

Extending R Markdown

Hao Zhu

2019-01-14

Recap

- Knit button workflow
 - ▮ R Markdown (.Rmd) -> knit() -> Markdown(.md) -> pandoc -> html, pdf, etc...
- `knitr::knit_print()`: *Define Custom Print method in rmarkdown for R class*
- `knitr::asis_output()`: *Render string (usually LaTeX/html) "asis" in rmarkdown*
- YAML options & Pandoc template:
 - `html_document()` example
 - YAML options -> Pandoc options -> Pandoc template
- HTML & LaTeX dependencies

In this session...

Three Categories of R Markdown extensions

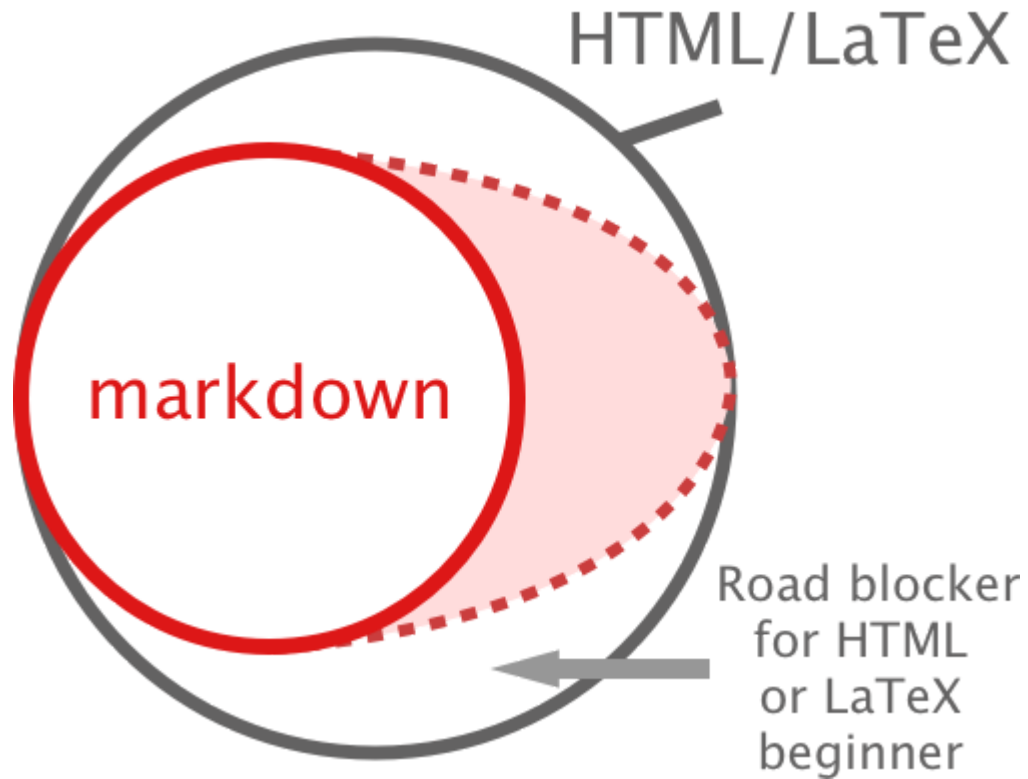
- **Document elements that drop beyond the scope of markdown**
 - `kableExtra` & `gt`
 - `rmdWidgets`
- **R Markdown templates**
 - `rticles`
 - `memor`
- **RStudio Addin & External Services to improve writing experience (we won't go through the details)**
 - Reference management: `rcrossref` + `citr`
 - GIFs: `giphyr`

Part #1. Beyond Markdown

Markdown can't do everything

Markdown is not a replacement for HTML, or even close to it. Its syntax is very small, corresponding only to a very small subset of HTML tags. The idea is not to create a syntax that makes it easier to insert HTML tags. In my opinion, HTML tags are already easy to insert. The idea for Markdown is to make it easy to read, write, and edit prose. HTML is a publishing format; Markdown is a writing format. Thus, **Markdown's formatting syntax only addresses issues that can be conveyed in plain text.** (Daring Fireball)

- Markdown was not designed to solve everything.
- Example:
 - Colored text is not something you can do with pure markdown.
 - Complex tables: Second-level header, highlighted row etc.



