Intermediate Programming in R Part I

Justin Post

August 10-11, 2017

Course Schedule

Daily agenda:

- 10-11:10 Session
- 10-minute break
- 11:20-12:30 Session
- 12:30-1:45 Lunch
- 1:45-2:55 Session
- 10-minute break
- 3:05-4:15 Session

What do we want to be able to do?

- Communicate findings effectively
- Encompass and document entire data analysis process
- Document findings
- Make process reproducible
- Share process

Where do we start?

- Review of Key Concepts
- R Markdown Basics
  - Code Chunks
  - Images/Equations/Misc.
- R Markdown Options
  - Documents: PDF, HTML
  - Presentations: Slides
  - Interactive Components
- R Shiny Applications/Presentations

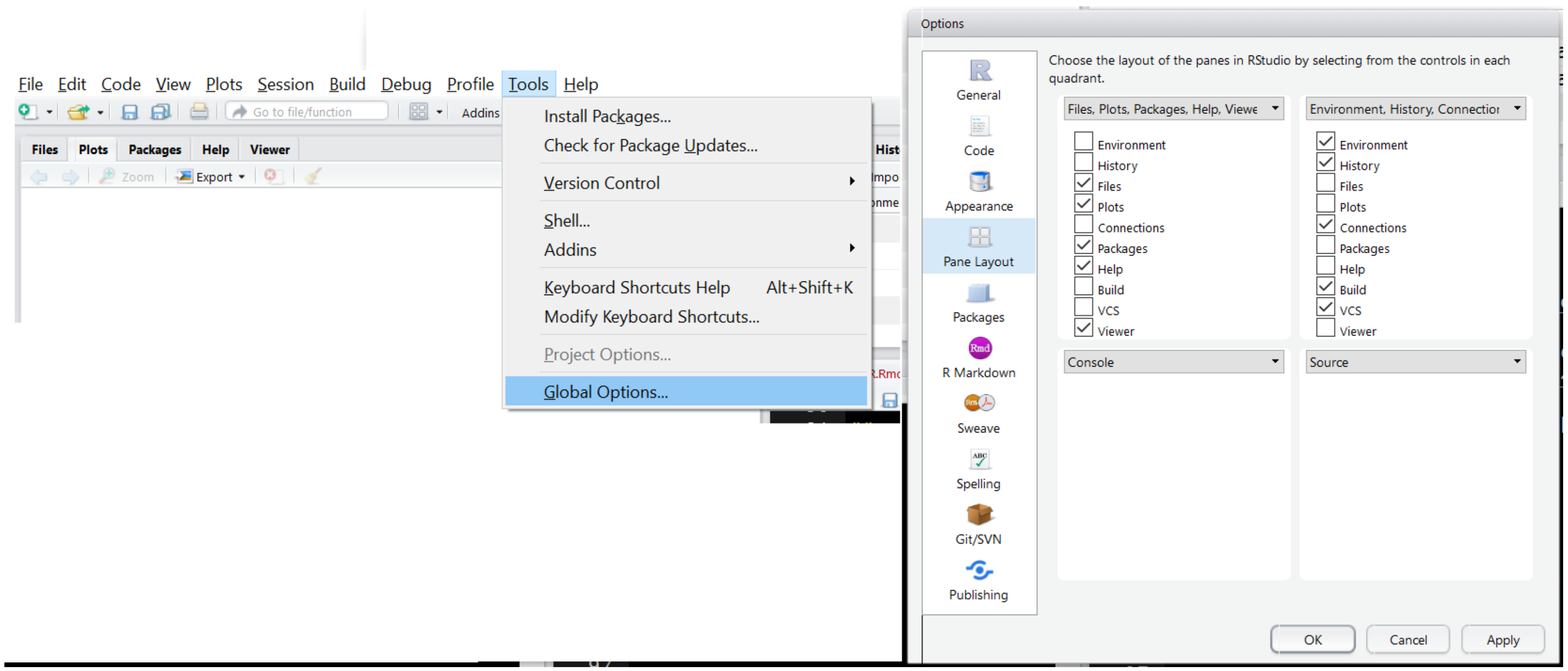
Review of Concepts

R Studio

- Great integrated development environment (IDE)
- Four main 'areas' we'll use
  - Scripting and Viewing Area
  - Workspace/History
  - Files/Plots/Help
  - Console

Review of Concepts

R Studio - Can rearrange panes



Choose the layout of the panes in RStudio by selecting from the controls in each quadrant.

Files, Plots, Packages, Help, Viewe

- [ ] Environment
- [ ] History
- [✓] Files
- [✓] Plots
- [ ] Connections
- [✓] Packages
- [✓] Help
- [ ] Build
- [ ] VCS
- [✓] Viewer

Environment, History, Connectior

- [✓] Environment
- [✓] History
- [ ] Files
- [ ] Plots
- [✓] Connections
- [ ] Packages
- [ ] Help
- [✓] Build
- [✓] VCS
- [ ] Viewer

Console

Source

- Global options –> Appearance allows font/background changes

6/93

Review of Concepts

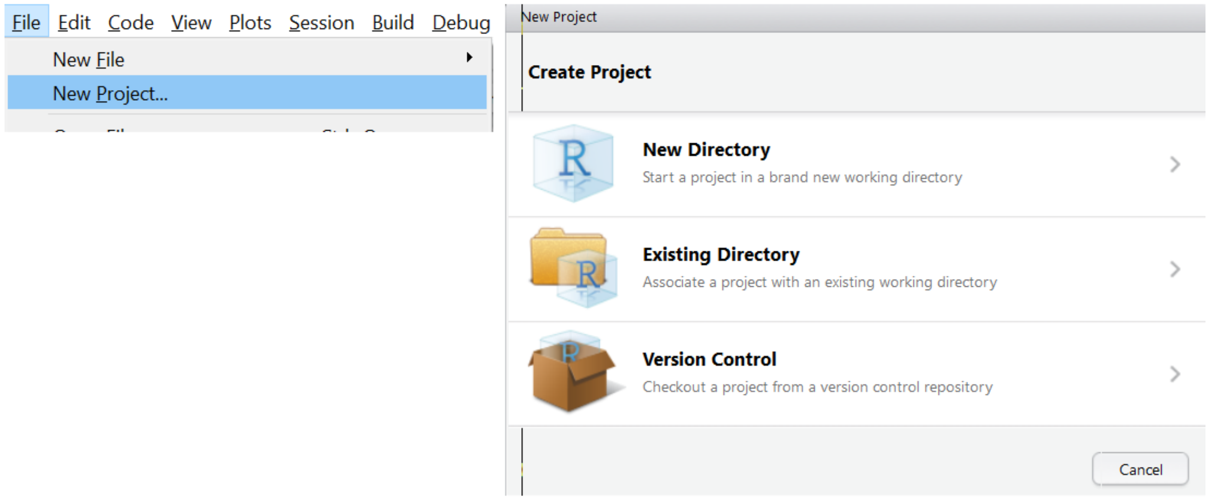
R Studio - Project

- Often have many files associated with each analysis
- Keeping different undertakings separate can be difficult!
- Can use "Project" feature in R Studio
- Provides straightforward way to divide your work into multiple contexts. Each with their own:
  - Working directory
  - Workspace
  - History
  - Source documents

Review of Concepts

R Studio - Project

- Easy to create!


