# Advanced R Markdown

## Day 2: Customization and Extensions

Yihui Xie and Hao Zhu

2019/01/16 @ rstudio::conf, Austin, TX


6
7

Slides: http://bit.ly/arm-xie

Examples: http://bit.ly/arm-exm

# Outline

- Welcome to the command-line world
  - Parameterized reports
- How R Markdown works: knitr + Pandoc
  - Pandoc's Markdown
  - knitr: Things you may not know
- R Markdown output formats
- Custom templates and formats
  - : LaTeX journal articles
  - : LaTeX customization
- Shiny and HTML widgets
- (Optional) knitr hooks and language engines

# Using R Markdown via command line

# rmarkdown::render()

- Under the hood, it calls ( ) and Pandoc ( to other formats)
- processes code chunks and inline R expressions
- Pandoc converts Markdown to other output formats
- Click the Knit button (in RStudio), and get one output document
- If you run a loop, you can easily get a thousand reports

# Understanding the argument

- has an argument for the environment in which the R code in the R Markdown document is evaluated
- The default is , which is usually the global environment of your workspace, unless you are calling this function inside other functions

# A quick example

A custom render function:

The source of :

Call the custom render function:

# Parameterized reports via

- The argument is extremely flexible, but it may be too technical (it is not trivially easy to understand R's environments)
- R Markdown introduced a special object to help you parameterize your reports
- You can use either the argument of , or define in YAML, e.g.,

  or command line:

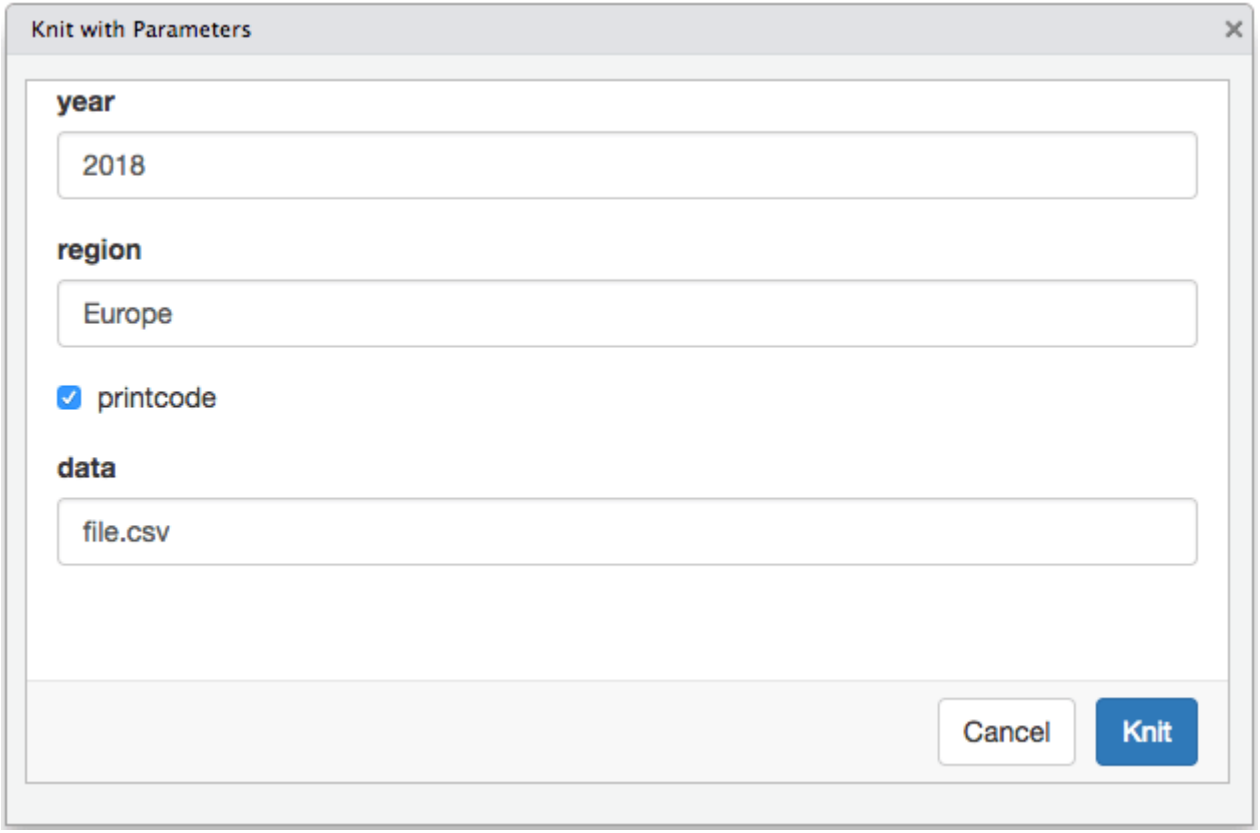
  Command-line will override in YAML; may contain multiple parameters.