Gap includes: Complex tables, formatted texts, textbox, checkbox, etc

kableExtra for Complex Tables

- `knitr::kable()` can generate tables in markdown(default), LaTeX, HTML, pandoc & rst.
- There is no way to format a markdown/pandoc table to publication quality.
- `kableExtra` can modify the outputs of `kable` only when the format is LaTeX or HTML.
- Design Logic: Use `kable` to generate a table first and then use a few functions to format different parts of the table.
- Detailed documentation:
 - [Documentation for HTML](#)
 - [Documentation for LaTeX](#)

STOP!

PIPE CHECK!



Sample kableExtra code

```
library(kableExtra)

# HTML Table
mtcars[1:4, 1:4] %>%
  kable() %>%
  kable_styling() %>%
  add_header_above(c(" ", "RStudio" = 2, "Conf" = 2)) %>%
  group_rows("ARM workshop", start_row = 3, end_row = 4)
```

	RStudio		Conf	
	mpg	cyl	disp	hp
Mazda RX4	21.0	6	160	110
Mazda RX4 Wag	21.0	6	160	110
ARM workshop				
Datsun 710	22.8	4	108	93
Hornet 4 Drive	21.4	6	258	110

Sample kableExtra code

```
# LaTeX Table
mtcars[1:4, 1:4] %>%
  kable("latex", booktabs = T) %>%
  add_header_above(c(" ", "RStudio" = 2, "Conf" = 2)) %>%
  group_rows("ARM workshop", start_row = 3, end_row = 4) %>%
  kable_styling(latex_options = "striped") %>%
  save_kable("img/kE_latex_example.png") # for this presentation only

knitr::include_graphics("img/kE_latex_example.png")
```

	RStudio		Conf	
	mpg	cyl	disp	hp
Mazda RX4	21.0	6	160	110
Mazda RX4 Wag	21.0	6	160	110
ARM workshop				
Datsun 710	22.8	4	108	93
Hornet 4 Drive	21.4	6	258	110

Functions in kableExtra

- `kable_styling`: Setups for general look of the entire table
- `add_header_above`, `group_rows`, `collapse_rows`: Create a layout that shows selected rows/columns belong to one group
- `column_spec`, `row_spec`: Specify styles for selected rows/columns
- `cell_spec/text_spec`: Generate raw HTML/LaTeX code for table or document texts.
- `footnote`: Add table footnotes
- `save_kable & as_image`: Save HTML/LaTeX to HTML, PDF, PNG or JPG or use them as images in rmarkdown.

How tables look like in HTML

```
x <- kable(mtcars[1:2, 1:2], "html")
```

```
attributes(x)
```

```
## $format  
## [1] "html"  
##  
## $class  
## [1] "knitr_kable"
```

```
cat(x)
```

```
## <table>  
##   <thead>  
##     <tr>  
##       <th style="text-align:left;">   </th>  
##       <th style="text-align:right;"> mpg </th>  
##       <th style="text-align:right;"> cyl </th>  
##     </tr>  
##   </thead>  
##   <tbody>
```

kableExtra uses xml2 for HTML table

- kableExtra uses `xml2` to read HTML table as XML and modify nodes as needed

```
kableExtra:::read_kable_as_xml(x)
```

```
## {xml_node}  
## <table>  
## [1] <thead>\n <tr>\n   <th style="text-align:left;">   </th>\n   <th st .  
## [2] <tbody>\n <tr>\n   <td style="text-align:left;"> Mazda RX4 </td>\n .
```

```
x_xml <- x %>%  
  kable_styling(bootstrap_options = "striped") %>%  
  kableExtra:::read_kable_as_xml()  
x_xml
```

```
## {xml_node}  
## <table class="table table-striped" style="margin-left: auto; margin-right:  
## [1] <thead>\n <tr>\n   <th style="text-align:left;">   </th>\n   <th st .  
## [2] <tbody>\n <tr>\n   <td style="text-align:left;"> Mazda RX4 </td>\n .
```

kableExtra uses xml2 for HTML table

```
library(xml2)

x_xml %>%           # table level
  xml_child(2) %>%  # Select tbody
  xml_child(1) %>%  # Select first row in body
  xml_child(1) %>%  # Select first column at first row
  xml_set_attr("style", "color: red;") # Adjust color through CSS
```

```
kableExtra::as_kable_xml(x_xml)
```