File Edit Code View Plots Session Build Debug
New File
New Project...
New Project
Create Project
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel

- Can save workspace, etc. and pick up right where you left off!
- Work on multiple projects at once

Review of Concepts

Data Frames

- Best R object for data sets
- Collection (list) of vectors of the same **length**

```
iris<-tbl_df(iris); iris
```

```
## # A tibble: 150 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl>  <fctr>
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## # ... with 145 more rows
```

Review of Concepts

Data Frames

- Accessing elements: multiple ways

```
iris[1:4, 2:4]
```

```
## # A tibble: 4 x 3
##   Sepal.Width Petal.Length Petal.Width
##         <dbl>        <dbl>       <dbl>
## 1         3.5          1.4         0.2
## 2         3.0          1.4         0.2
## 3         3.2          1.3         0.2
## 4         3.1          1.5         0.2
```

Review of Concepts

Data Frames

- Accessing elements: multiple ways

```
iris[1, ]
```

```
## # A tibble: 1 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl>  <fctr>
## 1          5.1         3.5          1.4         0.2  setosa
```

Review of Concepts

Data Frames

- Accessing elements: multiple ways

```
iris[ , c("Sepal.Length", "Species")]
```

```
## # A tibble: 150 x 2
##   Sepal.Length Species
##          <dbl>  <fctr>
## 1          5.1  setosa
## 2          4.9  setosa
## 3          4.7  setosa
## 4          4.6  setosa
## 5          5.0  setosa
## # ... with 145 more rows
```

Review of Concepts

Data Frames

- Accessing elements: multiple ways

```
iris$Sepal.Length
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4
##  [18] 5.1 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5
##  [35] 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0
##  [52] 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8
##  [69] 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4
##  [86] 6.0 6.7 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8
## [103] 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7
## [120] 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7
## [137] 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
```

Review of Concepts

Packages - Many ways to accomplish the same thing in R

- How to choose?
  - Want 'fast' code
  - Want 'easy' syntax
  - Good default settings on functions
- Base R has reasonable defaults and syntax but functions are slow
- "TidyVerse" - collection of R packages that share common philosophies and are designed to work together!
  - Very efficient code
  - Common syntax

Review of Concepts

- If not installed (downloaded) on computer

```
install.packages("tidyverse")
```

Review of Concepts

- Once installed, `library()` or `require()` to load

```
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## filter(): dplyr, stats
## lag():    dplyr, stats
```

Tidyverse Syntax

- All packages have similar syntax! All work on `tibbles` (epecial data frames)
- Convert any data frame (or matrix) to a tibble using `tbl_df()`
- Nice printing properties (can sometimes cause issues though)
- Most packages have syntax: `function(data.frame, options)`

Tidyverse Syntax

- All packages have similar syntax! All work on `tibbles` (epecial data frames)
- Convert any data frame (or matrix) to a tibble using `tbl_df()`
- Nice printing properties (can sometimes cause issues though)
- Syntax: `function(data.frame, options)`
- Examples:

```
select(iris, Sepal.Width)
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length)) + geom_point()
```

Review of Concepts

- Read in most any type of data with
  - readr (.csv, delimited data)
  - readxl (.xls, .xlsx)
  - haven (.sav, .dta, .sas7bdat)

```r
#read in data (readr, readxl, haven packages)
votingData <- read_csv("https://raw.githubusercontent.com/
                       jbpost2/IntermediateR/master/datasets/counties.csv")
```

Review of Concepts

```
votingData
```

```
## # A tibble: 3,141 x 20
##   region  county state   msa  pmsa pop.density   pop pop.change age6574
##    <chr>   <chr> <chr> <int> <int>       <int> <int>      <dbl>   <dbl>
## 1  South Autauga    AL  5240    NA          61 34222       11.9     5.7
## 2  South Baldwin    AL  5160    NA          67 98280       35.4     9.2
## 3  South Barbour    AL    NA    NA          29 25417        2.0     8.2
## 4  South    Bibb    AL    NA    NA          28 16576        9.2     6.7
## 5  South  Blount    AL  1000    NA          62 39248       10.6     7.4
## # ... with 3,136 more rows, and 11 more variables: age75 <dbl>,
## #   crime <int>, college <dbl>, income <int>, farm <dbl>, democrat <dbl>,
## #   republican <dbl>, Perot <dbl>, white <dbl>, black <dbl>, turnout <dbl>
```

20/93

Review of Concepts

Piping or Chaining

- Applying multiple functions: nesting hard to parse!
- Piping or Chaining with %>% operator helps

Review of Concepts

Piping or Chaining

- Applying multiple functions: nesting hard to parse!
- Piping or Chaining with %>% operator helps
- If `dplyr` or `magrittr` package loaded, can use anywhere

```
#Consider
votingData %>%
  filter((state == "NC") & (college > 20)) %>%
  select(county, msa, pop.density:turnout) %>%
  arrange(college, desc(turnout))
#vs
arrange(select(filter(votingData, ((state == "NC") & (college > 20))),
               county, msa, pop.density:turnout), college, desc(turnout))
```

Review of Concepts

Piping or Chaining

```
votingData %>%
  filter((state == "NC") & (college > 20)) %>%
  select(county, msa, pop.density:turnout) %>%
  arrange(college, desc(turnout))
```

```
## # A tibble: 11 x 17
##         county   msa pop.density    pop pop.change age6574 age75 crime
##          <chr> <int>       <int>  <int>      <dbl>   <dbl> <dbl> <int>
##  1        Polk    NA          63  14416       14.7    13.7  10.9  1802
##  2 New Hanover  9200         643 120284       23.5     7.8   4.7  9778
##  3        Dare    NA          62  22746       77.4     8.3   4.1  8315
##  4        Pitt  3150         173 107924       24.7     6.0   3.9  4214
##  5     Forsyth  3120         661 265878       11.2     7.1   5.2  7976
##  6    Guilford  3120         550 347420       12.8     7.0   4.9  7990
##  7     Watauga    NA         122  36952       20.0     6.3   4.3  2862
##  8 Mecklenburg  1520        1020 511433       33.0     5.7   3.7 11154
##  9      Durham  6640         647 181835       23.5     6.1   4.6  8375
## 10        Wake  6640         548 423380       51.7     4.7   3.1  6057
```

Review of Concepts

Plotting

- R great for plotting
- We'll use `ggplot2` in `tidyverse`! cheatsheet
- Needs: Data Frame
- Aesthetic (aes) - maps variables to properties of geom
    - Ex: size, color, and x, y location(s)
- Geom layer(s) (visualizaton type(s))
- Coordinate system (mostly use Cartesian plane)
- Optional: Stat layer, titles, etc.