# Using inside R Markdown

Typically is a list, so you can extract its elements via (or ).

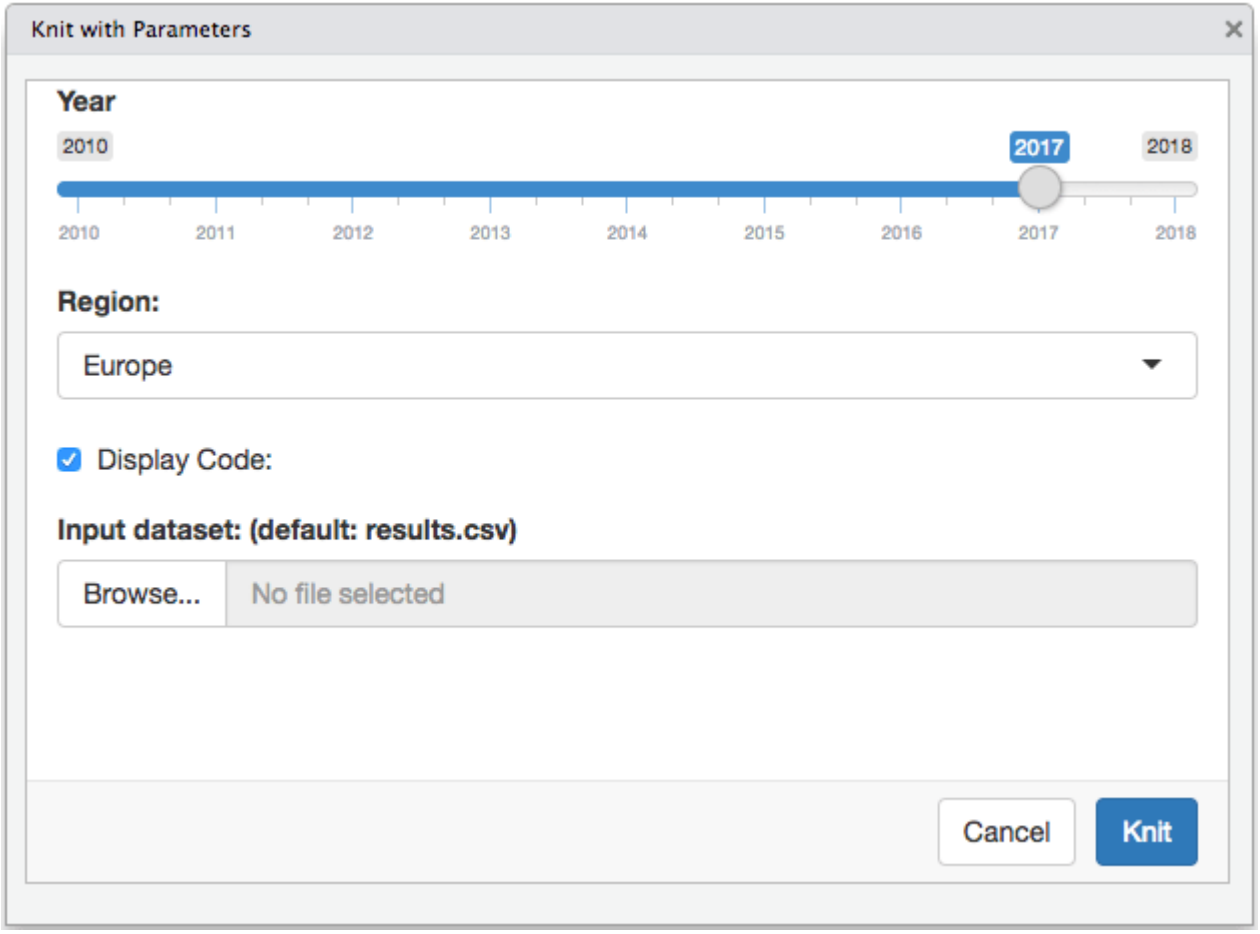
Render reports by a changing parameter through a loop:

# Input parameters interactively




in RStudio. Section 15.3.3 of the R Markdown book.

# More input controls




Example: 01-params-controls.Rmd

# Parameterized reports on RStudio Connect

- https://www.rstudio.com/products/connect/
- Input parameters through the web interface of RStudio Connect
- View reports built previously
- Automated emails
- Example

# Render & download a report in a Shiny app

- Example: http://shiny.rstudio.com/gallery/download-knitr-reports.html
- Source: https://github.com/rstudio/shiny-examples/tree/master/016-knitr-pdf

# Debugging R Markdown documents

- For non-trivial debugging tasks (e.g., debugging complicated functions), you have to call interactively.
  - Inside the R Markdown document, you may use usual debugging techniques such as or inserting in functions.
- To debug the Pandoc conversion, try . Then intermediate files (such as ) will be preserved, so you can check what's possibly wrong there.

# How R Markdown works

> Good morning, #rstats friends! I mentioned in class how learning R is a lifelong process, there isn't always a "right" answer, & our community is kind & supportive of beginners. In the spirit of being vulnerable, what's one thing in R you don't yet quite understand?
>
> --- Jesse Mostipak (@kierisi)

> Good morning, #rstats friends! I mentioned in class how learning R is a lifelong process, there isn't always a "right" answer, & our community is kind & supportive of beginners. In the spirit of being vulnerable, what's one thing in R you don't yet quite understand?
>
> --- Jesse Mostipak (@kierisi)
>
> Anything about the inner workings of rmarkdown/knitr/pandoc. I press knit, a document appears, and I believe that anything happening in between could be actual magic.
>
> --- Allison Horst (@allison_horst)


https://twitter.com/AlexisLNorris/status/1082039311820836864

# The Knit button

- It calls
- R Markdown ≈ knitr (R) + Pandoc (Markdown)
- ≈ + a call to
- R Markdown ( ) -> -> Markdown ( ) -> ->
  - 
  - (LaTeX)
  - 
  - 
  - 
  - …

