If you want to change the font of your words.. you can try this for your next assignment:

```
<p style="color:violet;">This looks pretty</p>
```

This looks pretty

You can also create buttons.

```
<button class="btn btn-primary btn-lg">Do not press this button!</button>
```

Do not press this button!

There are many other features available through html. They are just difficult to use hahahahaa.

R Markdown Features

YAML Header

YAML is actually a language that is specifically used for configuring files. It was created by Clark Evans in 2001.

There is obviously the Title, date, and author, but there are many ways you can use the YAML to change the output of the file. I will cover some file types that we haven't tried. You can create:

- ioslides_presentation - Another Slide Presentation
- md_document - different then github document
- odt_document - side by side interface document.
- pdf_document - Pdf document
- rtf_document - This creates a side by side interface document.
- slidy_presentation - a Slide Presentation
- word_document - Yeah…

There are other features you can use in a YAML header to configure the output, but a lot of them seemed more advanced then I could understand.

Code Chunks

To create a coding chunk, one must type three backwards apostrophies, bracket (with some language), hit return twice and then type three more backwards apostrophies.

Example of brackets: {python}

```
If you are wondering how to put blank text that isn't influenced
by markdown, delete the bracket with the language inserted in it
```

Rmarkdown gives access to the following languages for code chunks:
- Python
- SQL
- Bash
- CSS
- Javascript
- RCPP
- Stan

You can also put special commands within the brackets in order to change the output. Some included that haven't been covered in class:
- Include. When this is set to false, you can evaluate code without printing the output. This is nice when you are trying to restrict output to graphical figures.
- Fig options. You can control the appearance of your resulting graph. - Size. You can set the size of the output using this parameter.
- Background. You can change the color of your code using this parameter. This is nice when grey gets old.
- Dev. This can be used to record the file output of a plot created.

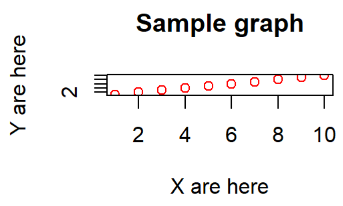
```
1 + 2
```

```
## [1] 3
```

```
1 + 2
```

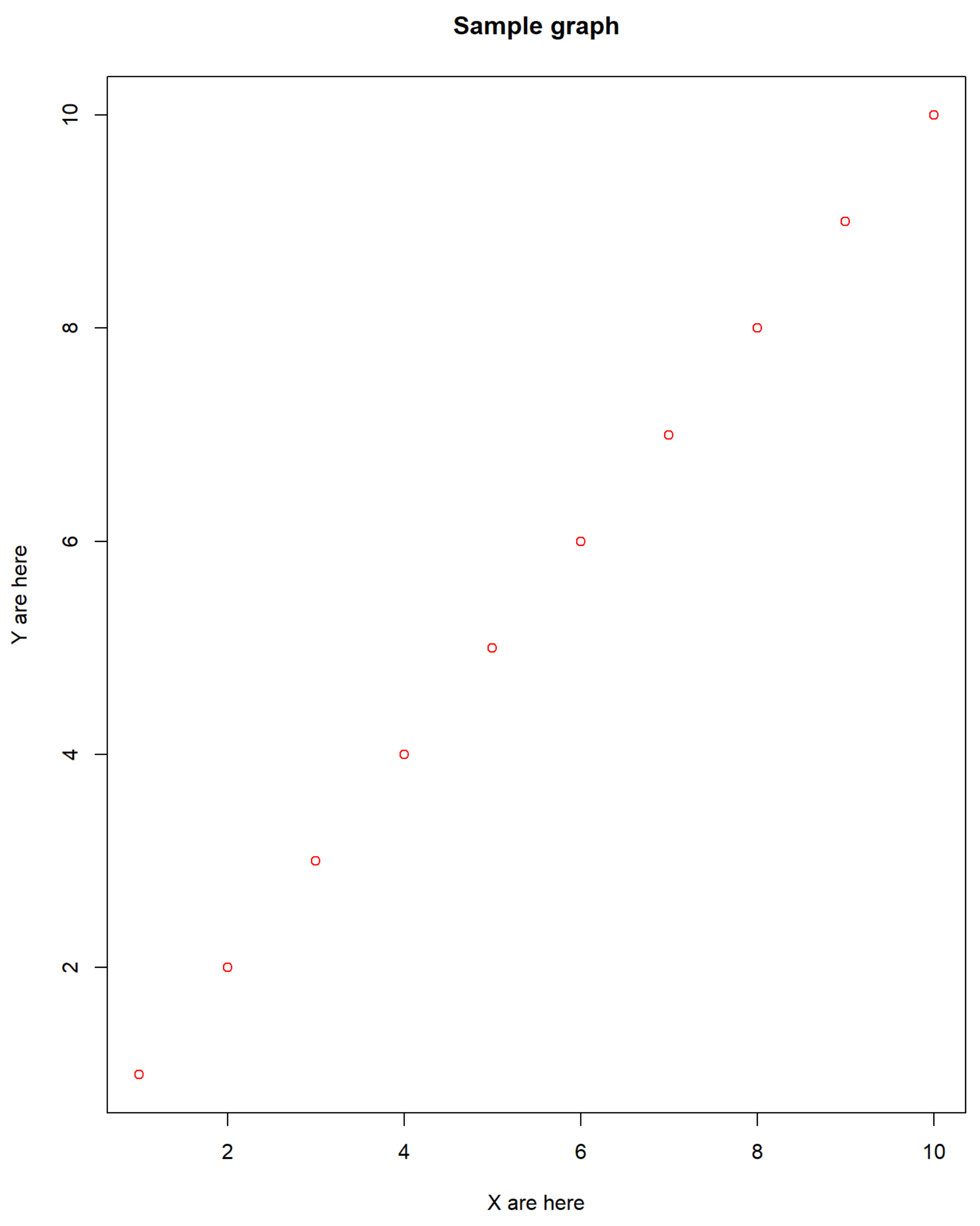
```
## [1] 3
```

```
x = c(1:10)
y = c(1:10)
plot(x,y, xlab = "X are here", ylab = "Y are here", main = "Sample graph", col = 2)
```



```
#fig height and width are set to 2 and 3
```

```
x = c(1:10)
y = c(1:10)
plot(x,y, xlab = "X are here", ylab = "Y are here", main = "Sample graph", col = 2)
```



```
#fig height and width are set to 10 and 8
```