Review of Concepts

`ggplot2` needs and syntax

Needs:

- Data Frame
- Aesthetic (aes) - maps variables to properties of geom
- Geom layer(s) (visualizaton type(s))
- Optional: Stat layer, titles, etc.
- Syntax:

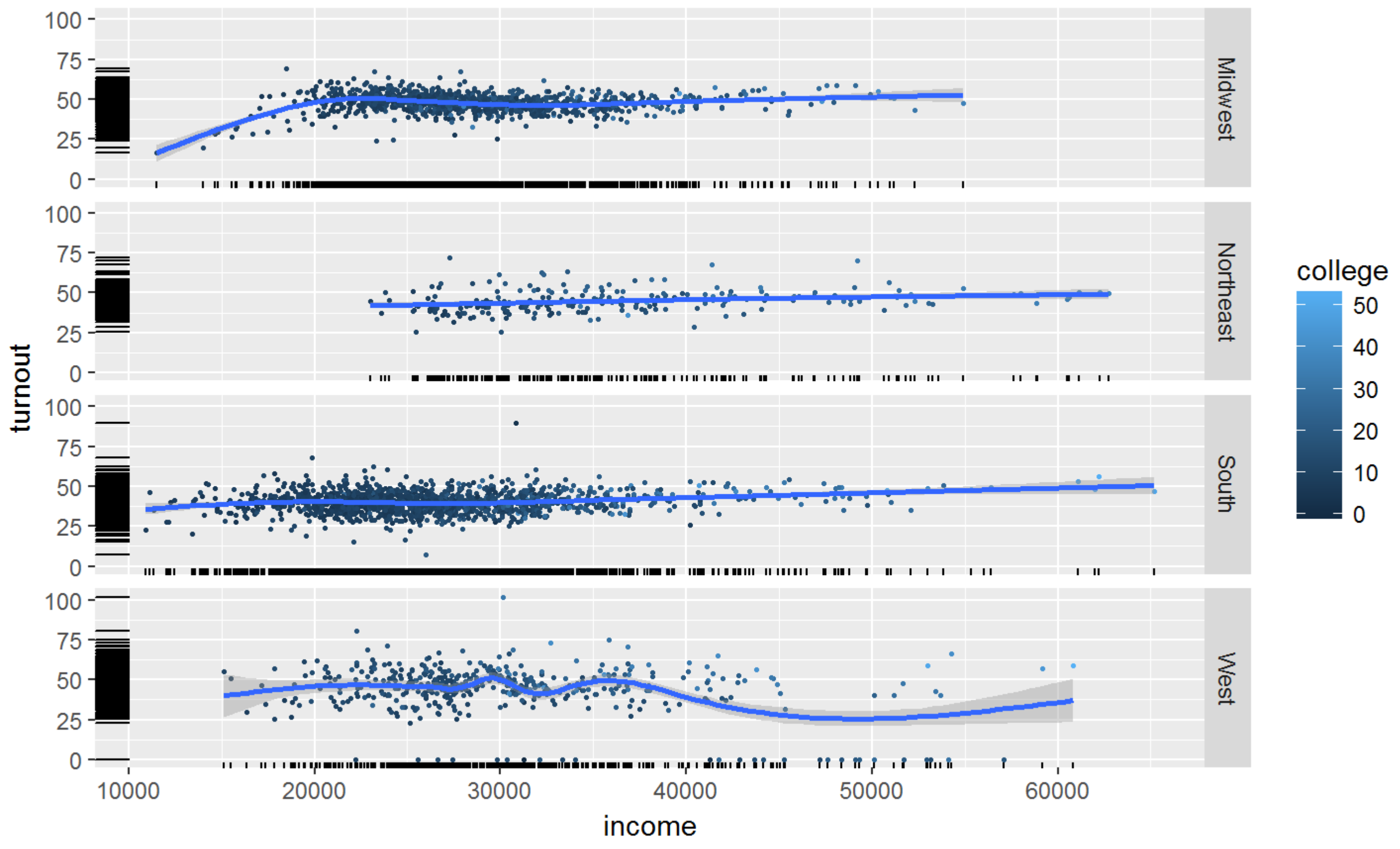
```
g <- ggplot(dataframe,aes(x = , y = , ...))
g + geom_type(...) +
  stat_type(...) +
  labs(...)
```

Review of Concepts

- Settings that depend on a variable go in `aes`

```
g <- ggplot(votingData, aes(x = income, y = turnout)) +
  geom_point(size = 0.5, aes(color = college)) +
  geom_smooth() +
  geom_rug() +
  facet_grid(region ~ .)
g
```