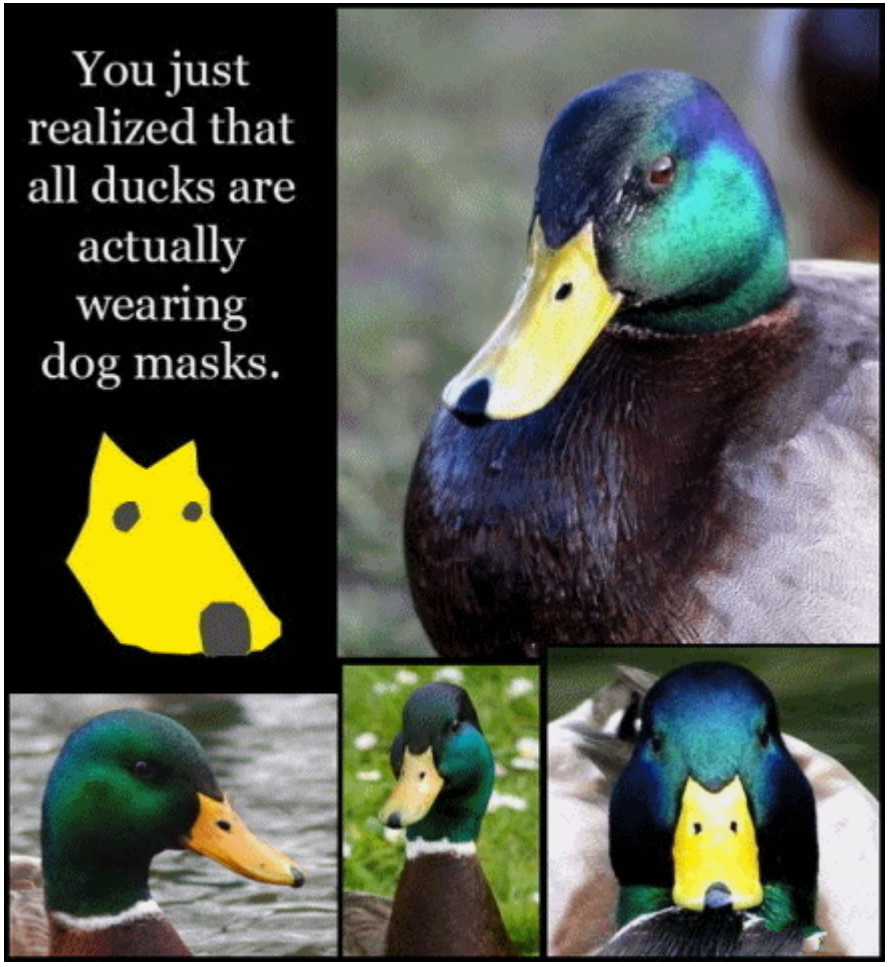
# A minimal R Markdown document

# Markdown output after knitting

# After Pandoc conversion (HTML output)

# After Pandoc conversion (LaTeX output)

# Same ducks, different masks



## The (R) Markdown philosophy

Similar to KISS

# Keep the Duck Simple and Stupid

and wear a mask as fancy as you want

# Pandoc's Markdown

- You should read the Pandoc Manual at least once to learn the possibilities of Pandoc's Markdown: https://pandoc.org/MANUAL.html#pandocs-markdown
- Original Markdown (John Gruber)
  - primarily for HTML
  - paragraphs, ,
  - ,
  -
  -
  -
  - code blocks (indent by four spaces)

# Pandoc's Markdown

- Markdown extensions
  - YAML metadata
  - LaTeX math =
  - syntax highlighting of code blocks (three backticks followed by the language name, e.g. )
  - tables
  - footnotes
  - citations (database can be BibTeX or in YAML)
  - raw HTML/LaTeX

# Pandoc's Markdown

- Types of output documents
  - LaTeX/PDF, HTML, Word (MS Word, OpenOffice)
  - beamer, ioslides, Slidy, reveal.js
  - E-books
  - ...


https://pandoc.org

# Command-line usage of Pandoc

Some examples:

To run system commands in R, use functions or .

The package provides a helper function to convert Markdown documents to other formats using Pandoc.

When you click the Knit button in RStudio, you will see the actual (usually very long) command that is executed.

# Example: Markdown in the eyes of Pandoc

## The Pandoc abstract syntax tree (AST)

Let's explore a Markdown file with R:

How to change to ? Or in general, level-N headers to level-(N-1) headers?

# More power (and speed) with Lua filters

Rewrite the previous R function with a Lua filter :

Run it:

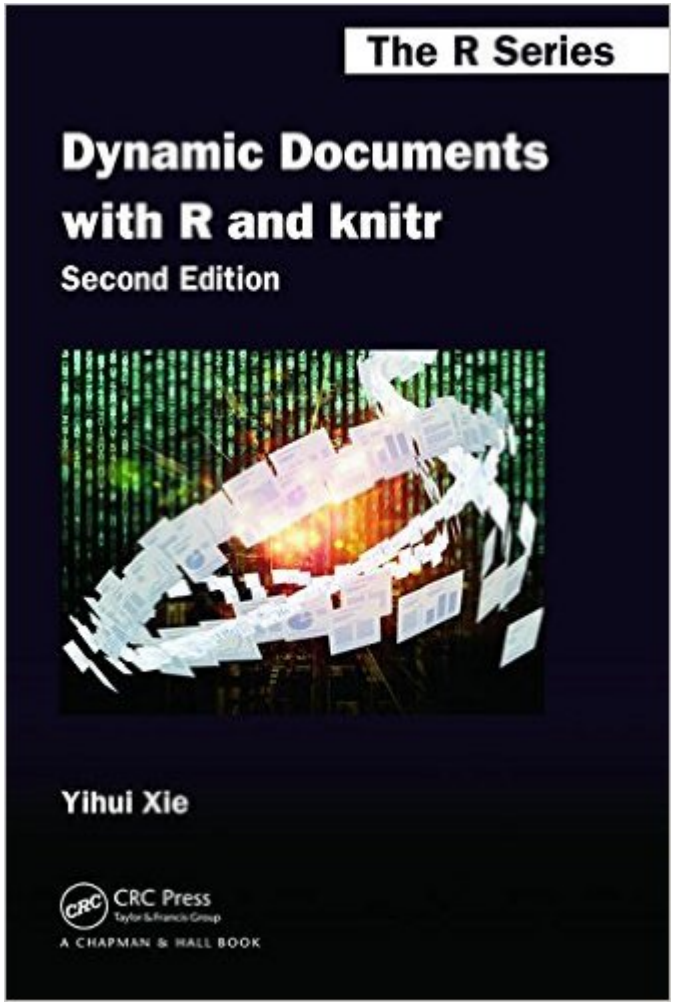
More about Lua filters: https://pandoc.org/lua-filters.html

# The Pandoc version

- RStudio has bundled a version of Pandoc, so you don't need to install Pandoc separately if you use RStudio
- If you install Pandoc by yourself, will use the highest version of Pandoc that it can find
- Check
- RStudio 1.1.x included Pandoc 1.19.x; RStudio 1.2.x will include Pandoc 2.x
  - Pandoc 2.x is not fully compatible with 1.x, but we have solved these issues in the package and other R packages we maintain (e.g., was renamed to )

# knitr

the other cornerstone of R Markdown




The book is a comprehensive guide, but is unfortunately not free. Stay tuned for a free book this year.

# knitr is not only for R

- It contains many, many other language engines: https://bookdown.org/yihui/rmarkdown/language-engines.html
- For example, Shell/Bash scripts, SQL, Python, C, C++, Fortran, Stan, ...
- Demo of two engines: and .

# knitr is not only for Markdown, either

R Markdown may be the most popular document format, but you could also use other authoring languages such as LaTeX, HTML, AsciiDoc, and reStructuredText.

Demo: ,

# knitr works on R scripts, too

- Most of time you may be using , but sometimes you may want .
- first converts an R script to R Markdown (or other document formats that supports, such as ).
- If you use RStudio, you can click the button "Compile Report" on the toolbar.
- Demo: https://github.com/yihui/knitr/blob/master/inst/examples/knitr-spin.R

# The chunk option include=FALSE

Have you ever used these chunk options?

```

```

or

```

```

or even

```

```

You probably only need a single chunk option : https://yihui.name/en/2017/11/knitr-include-false/.

# Conditional evaluation/inclusion

Include a chunk in the output only if the output format is :

Helper functions ( or ) and ( , , , ...). Evaluate a code chunk only if the output format is LaTeX:

BTW, the package makes heavy use of these functions so that its functions work for both HTML and LaTeX output, e.g., .

# Live-preview HTML output documents

- Tired of clicking the Knit button to view your results?
- Just use .
  - 
  - You can also use the RStudio addin "Infinite Moon Reader".
- Demo
- For more info, see
  - https://bookdown.org/yihui/rmarkdown/compile.html
  - https://bookdown.org/yihui/rmarkdown/xaringan-preview.html
  - https://yihui.name/en/2017/08/why-xaringan-remark-js/


DB 3584 AY

# knitr::knit_watch()

Watch an input file continuously, and knit it when it is updated, e.g.,

```
```

This function works for any documents with any output formats, but unlike , it does not automatically refresh the output page. However, if the output format is PDF, your PDF viewer might be able to automatically refresh the page when the PDF has been updated.