	mpg	cyl
Mazda RX4	21	6
Mazda RX4 Wag	21	6

Create new html node using htmltools

```
library(htmltools)

simple_html_tag <- htmltools::tags$tr(list(
  htmltools::tags$td("a"),
  htmltools::tags$td("b")
))

print(simple_html_tag)
```

```
## <tr>
##   <td>a</td>
##   <td>b</td>
## </tr>
```

```
read_xml(as.character(simple_html_tag)) # Convert to xml node
```

```
## {xml_document}
## <tr>
## [1] <td>a</td>
## [2] <td>b</td>
```

Exercise #1: Add a row to HTML table

Add a row to the end of the table. This row has 3 cells "RStudio", "Conf", "2019".

```
library(kableExtra)
library(xml2)
quiz1 <- kable(mtcars[1:2, 1:2], "html") %>%
  kable_styling(full_width = F)
quiz1_xml <- kableExtra::read_kable_as_xml(quiz1)
new_row <- c("RStudio", "Conf", "2019")
# You Start Here ---
```

1. HTML Table 101

- HTML table is created **by row**.
- <table> -> <thead> -> <tr> (Row) -> <th> (Table Header Cell)
- <table> -> <tbody> -> <tr> (Row) -> <td> (Table Body Cell)

2. Use xml2::xml_add_child to add a child xml to <tbody>

Exercise #1: Add a row to HTML table

```
library(kableExtra); library(xml2)
quiz1 <- kable(mtcars[1:2, 1:2], "html") %>%
  kable_styling(full_width = F)
quiz1_xml <- kableExtra::read_kable_as_xml(quiz1)
new_row <- c("RStudio", "Conf", "2019")
# You Start Here ---
new_row_html <- lapply(new_row, htmltools::tags$td) %>%
  htmltools::tags$tr() %>%
  as.character() %>%
  read_xml()
quiz1_xml %>%
  xml_child(2) %>%
  xml_add_child(new_row_html)
kableExtra::as_kable_xml(quiz1_xml)
```

	mpg	cyl
Mazda RX4	21	6
Mazda RX4 Wag	21	6
RStudio	Conf	2019

Custom Print methods for kable

- `knitr::knit_print` allows us to define custom print methods in rmarkdown
 - Original `knitr::kable()`'s `knit_print`

```
knit_print.knitr_kable = function(x, ...) {  
  x = paste(c(  
    if (!(attr(x, 'format') %in% c('html', 'latex')) c('', ''), x, '  
  ), collapse = '\\n')  
  asis_output(x)  
}
```

`asis_output()` puts down the string "asis" in rmarkdown

What about live preview for HTML tables in R console?

What about live preview for HTML tables in R console?

Yes! A regular print method!

kableExtra example

```
print.kableExtra <- function(x, ...) {  
  html_header <- htmltools::tags$head(  
    rmarkdown::html_dependency_jquery(),  
    rmarkdown::html_dependency_bootstrap(theme = "simplex"),  
    html_dependency_kePrint()  
  )  
  html_table <- htmltools::HTML(as.character(x))  
  html_result <- htmltools::tagList(html_header, html_table)  
  if (interactive() & rstudioapi::isAvailable()) {  
    htmltools::html_print(html_result, viewer = rstudioapi::viewer)  
  }  
}
```

Questions?

LaTeX tables

```
sample_latex <- kable(mtcars[1:2, 1:2], "latex", booktabs = T)
print(sample_latex)
```

```
##
## \begin{tabular}{lrr}
## \toprule
##   & mpg & cyl\\
## \midrule
## Mazda RX4 & 21 & 6\\
## Mazda RX4 Wag & 21 & 6\\
## \bottomrule
## \end{tabular}
```