Review of Concepts



27/93

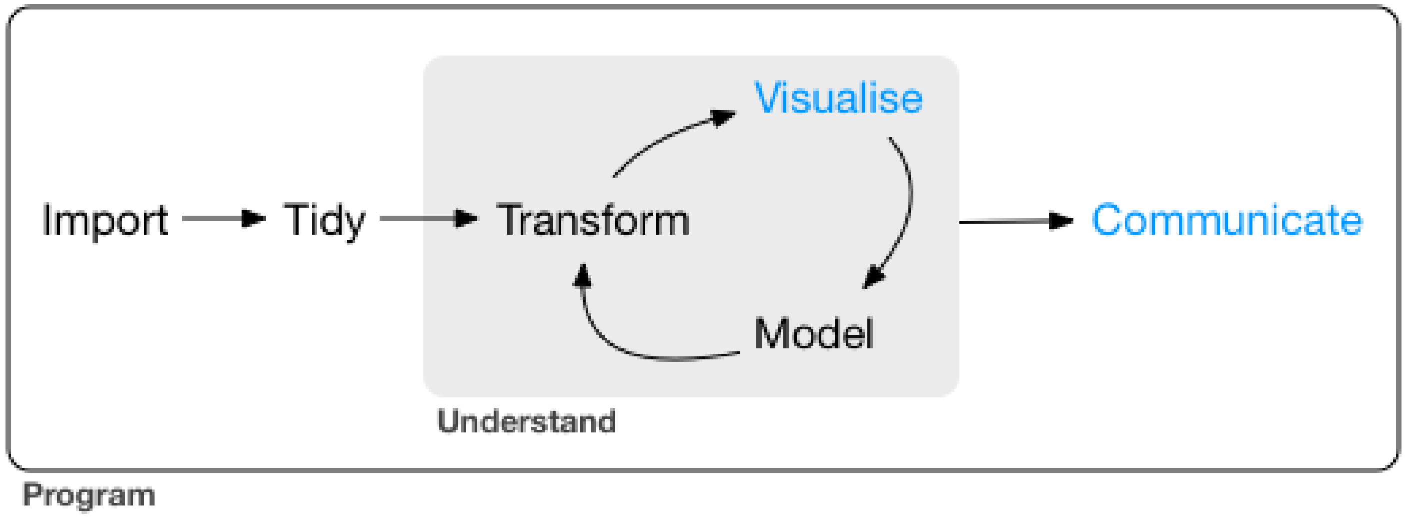
What do we want to be able to do?

- Communicate findings effectively
- Encompass and document entire data analysis process
- Document findings
- Make reproducible process
- Share process

Where do we start?

- Review of Key Concepts
- R Markdown Basics
  - Code Chunks
  - Images/Equations/Misc.
- R Markdown Options
  - Documents: PDF, HTML
  - Presentations: Slides
  - Interactive Components
- R Shiny Applications/Presentations

R Markdown Basics



(From R for Data Science)

R Markdown Basics

- Can read data into R
- Know how to manipulate it
- Likely know best ways to model and visualize it
- Doesnâ□□t matter how great your analysis is unless you can explain it to others :)
- Need to communicate results effectively

R Markdown Basics

What is Markdown?

- Formatting syntax for authoring HTML, PDF, slide shows, books, and more.
- Digitial "Notebook": Program that weaves word processing and code. Example
- Can do interactive documents!

R Markdown Basics

How to use Markdown?

Designed to be used in three ways (R for Data Science)

- Communicating to decision makers (focus on conclusions not code)
- Collaborating with other data scientists (including future you!)
- As environment to do data science (documents what you did and what you were thinking)

33/93

R Markdown Basics

Error: Cannot load file https://raw.githubusercontent.com/jbpost2/IntermediateR/master/video/Markdown.mp4

R Markdown Basics

Examples of markdown documents

R Markdown Basics

Verbage

- Most have heard of HTML (HyperText Mark-up Language)
  - Write plain text that the browser interprets and renders

R Markdown Basics

Verbage

- Most have heard of HTML (HyperText Mark-up Language)
    - Write plain text that the browser interprets and renders
- Markdown is a specific markup language
    - Easier syntax
    - Not as powerful
- Any plain text file with .Rmd extension can be used

R Markdown Basics

R Markdown Basics

R Markdown file contains three important types of content:

1. (Optional) YAML header surrounded by `---`s

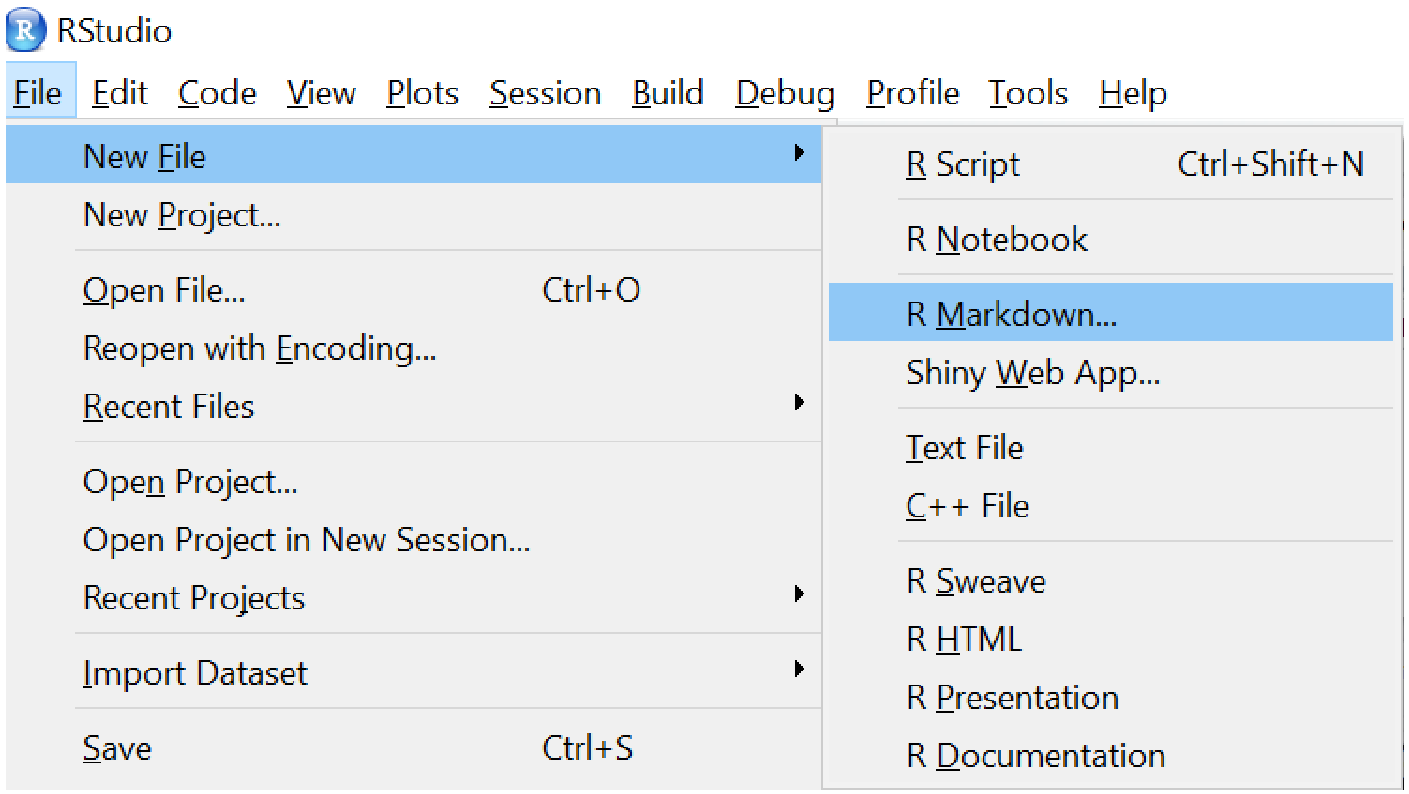
1. Chunks of R code surrounded by ```` ``` ````

1. Text mixed with simple text formatting like # heading and *italics*

R Markdown Basics

Creating an R Markdown Document

- R Studio makes it easy!