R Markdown in Practice

According to sources, R markdown is good for the following purposes:
- Communicating information to decision makers who don't fully understand the analysis.
- Collaboration with other data scientists.
- A notebook, it can show you how you approach problems for future reference.

Since R is a language geared towards data analysis and statistics, it is commonly used in what is called reproducible research. Reproducible research is a type of research that has three major intentions (according to sources): - All methods are reported
- All data and files used for the analysis are (publicly) available
- The process of analyzing raw data is well reported and preserved

R Markdown is extremely valueable for this type of research, since one can show all data, show all analysis based in understanding the data and all code used in processing the data.

Takehome Message

The takehome message is very simple. R Markdown is extremely easy to understand and use, and has much reason to continue to be used by the data science and research community. Take it from me, it was difficult finding information that we hadn't already learned in class.

References:

Broman, Karl. "Knitr with R Markdown." Sitewide ATOM,
kbroman.org/knitr_knutshell/pages/Rmarkdown.html
Karlijn, Willems. "Jupyter And R Markdown: Notebooks With R." DataCamp Community, DataCamp,
www.datacamp.com/community/blog/jupyter-notebook-r#markdown
"R for Researchers: R Markdown", Social Science Computing Cooperative, 16 Apr. 2015,
www.ssc.wisc.edu/sscc/pubs/RFR/RFR_RMarkdown.html
"R Markdown." Github, Github, 3 May 2017,

github.com/hadley/r4ds/blob/master/rmarkdown.Rmd
“Sweave.” Wikipedia, Wikimedia Foundation, 17 Oct. 2017,
en.wikipedia.org/wiki/Sweave
“LaTeX.” Wikipedia, Wikimedia Foundation, 11 Oct. 2017,
en.wikipedia.org/wiki/LaTeX#Licensing
Latex Cheat Sheet. wch.github.io/latexsheet/latexsheet-0.png “What is reproducible research?” R-Bloggers, 22 June 2016,
www.r-bloggers.com/what-is-reproducible-research/ Snid111. “Github Markdown Cheat Sheet.” Cheatography, Crosswordcheatsheets.com, 12 June 2015, media.cheatography.com/storage/thumb/snidd111_gitlab-markdown.750.jpg?last=1463106685 Xie, Yihui. “Knitr.” Yihui Xie,
yihui.name/knitr/
Xie, Yihui. “Sweave - Transition from Sweave to Knitr.” Yihui Xie, 24 Feb. 2012,
yihui.name/knitr/demo/sweave/

---

1. They look nice in the knitted file.↩
2. [illegible]↩

Loading [MathJax]/jax/output/HTML-CSS/jax.js