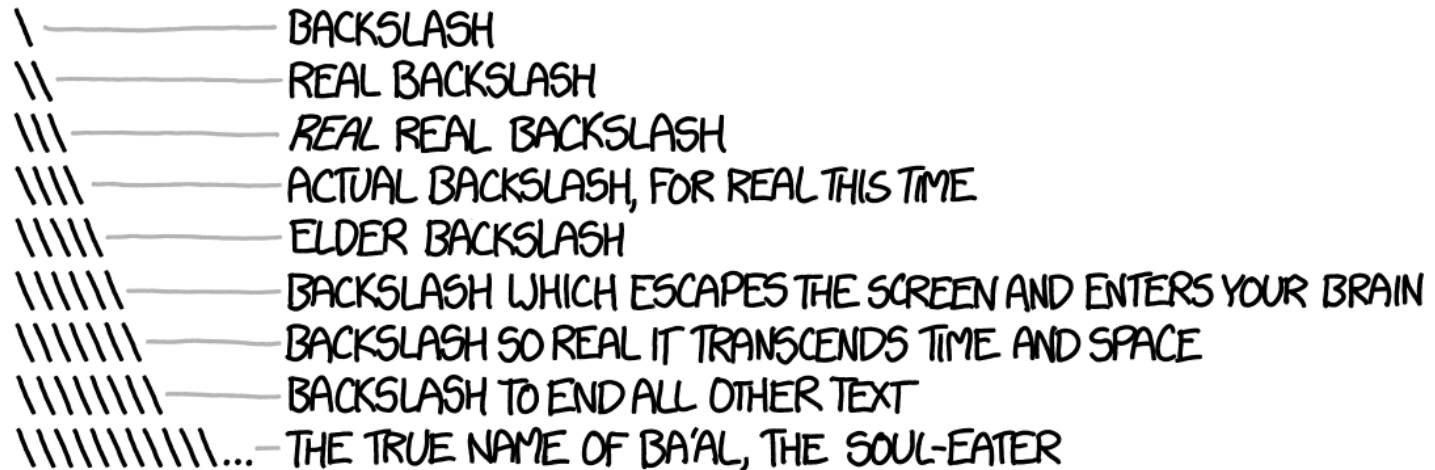
LaTeX tables

- kableExtra uses regex to understand and modify LaTeX tables.

```
kableExtra::magic_mirror(sample_latex)
```

```
## $tabular  
## [1] "tabular"  
##  
## $booktabs  
## [1] TRUE  
##  
## $align  
## [1] "lrr"  
##  
## $valign  
## [1] ""  
##  
## $ncol  
## [1] 3  
##  
## $nrow  
## [1] 3  
##
```

When you use regex with LaTeX...



A hand-drawn list of backslash sequences and their 'meanings' in a humorous, escalating manner. The list is enclosed in a rectangular box and consists of nine entries, each with a backslash sequence followed by a horizontal line and a description. The descriptions become increasingly absurd and dramatic as the number of backslashes increases.

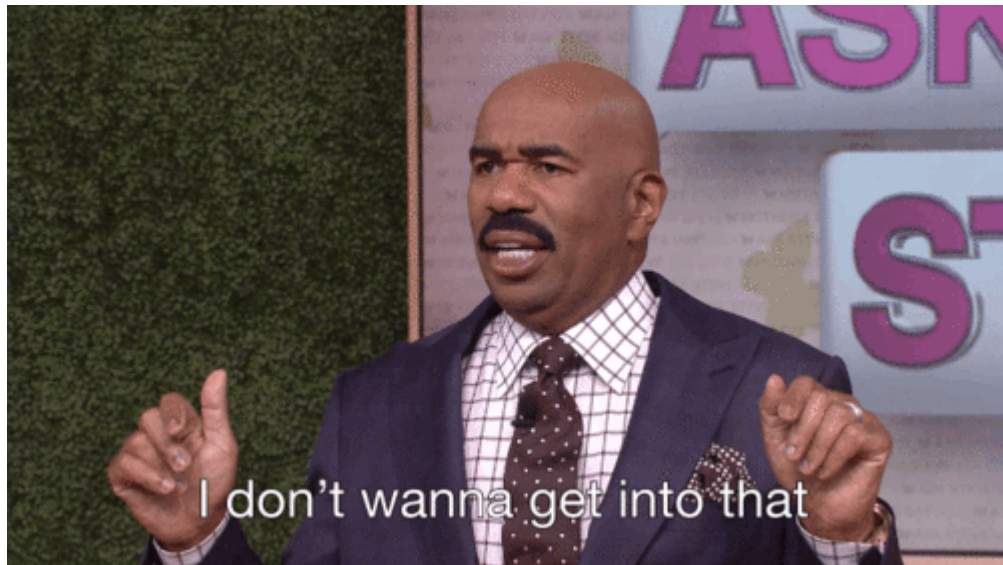
- \ — BACKSLASH
- \\ — REAL BACKSLASH
- \\\ — REAL REAL BACKSLASH
- \\\\ — ACTUAL BACKSLASH, FOR REAL THIS TIME
- \\\\\\ — ELDER BACKSLASH
- \\\\\\\\ — BACKSLASH WHICH ESCAPES THE SCREEN AND ENTERS YOUR BRAIN
- \\\\\\\\\\ — BACKSLASH SO REAL IT TRANSCENDS TIME AND SPACE
- \\\\\\\\\\\\ — BACKSLASH TO END ALL OTHER TEXT
- \\\\\\\\\\\\\\... — THE TRUE NAME OF BA'AL, THE SOUL-EATER