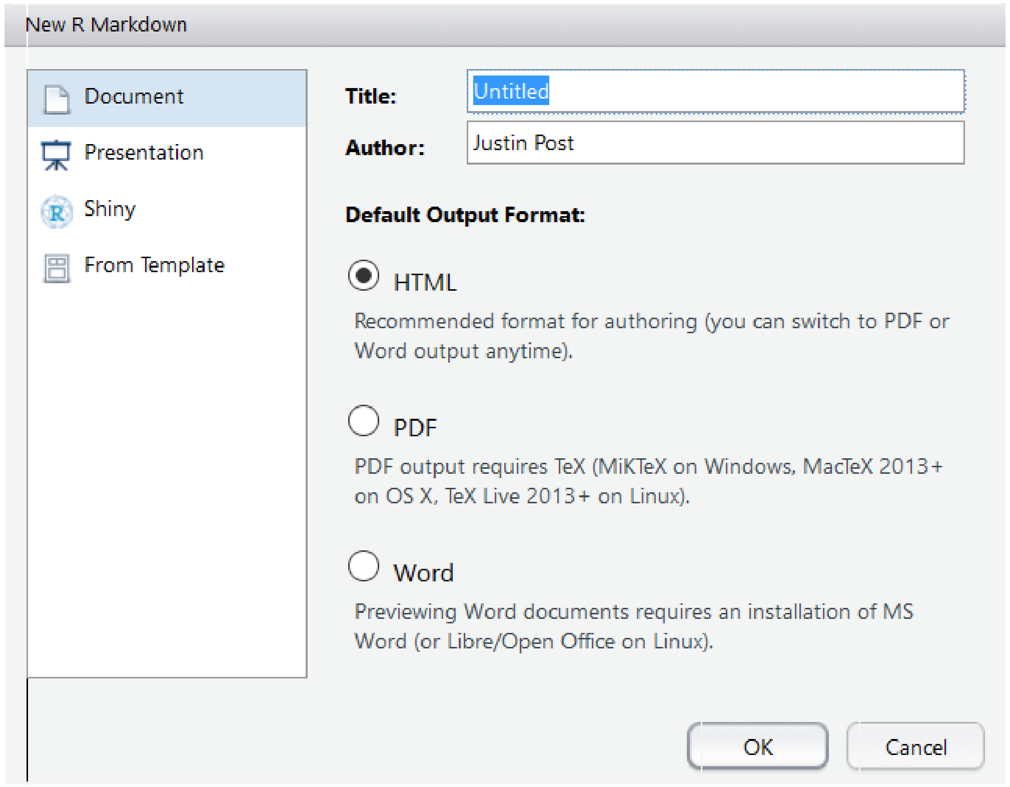


39/93

R Markdown Basics

Creating an R Markdown Document

- Commonly used document types can be created



Title: Untitled

Author: Justin Post

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).

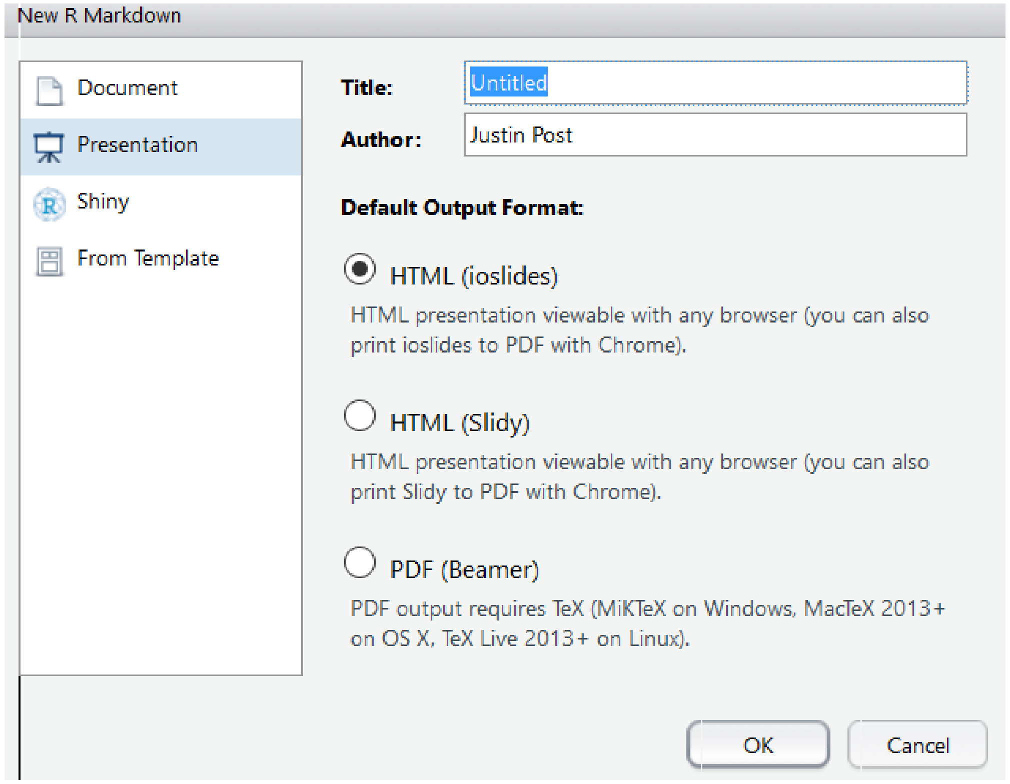
Word
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

40/93

R Markdown Basics

Creating an R Markdown Document

- Slide presentations



Title: Untitled

Author: Justin Post

Default Output Format:

[✓ HTML (ioslides)]
HTML presentation viewable with any browser (you can also print ioslides to PDF with Chrome).

HTML (Slidy)
HTML presentation viewable with any browser (you can also print Slidy to PDF with Chrome).