# Caching

- The chunk option
- Basic idea: if nothing has changed from the previous run, just load the results instead of executing the code chunk again.

```
```

- Further reading (why caching is one of the two hard things in computer science): https://yihui.name/en/2018/06/cache-invalidation/

# You can generate animations from R plots

- Requires
- Demo

# You can generate animations from R plots

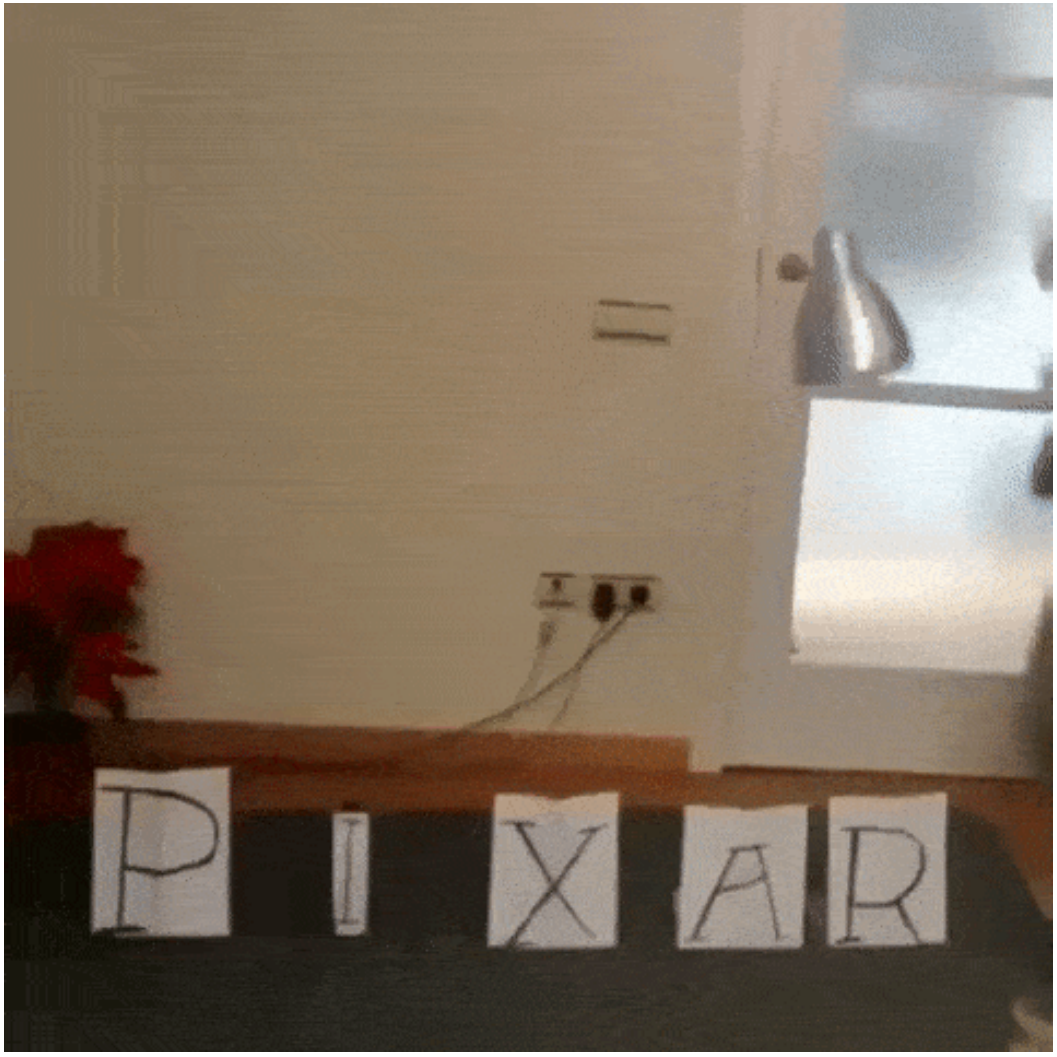
- Requires
- Demo

- You may also use FFmpeg (https://ffmpeg.org) (easy to install for macOS users using Homebrew: )

  The animation format is specified by the chunk option . It could be , , , or any other formats that FFmpeg supports.