xkcd

Anyway, thanks to everyone who reported bugs on github, this methodology is still working.

Managing LaTeX dependency through code

- Like R, in LaTeX, people use `\usepackage{}` to load packages
- Most advanced features for tables (and other things) in LaTeX relies on these packages. For example: `booktabs`, `longtable`, `xcolor`...
- Most R table packages require users to put `\usepackage{}` in `includes` by themselves. However, for LaTeX beginners...



Managing LaTeX dependency through code

- If you know your users need packages like xcolor or booktabs, load the packages for them!

kableExtra Example

```
usepackage_latex <- function(name, options = NULL) {  
  invisible(knitr::knit_meta_add(  
    list(rmarkdown::latex_dependency(name, options))  
  ))  
}  
  
usepackage_latex("booktabs")
```

By combining `knitr::knit_meta_add` and `rmarkdown::latex_dependency`, you can pass the latex package information to `rmarkdown::meta_data`, which will then be loaded into document right before body.

Cross-format Tables - Approach #1

- Let kable determine output format automatically
 - Inside a rmarkdown -> PDF session, kableExtra will set global option `knitr.table.format` as "latex". Otherwise, let's go with html. :)
 - Useful in bookdown or cases where users need to render to different formats at the same time.
 - This code below will produce a table work in both HTML & LaTeX.

```
mtcars[1:2, 1:2] %>%  
  kable(booktabs = T) %>%  
  kable_styling(  
    bootstrap_options = "striped",  
    latex_options = "striped"  
  ) %>%  
  column_spec(1, bold = T)
```

Cross-format Tables - Approach #2

- Render tables as graphics?
 - `save_kable` can save HTML/LaTeX tables to files as html, pdf, png, jpg...
 - `as_image` calls `save_kable` internally, renders a png and includes the png as graphics in rmarkdown
- For LaTeX tables (Workflow inspired by [texPreview](#))
 1. Render a standalone PDF snippet
 2. Use [magick](#) to readin
 3. Convert to image
- For HTML tables
 1. Render HTML to a file
 2. Use [webshot](#) to take a screenshot of the page
 3. Use [magick](#) to adjust if available.

gt from RStudio

- [Project Page](#)
- A cleaner approach - Instead of modifying existing table code, it defines an object that contains information and renders it out when necessary
- Great integration with tidyverse
- Comes with its own CSS for HTML

```
library(gt)
```

```
gt_example <- mtcars[1:3, 1:4] %>%  
  tibble::rownames_to_column() %>%  
  gt()  
gt_example
```

	mpg	cyl	disp	hp
Mazda RX4	21.0	6	160	110
Mazda RX4 Wag	21.0	6	160	110
Datsun 710	22.8	4	108	93

More gt

```
mtcars[1:3, 1:4] %>%  
  tibble::rownames_to_column() %>%  
  dplyr::group_by(cyl) %>%  
  gt() %>%  
  fmt_currency(vars(mpg))
```

	mpg	disp	hp
6			
Mazda RX4	\$21.00	160	110
Mazda RX4 Wag	\$21.00	160	110
4			
Datsun 710	\$22.80	108	93

Combine gt with kableExtra

Certainly possible (in the future), at least for HTML.

```
gt2k <- function(x) {  
  out <- gt::as_raw_html(x)  
  attr(out, "format") <- "html"  
  return(out)  
}  
gt_example %>%  
  gt2k() %>%  
  column_spec(3, color = "red", bold = T, italic = T) %>%  
  group_rows("Test", 2, 3,  
    label_row_css = "background-color: gray;color:white;padc
```

	mpg	cyl	disp	hp
Mazda RX4	21.0	6	160	110
Test				
Mazda RX4 Wag	21.0	6	160	110
Datsun 710	22.8	4	108	93

Other than table..