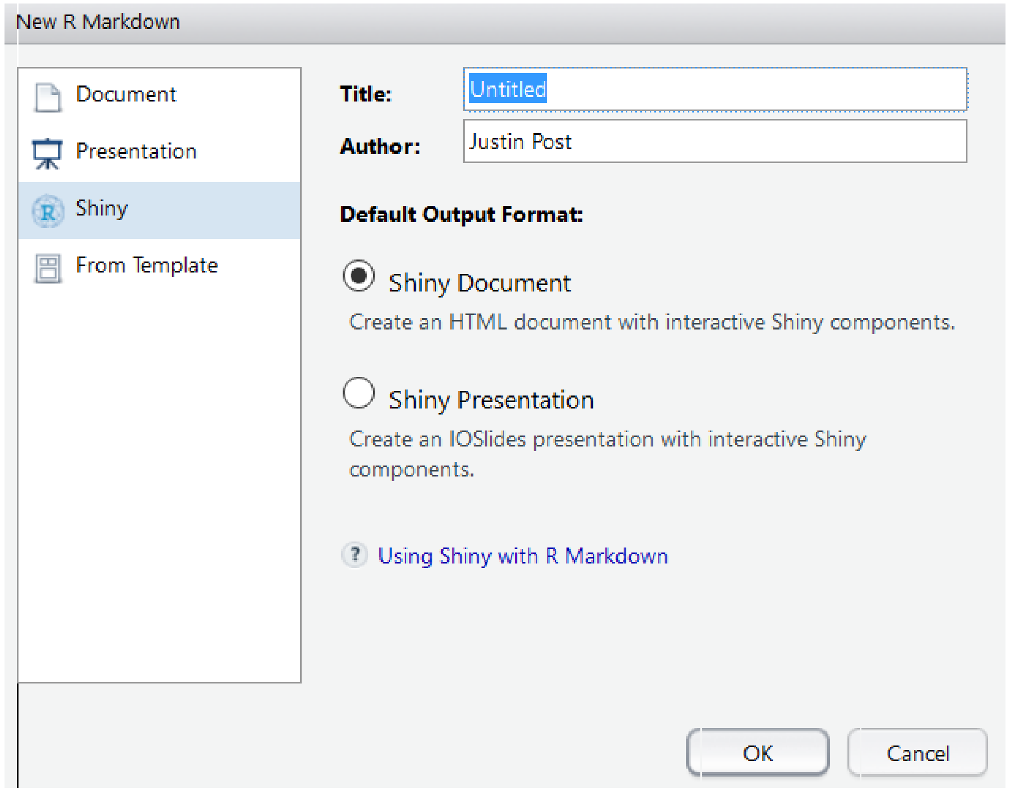
PDF (Beamer)
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

41/93

R Markdown Basics

Creating an R Markdown Document

- Truly Interactive Documents/Pages (requires R backend)



Title: Untitled

Author: Justin Post

Default Output Format:

[✓ Shiny Document]
Create an HTML document with interactive Shiny components.

Shiny Presentation
Create an IOSlides presentation with interactive Shiny components.

Using Shiny with R Markdown

R Markdown Basics

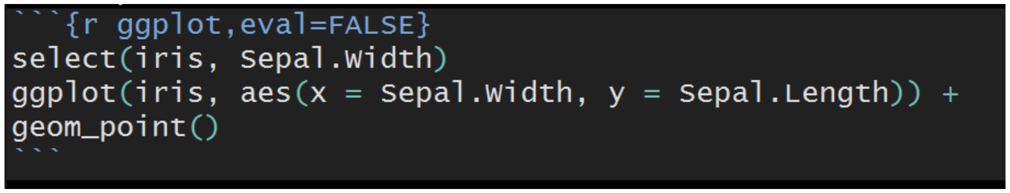
- Create an HTML Markdown document!

```
---
title: "Untitled"
author: "Justin Post"
date: "August 10, 2017"
output: html_document
---
```

- Top section: YAML header
- Define settings for document
- Author, Title, etc.
- Output type/Options

R Markdown Basics

- Below YAML header: 'r chunk'



~~~
```{r ggplot,eval=FALSE}
select(iris, Sepal.Width)
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length)) +
geom_point()
```
~~~

- Start code chunk by typing it out or with CTRL/CMD + Alt + I
- Code will be executed when document is created
- Specify options about individual chunk here

R Markdown Basics

- Below code chunk is plain text with markdown sytnax

```
## R Markdown

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>.

When you click the **Knit** button a document will be generated that
includes both content as well as the output of any embedded R code
chunks within the document.
```

- When file created, "##" becomes a header, "<…>" a link, and "**…**" bold font

R Markdown Basics

R Markdown

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.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

Where do we go from here?

- Figure out markdown syntax
- Look at "Notebook" feature
- Check options for code chunks
- Automate some things
- Change type of output
- Work with interactivity (shiny)

R Markdown Syntax

You can include:

- Plain text
- End a line with two spaces to start a new paragraph
  - Line breaks are not always added when you return!
  - Two spaces and a return drop marked up text down.
  - Can specify `<br>` as a line break
- `*italics*` *and* `_italics_`
- `**bold**` **and** `__bold__`
- `superscript^2^` becomes superscript^2
- `~~strikethrough~~` becomes ~~strikethrough~~

R Markdown Syntax

- `[link](https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf)` becomes link
- `# Header 1` becomes a large font header
- `## Header 2` becomes a slightly smaller font header
- Goes to 6 headers
- Use of headers can automatically create a Table of Contents!
- Include an image: `![](path/to/file.png)`
- \`code\` becomes `code`

R Markdown Syntax

- Can do lists: be sure to end each line with two spaces!
- Indent sub lists two spaces (I often do four for both)

```
* unordered list
* item 2
  + sub-item 1
  + sub-item 2

1. ordered list
2. item 2
  + sub-item 1
  + sub-item 2
```

- unordered list
- item 2
    - sub-item 1
    - sub-item 2

1. ordered list
2. item 2
    - sub-item 1
    - sub-item 2

R Markdown Syntax

- Can include nice tables

```
Table Header  | Second Header | Col 3
------------- | ------------- | -----------
Table Cell    | Cell (1, 2)   | Cell (1, 3)
Cell (2, 1)   | Cell (2, 2)   | Cell (2, 3)
```

Table Header	Second Header	Col 3
Table Cell	Cell (1, 2)	Cell (1, 3)
Cell (2, 1)	Cell (2, 2)	Cell (2, 3)

51/93

Activity

- **Formatting Text Activity** instructions available on web
- Work in small groups
- Ask questions! TAs and I will float about the room
- Feel free to ask questions about anything you didn't understand as well!