- You may use when the animation takes long time to generate.

- The package works out of the box with ( in a code chunk).


PIXAR

# Reuse a code chunk

- If you want to reuse the code from a chunk, don't copy and paste.
- Three ways:
  1. Use the same label, but leave the chunk empty. Useful when you want to run the same code twice with different chunk options.
  2. Use the option, and leave the chunk empty; can be a vector of chunk labels.
  3. Use the syntax to embed one chunk in another.
- Demo
- More info: https://yihui.name/knitr/demo/reference/

# Child documents

Don't want to write everything in a single document? You can use child documents, and include them in the main document via the option, e.g.,

You can also be creative, e.g., conditionally include child documents:

Remember: 's chunk options can be arbitrary valid R code, so feel free to use -statements.

# knitr::knit_expand()

More info:
https://cran.rstudio.com/web/packages/knitr/vignettes/knit_expand.html

# knitr::knit_expand() with file templates

A (child) template document :

Build linear regression models using all variables against in the dataset:

# knitr::fig_chunk()

- When you draw a plot in a code chunk, but want to show it elsewhere (not in the code chunk), gives you the path to the plot file.

```
```

- More info: https://yihui.name/en/2017/09/knitr-fig-chunk/

# knitr::write_bib()

Normally you want to write citation entries to a file (the default is to write to the R console), e.g., .

What I often do:

# knitr::knit_print()

- Visible objects in code chunks are printed through this S3 generic function
- You can register custom printing methods
- See the vignette for details: https://cran.rstudio.com/web/packages/knitr/vignettes/knit_print.html
- The package
- Example

# R Markdown output formats

# R Markdown output formats

- An output format is an abstraction in as a uniform (programming) interface to deal with
  - options (chunk options, hooks, package options, ...)
  - pandoc options ( , , , ...)
  - pre/post-processors
  - and other options (e.g., whether to keep the intermediate .md)
- Can be created via
- Note the argument: output formats are . If you only want to modify a few options of an existing format, you can use it as the base, e.g., you can add a custom post-processor on top of the existing one.

# Built-in formats

- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 

# YAML options for output formats

The YAML metadata

will be translated to

# Example: html_document()

# Example: html_document()

Some options:

- : you can set it to to reduce the HTML file size significantly (because of Bootstrap)
- : tweak the styles of certain elements
- : a custom Pandoc template

# Pandoc templates

- Official Pandoc templates: https://github.com/jgm/pandoc-templates
- 's templates: https://github.com/rstudio/rmarkdown/tree/master/inst/rmd

# A minimal HTML template

# A minimal LaTeX example

# Simple customization

There are many options you can set in YAML. Two types of options:

- Options for Pandoc: make sure you read the Pandoc manual to know the possible options (e.g., for LaTeX output: https://pandoc.org/MANUAL.html#variables-for-latex).

- Options for an R Markdown output format in the field in YAML: consult the specific R help page.

You can certainly create your own template, but it may not be necessary to do so if your problem can be solved by setting a few options in YAML.

# A crash course on HTML/CSS/JavaScript?

or

Learn to use the Developer Tools of your web browser. They are very powerful!

# Custom Word/PPT templates

Idea: generate an arbitrary document with Pandoc first, customize the style of this document, and use it as the "reference document".

PowerPoint output requires Pandoc 2.x, which has been bundled in RStudio 1.2.x (currently a preview version).

# Deeper customization

A common use case: inject a snippet of code to the HTML (e.g., JS/CSS code), or the LaTeX preamble (e.g., load some LaTeX packages before ).

Even deeper customization? Sure, write a package with custom output formats! Let's study a few relatively simple examples in first.


Take a deep breath and read some source code!