- checkbox from `rmdWidgets`

```
rmd_checkbox(c("Yes", "No"), selected = 1, label = "Hello World?")
```

Hello World?

☒ Yes ☐ No

```
rmd_checkbox(c("Yes", "No"), selected = 1, label = "Hello World?",  
             label_inline = F, inline = F)
```

Hello World?

☒ Yes

☐ No

(This package is still in proof-of-concept stage.)

Questions?

Part #2. R Markdown Templates

R Markdown Template Basics

- R Markdown Template is more than stylesheet!!
 - It's a bundle of stylesheet, document content skeleton, citation info & other misc files (such as logo images)
- For different formats
 - LaTeX: LaTeX template file
 - HTML: css and javascript
 - Word: Word reference files
- There are about 60 ~ 70 templates on CRAN but a lot others on github only
 - How do I know?

Template packages you should know

- LaTeX template
 - **rticles**
 - **memor**
 - **papaja**
 - **vitae**
- HTML template
 - **radix**
 - **prettydoc**
 - **xaringan**
 - **flexdashboard**
 - **rmdformats**
- Mixed Formats
 - **tufte**
 - **rmdTemplates**

Template Package Structure

```
- DESCRIPTION
- R/
  - my_template.R          # An R binding (function) that can be called
- inst/
  - rmarkdown/
    - templates/
      - my_template/
        - resources/      # Stylesheet and that resources needed in the template
        - skeleton/       # Rmd skeleton. All file here will be copied
        - template.yaml   # Declare the existence of this template
```

- All files in skeleton will be copied to users folder.
- All files in resources are supposed to be called by either R or the template itself at least once
- How many rmarkdown templates are there on CRAN?

rticles: LaTeX Journal Article Template

- [ACM](#) articles
- [ACS](#) articles
- [AEA](#) journal submissions
- [AMS](#) articles
- [Biometrics](#) articles
- [Bulletin de l'AMQ](#) journal submissions
- [CTeX](#) documents
- [Elsevier](#) journal submissions
- [IEEE Transaction](#) journal submissions
- [JSS](#) articles
- [MDPI](#) journal submissions
- [Monthly Notices of the Royal Astronomical Society](#) articles
- [NNRAS](#) journal submissions
- [PeerJ](#) articles
- [Royal Society Open Science](#) journal submissions
- [Sage](#) journal submissions
- [Springer](#) journal submissions
- [Statistics in Medicine](#)1097-0258/homepage/la_tex_class_file.htm) journal submissions
- [Copernicus Publications](#) journal submissions
- [The R Journal](#) articles

Example: Create a Springer journal submission

Let's take a closer look

- `inst/rmarkdown/templates/springer_article/template.yaml`

```
1  name: Springer Journal Article
2  description: >
3    Template for creating an article for submission to any Springer journal
4  create_dir: true
```

`create_dir` Rule of Thumb:

If you have more than 1 additional files other than `skeleton.Rmd` in the `skeleton` folder, you should set `create_dir` to be `true`

- `inst/rmarkdown/templates/springer_article`
- `R/springer_article.R`


```
rmarkdown::pdf_document()
```

```
## $knitr
## $knitr$opts_knit
## NULL
##
## $knitr$opts_chunk
## $knitr$opts_chunk$dev
## [1] "pdf"
##
## $knitr$opts_chunk$fig.width
## [1] 6.5
##
## $knitr$opts_chunk$fig.height
## [1] 4.5
##
## $knitr$opts_chunk$dev.args
## $knitr$opts_chunk$dev.args$pdf
## $knitr$opts_chunk$dev.args$pdf$useDingbats
## [1] FALSE
##
##
##
## $knitr$opts_chunk$crop
## [1] TRUE
##
##
```

The Golem of R Markdown Template

Questions?