What do we want to be able to do?

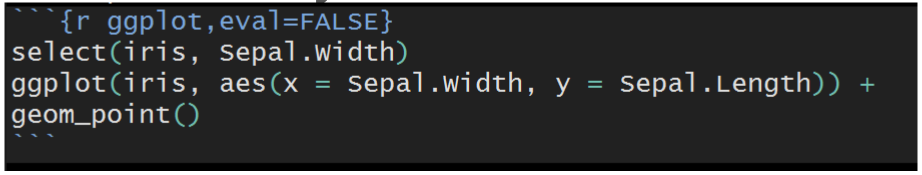
- Communicate findings effectively
- Encompass and document entire data analysis process
- Document findings
- Make process reproducible
- Share process

Where are we at?

- Review of Key Concepts
- R Markdown Basics
  - **Code Chunks**
  - **Images/Equations/Misc.**
- R Markdown Options
  - Documents: PDF, HTML
  - Presentations: Slides
  - Interactive Components
- R Shiny Applications/Presentations

Code Chunks

We've already seen how to include an R code chunk:



~~~
```{r ggplot,eval=FALSE}
select(iris, Sepal.Width)
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length)) +
geom_point()
```
~~~

- Add chunk via typing
  ```{r}
  code
  ```
- or `Ctrl/Cmd + Alt + I`
- Any R code can go into the chunk

55/93

Notebook Functionality

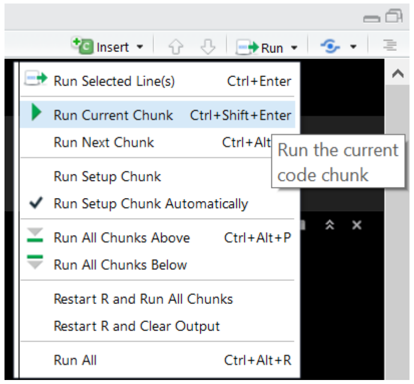
Data science notebook - virtual notebook environment used for literate programming

- Pairs the functionality of word processing software with a programming language
- Rendered markdown document captures R code and process
- R Markdown brings together the console and the script editor too!
- Blurs line between interactive exploration and long-term code capture.

Notebook Functionality

Within a chunk:

- Execute code with `Cmd/Ctrl + Shift + Enter` or with "Run"


Insert
Run
Run Selected Line(s) Ctrl+Enter
Run Current Chunk Ctrl+Shift+Enter
Run Next Chunk Ctrl+Alt
Run the current code chunk
Run Setup Chunk
Run Setup Chunk Automatically
Run All Chunks Above Ctrl+Alt+P
Run All Chunks Below
Restart R and Run All Chunks
Restart R and Clear Output
Run All Ctrl+Alt+R

- Results show up in editor!

57/93

Notebook Functionality

- Allows for quick iteration within a chunk: editing and re-executing - when you are happy, you move on and start a new chunk.
- Go back to markdown template document, execute code chunk in-line
- Can run all code chunks with `Ctrl/Cmd + Alt + R`
- Can develop code and record your thoughts - similar to classic lab notebook in the physical sciences

Back to Code Chunks

- Many options depending on chunk purpose!
- Can hide/show code with `echo = FALSE/TRUE`
- Can choose if code is evaluated with `eval = TRUE/FALSE`
- `Include = FALSE` is equivalent to `echo = FALSE, eval = TRUE`
- Useful for set-up code (usually first chunk after YAML header)

Code Chunks

- `message = TRUE/FALSE` and `warning = TRUE/FALSE` can turn on/off displaying messages/warnings
- `error = TRUE` allows file to be created with code that has an error
- Code can be added in line: Ex: Iris has 150 observations
- Added by beginning with back-tick `r` and ending with a back-tick: Iris has `` `r `` `length(iris$Sepal.Length)` `` ` ``

Code Chunks

Meantioned using `Include = FALSE` for 'set-up code' (usually first chunk after YAML header)

- Can set global options for chunks
- Allows for easy change of audience!

```
opts_chunk$set(echo = FALSE, eval = TRUE, warning = FALSE)
```

Code Chunks

Can name code chunks to help organization!

- When calling a chunk, add name after `r`

~~~
```{r name-of-chunk, options…}
code
```
~~~

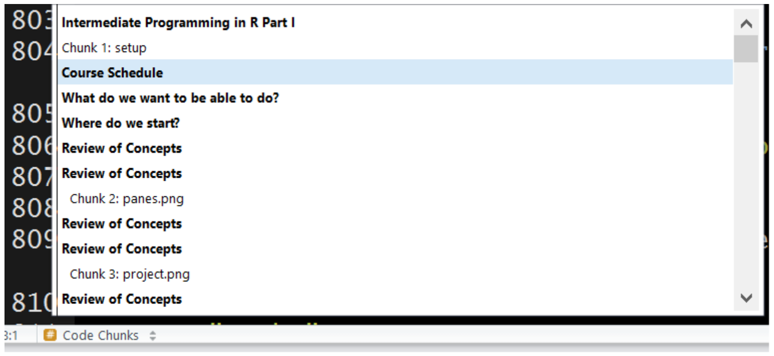
Code Chunks

Can name code chunks to help organization!

- When calling a chunk, add name after `r`

~~~
```{r name-of-chunk, options...}
code
```
~~~

- TOC type menu in bottom left of notebook editor!



63/93

Code Chunks

In a large analysis it may take a long time to run code chunks/knit your document

- Can "Cache" results! Code will only rerun if it has changed.
- Option to set up code dependencies using chunk names
- Use `cache = TRUE` in code chunk definition
- Can do global option for caching!
- Delete folders created to rerun everything

Images/Equations and Misc.

Adding images in markdown: `![](path/to/file)`

- Not ideal… difficult to control size/scale
- Better way to add images use R code!
- `knitr` package has `include_graphics` function
- Use knitr or code chunk options to control size/scale!
- Ex:
  ```{r graphics, out.width = "800px", echo = FALSE}
  knitr::include_graphics(path/to/file)
  ```

Images/Equations and Misc.

Adding Equations

- Inline equation: `$A = \pi*r^{2}$` becomes $A = \pi * r^2$
- Block equation `$$A = \pi*r^{2}$$` becomes

$$A = \pi * r^2$$

- Outputting equations for HTML is done through MathJax (javascript)
- For PDFs it is done through LaTeX (may need to install)

Images/Equations and Misc.