# Example: latex_fragment

- https://github.com/rstudio/rmarkdown/blob/b209cdc/R/pdf_document.R#L252-L256
- The key: use a custom template https://github.com/rstudio/rmarkdown/blob/master/inst/rmd/fragment/default.tex
- Similarly: https://github.com/rstudio/rmarkdown/blob/master/R/html_fragment.R and https://github.com/rstudio/rmarkdown/blob/master/inst/rmd/fragment/default.html

# Example: powerpoint_presentation

A minimal example of the PowerPoint output format (not really ):

# Example: rtf_document

- https://github.com/rstudio/rmarkdown/blob/master/R/rtf_document.R
- pre-processor (protect raw RTF content)
- post-processor (restore raw RTF content)
- raw RTF looks like this

# Custom Templates and Formats

# Hao Zhu's session

https://arm.rbind.io/days/day2/

# Shiny documents

# Shiny documents vs Shiny apps

- R Markdown + in YAML
- In a Shiny document, you render output wherever you need it in the document. No need to write a UI. A Shiny app requires both a UI and the server logic ( ).
- In other words, the R Markdown document itself is the UI.

# Render output inline

- I assume most people are familiar with using in code blocks.
- You can also in an inline R expression in R Markdown.
- https://shiny.rstudio.com/gallery/inline-output.html
- Source: https://github.com/rstudio/shiny-examples/blob/master/026-shiny-inline/index.Rmd

# Render output inline

- I assume most people are familiar with using in code blocks.
- You can also in an inline R expression in R Markdown.
- https://shiny.rstudio.com/gallery/inline-output.html
- Source: https://github.com/rstudio/shiny-examples/blob/master/026-shiny-inline/index.Rmd
- Potential application: a recipe website? I really need this for making moon cakes.

# Delayed rendering

- Wrap your in to delay rendering output until the document has been compiled.
- Useful when the Shiny output takes long time to render.
- Demo

# HTML widgets

# HTML widgets

- (Often interactive) JavaScript applications created from R and displayed on HTML pages
- Can be viewed (1) as a standalone page when printed in the R console (2) in R Markdown output documents (HTML) (3) in Shiny apps
- You can pretty much think them like normal R plots
- See Chapter 16 of the R Markdown book

# The three components

- R binding: pass data and options from R to JS
- JS binding: receive data from R and create the widget
- A YAML configuration file to specify HTML/JS/CSS dependencies

# A self-contained minimal example

# Example: the sigma package

- Source: https://github.com/jjallaire/sigma
- sigma.js: http://sigmajs.org
- Basic file structure:

# sigma.yaml

# sigma.R

# sigma.js

# Demo

# Shiny output wrappers

# HTML widgets for non-HTML output

- HTML widgets are for HTML output formats (of course!). What if we embed a widget in a PDF document? In this case, will take a screenshot of the widget automatically if you have installed and PhantomJS:

- Demo

# Misc topics (time permitting)

# knitr hooks

- Chunk hooks: you can run extra code before/after each code chunk
- Output hooks: you have control over every single piece of the output (text, plots, messages)
- https://yihui.name/knitr/hooks/

# Chunk hooks

A chunk hook is a function with three arguments (the latter two are optional). Register the hook function via :

Chunk hooks are triggered when the corresponding chunk option is not , e.g.,

Of course, you can set the option globally so that the hook is executed for all chunks:

Use your imagination.

# Output hooks

Hook names preserved for output:

# Example: truncate long text output

Example 052: https://github.com/yihui/knitr-examples/

# knitr's language engines

See Section 2.7 of the R Markdown book for some examples.

For curious hackers: https://github.com/yihui/knitr/blob/master/R/engine.R

# A minimal Python engine

You can execute Python code via the command line

.

Now you can use the new engine , e.g.,

Use your imagination. Language engines don't have to involve command-line tools. I give you the code and chunk options. You do whatever you like.

HELLO, KNITR ENGINES!

# Thank you!

## All materials can be found at https://arm.rbind.io

You will receive an email request to fill out a workshop feedback survey at the end of the day. We will truly appreciate it if you could fill it out to help us improve our workshops in the future.