Let's look even closer at the template file

- Springer template.tex

Programming with Pandoc variable

- [Pandoc's Official Documentation](#)
- Basic Syntax
 - Quote variables in a pair of `$`
 - `$if()$...$endif$`
 - `$for()$...$endfor$`
- How does Pandoc know these variables we set in Rmarkdown?
 - Recap
 - R Markdown (.Rmd) -> knit() -> Markdown(.md) -> pandoc -> html, pdf, etc...
 - YAML options -> Pandoc options -> Pandoc template
 - How exactly? **(Tricky)**
 - Pandoc processes .md which has those yaml variable and use them with template.
 - rmarkdown processes information and pass them as arguments and variables to pandoc through a system call

[Source code for pdf_document](#)

You can see this system call in your console

```
.../Springer/Springer.Rmd

processing file: Springer.Rmd
|.....| 100%
ordinary text without R code

/Applications/RStudio.app/Contents/MacOS/pandoc/pandoc +RTS -K512m -RTS Springer.utf8.md --to latex --from markdown+autolink_bare_uris+ascii_identifiers+tex_math_single_backslash --output Springer.tex --template /Library/Frameworks/R.framework/Versions/3.5/Resources/library/rmarkdown/templates/springer_article/resources/template.tex --highlight-style tango --pdf-engine pdflatex --filter /Applications/RStudio.app/Contents/MacOS/pandoc/pandoc-citeproc
output file: Springer.knit.md

Output created: Springer.pdf
```

Exercise #2: Create a pdf_document-like template can allow users to put in some description after title.

You can start from 02_desc_document in your folder.

Tips:

- For simplicity, please use the Pandoc approach. **In this case, the only file you only need to modify is template.tex.**
- Right after title is the place where you put the description.
- You need to build your package to let it work. Raise your hand if you have questions.

![img/pandoc_hint.png]

How about the second approach? (more advanced)

- Here you need to do two things:
 - Create a new document function that inherits `pdf_document` and send session info as part of Pandoc Arguments
 - Mark where you want to put this pandoc argument in `template.tex`
- You can find all the current pandoc variables in `pdf_document()$pandoc$args`
- In addition, you can add items to this vector.
- You can use `rmarkdown::pandoc_variable_arg()` to generate a valid pandoc argument

For example, if I want to pass event from R to pandoc as a variable also called "event"

```
out <- rmarkdown::pdf_document()

event <- "RStudio Conf 2019"
event_pandoc <- rmarkdown::pandoc_variable_arg(
  "event", event
)

out$pandoc$args <- c(out$pandoc$args, event_pandoc)
```


memor: Customizable memo in R Markdown

- [Project Page](#) & [demo document](#)



memor: Customizable LaTeX Template for rmarkdown

Hao Zhu, Timothy Tsai, Thomas Trivison

2018-07-09

1 INTRODUCTION

We love rmarkdown. In practice, however, we often have specific customization requirements for reporting of reproducible research. Some of these are universal, such as company logo or letterhead, contact info and so on.

We created this `memor` package to allow for easier customization of LaTeX-based documents combining

Many elements are customizable with memor

title: "memor: Customizable LaTeX Template for rmarkdown"

author: Hao Zhu, Timothy Tsai, Thomas Trivison

date: "2018-07-09"

output:

memor::pdf_memo:

logo: "memor.png"

logo_height: 2.5cm

use_profile: false

company:

name: Institute for Aging Research

address: 1200 Centre St, Boston, MA

phone: 617-971-5386

email: stats@hsl.harvard.edu

confidential: false

watermark: Open Access

libertine: true

memor profile

- You don't want to have a long yaml section in every document, right?
- You can save information of your school/workplace separately.
- And controlled with an RStudio Addin!

Word Templates

- R Markdown Word Templates can either bundle with a package or be standalone as a template reference file
- Good Resources to learn:
 - [R Markdown: The Definitive Guide Chap 17](#)
 - [Happy collaboration with Rmd to docx](#)
 - [A Coursera course video from Emory by Melinda Higgins](#)
 - [Create A MS Word Template for R Markdown](#)
 - [rmdTemplates](#)

Bookdown Template?

- [The First Bookdown Contest](#)

Questions?

Part #3. RStudio Addins for better writing experience

Reference Management in R Markdown

- [rcrossref](#) allows you to search articles and download their bibtex
- [citr](#) allows you to add citation from the bibtex

Insert GIF to R Markdown

- [giphyr](#)

Thanks!