Outputting data tables better with `kable` from `knitr` package

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00  
```

```
kable(summary(cars))
```

speed	dist
Min. : 4.0	Min. : 2.00
1st Qu.:12.0	1st Qu.: 26.00
Median :15.0	Median : 36.00
Mean :15.4	Mean : 42.98
3rd Qu.:19.0	3rd Qu.: 56.00
Max. :25.0	Max. :120.00

</div >

Activity

- **Using Notebook Activity** instructions available on web
- Work in small groups
- Ask questions! TAs and I will float about the room
- Feel free to ask questions about anything you didn't understand as well!

What do we want to be able to do?

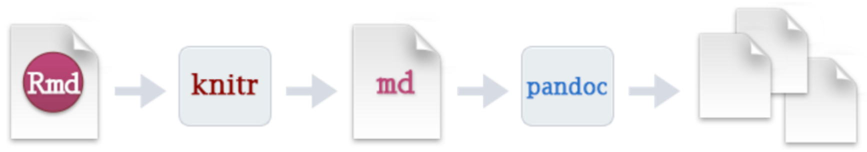
- Communicate findings effectively
- Encompass and document entire data analysis process
- Document findings
- Make process reproducible
- Share process

Where are we at?

- Review of Key Concepts
- R Markdown Basics
  - Code Chunks
  - Images/Equations/Misc.
- R Markdown Options
  - Documents: PDF, HTML
  - Presentations: Slides
  - Interactive Components
- R Shiny Applications/Presentations

Common Outputs

R Markdown really flexible!



Common Outputs

Change output type in the YAML header:

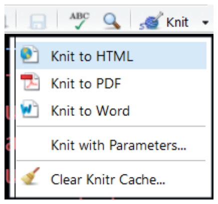
- HTML (a web page)

```
output: html_document
```

Use code explicity:

```
rmarkdown::render("file.Rmd", output_format = "word_document")
```

Use Knit menu:



72/93

Common Outputs

For HTML can include Table of Contents with options

```
output:
  html_document:
    toc: true
    toc_float: true
```

73/93

Common Outputs

For HTML can include Table of Contents with options

```
output:
  html_document:
    toc: true
    toc_float: true
```

For html_documents another option is to make the code chunks hidden by default, but visible with a click:

```
output:
  html_document:
    code_folding: hide
```

Common Outputs

- PDF
- May want to install LaTeX for equations

```
output: pdf_document
```

Common Outputs

- PDF
- May want to install LaTeX for equations

```
output: pdf_document
```

- Word

```
output: word_document
```

Common Outputs

Presentations/Slides

- `output: ioslides_presentation` - HTML presentation
- `slidy_presentation` - HTML presentation
- `beamer_presentation` - PDF presentation with LaTeX Beamer

Common Outputs

Presentations/Slides

- `output: ioslides_presentation` - HTML presentation
- `slidy_presentation` - HTML presentation
- `beamer_presentation` - PDF presentation with LaTeX Beamer
- Shiny (covered later) slides

```
output: html_document
runtime: shinyShiny Slides
```

Common Outputs

Can create more than one document at a time!

- Just add another output statement in the YAML header

```
output:
  html_document:
    toc: true
    toc_float: true
  word_document: default
```

Then use code:

```
rmarkdown::render("file.Rmd", output_format = "all")
```

Parameters

Parameters can be added to the YAML header

- Can help to automate reports!

```
title: "NFL Reports"
author: "Justin Post"
date: "August 10-11, 2017"
output: html_document
params:
      team: "Pittsburgh Steelers"
```

- Access via `params$team`
- Can 'Knit with parameters'
- Example: Let's open up the NFL.Rmd document

Automation of Documents

- Create data frame for each class (here team)

```
scoreData <- read_csv("https://github.com/jbpost2/
                            IntermediateR/blob/master/datasets/scoresFull.csv?raw=true")


reports <- tibble(
  teamIDs = unique(scoreData$awayTeam),
  filename = stringr::str_c("TeamID-", teamIDs, ".html"),
  params = purrr::map(teamIDs, ~ list(team = .))
)
```

Automation of Documents

```
reports
```

```
## # A tibble: 32 x 3
##               teamIDs                           filename     params
##                 <chr>                              <chr>     <list>
## 1 San Francisco 49ers TeamID-San Francisco 49ers.html <list [1]>
## 2   Minnesota Vikings   TeamID-Minnesota Vikings.html <list [1]>
## 3  New Orleans Saints  TeamID-New Orleans Saints.html <list [1]>
## 4       New York Jets       TeamID-New York Jets.html <list [1]>
## 5   Arizona Cardinals   TeamID-Arizona Cardinals.html <list [1]>
## # ... with 27 more rows
```

Automation of Documents

Now knit via the following code:

```
reports %>%
  select(output_file = filename, params) %>%
  purrr::pwalk(rmarkdown::render, input = "NFL.Rmd")
```

Interactivity

HTML documents inherently interactive

- Widgets can be included

```
library(leaflet)
leaflet() %>%
  setView(174.764, -36.877, zoom = 16) %>%
  addTiles() %>%
  addMarkers(174.764, -36.877, popup = "Maungawhau")
```

Interactivity

Interactivity

Interactive tables with `DT` library

```
library(DT)
datatable(iris)
```

Interactivity

Interactivity

- 3d scatterplots with `rthreejs` package

```r
if(!require("devtools")) install.packages("devtools")
devtools::install_github("bwlewis/rthreejs")

library(threejs)

scatterplot3js(x = iris$Sepal.Width, y = iris$Sepal.Length,
                             z = iris$Petal.Width, color =
                                c(rep("blue", 50), rep("red", 50),
                                    rep("green", 50)),
                             size = 0.5)
```

Interactivity

Interactivity

Previous interactivity happened in the browser

- Great because anyone can access with a browser
- Bad because you can't have as much functionality as you want…
- Shiny allows for interactivity with R!
- Only con: Requires R running somewhere
- Examples: Shiny Showcase, Shiny Gallery

Activity

- **Outputs and Interactivity Activity** instructions available on web
- Work in small groups
- Ask questions! TAs and I will float about the room
- Feel free to ask questions about anything you didn't understand as well!

What do we want to be able to do?

- Communicate findings effectively
- Encompass and document entire data analysis process
- Document findings
- Make process reproducible
- Share process

Where are we at?

- Review of Key Concepts
- R Markdown Basics
  - Code Chunks
  - Images/Equations/Misc.
- R Markdown Options
  - Documents: PDF, HTML
  - Presentations: Slides
  - Interactive Components
- R Shiny Applications/Presentations