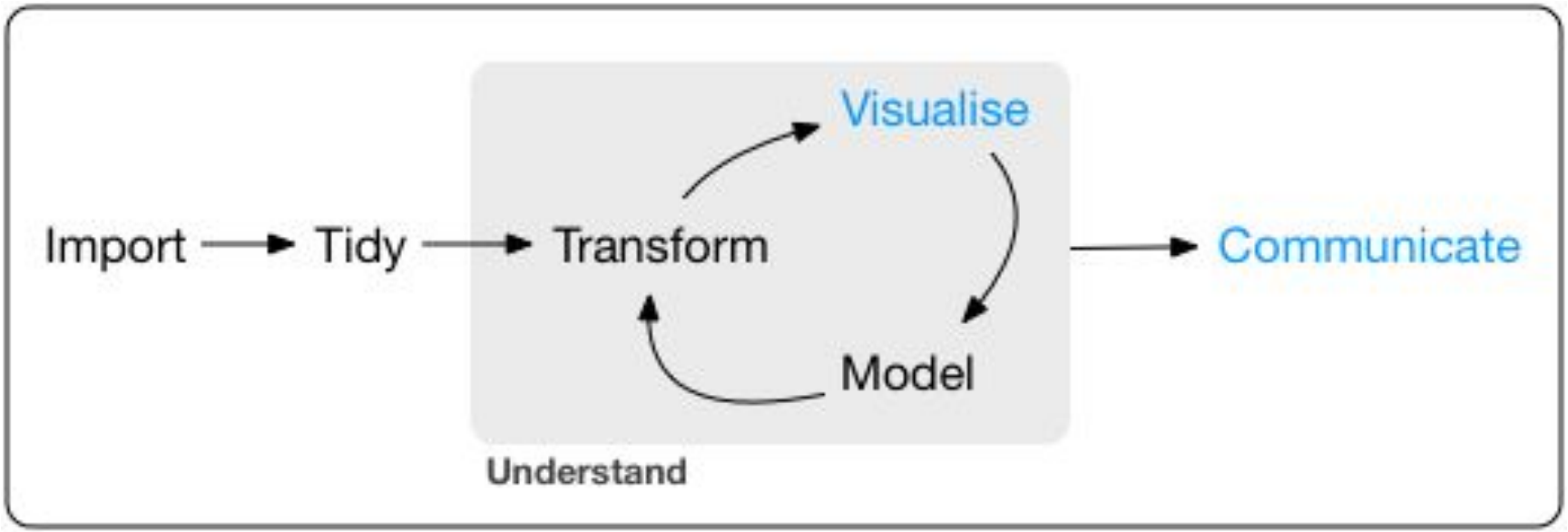
R for Data Science

Chapters 26 - 30

V COMMUNICATE

Chapter 26 Introduction


Visualise
Import
Tidy
Transform
Communicate
Model
Understand
Program

Chapter 27 R Markdown

R Markdown : : CHEAT SHEET

What is R Markdown?

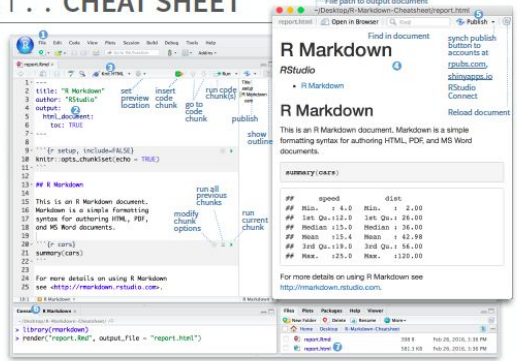
.Rmd files - An R Markdown (.Rmd) file is a record of your research. It contains the code that a scientist needs to reproduce your work along with the narration that a reader needs to understand your work.

Reproducible Research - At the click of a button, or the type of a command, you can rerun the code in an R Markdown file to reproduce your work and export the results as a finished report.

Dynamic Documents - You can choose to export the finished report in a variety of formats, including html, pdf, MS Word, or RTF documents; html or pdf based slides, Notebooks, and more.

Workflow

1. **Open a new .Rmd file** at File ▶ New File ▶ R Markdown. Use the wizard that opens to pre-populate the file with a template
2. **Write document** by editing template
3. **Knit document to create report**; use knit button or **render()** to knit
4. **Preview Output** in IDE window
5. **Publish** (optional) to web server
6. **Examine build log** in R Markdown console
7. **Use output file** that is saved along side .Rmd



render

Use rmarkdown::**render()** to render/knit at cmd line. Important args:

input - file to render **output_format**	**output_options** - List of render options (as in YAML)	**output_file** **output_dir**	**params** - list of params to use	**envir** - environment to evaluate code chunks in	**encoding** - of input file

Embed code with knitr syntax

INLINE CODE
Insert with `` `r <code>` ``. Results appear as text without code.
Built with `` `r getRversion()` `` ➡ Built with 3.2.3

CODE CHUNKS
One or more lines surrounded with ```` ```{r} ```` and ```` ``` ````. Place chunk options within curly braces, after **r**. Insert with
```{r echo=TRUE}
getRversion()
```
➡
```
getRversion()
## [1] '3.2.3'
```

GLOBAL OPTIONS
Set with knitr::**opts_chunk$set()**, e.g.
```{r include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

IMPORTANT CHUNK OPTIONS

cache - cache results for future knits (default = FALSE)

cache.path - directory to save cached results in (default = "cache/")

child - file(s) to knit and then include (default = NULL)

collapse - collapse all output into single block (default = FALSE)

comment - prefix for each line of results (default = '##')

dependson - chunk dependencies for caching (default = NULL)

echo - Display code in output document (default = TRUE)

engine - code language used in chunk (default = 'r')

error - Display error messages in doc (TRUE) or stop render when errors occur (FALSE) (default = FALSE)

eval - Run code in chunk (default = TRUE)

fig.align - 'left', 'right', or 'center' (default = 'default')

fig.cap - figure caption as character string (default = NULL)

fig.height, fig.width - Dimensions of plots in inches

highlight - highlight source code (default = TRUE)

include - Include chunk in doc after running (default = TRUE)

message - display code messages in document (default = TRUE)

results (default = 'markup')
'asis' - passthrough results
'hide' - do not display results
'hold' - put all results below all code

tidy - tidy code for display (default = FALSE)

warning - display code warnings in document (default = TRUE)

Options not listed above: R options, aniopts, autodep, background, cache.comments, cache.lazy, cache.rebuild, cache.vars, dev, dev.args, dpi, engine.opts, engine.path, fig.asp, fig.env, fig.ext, fig.keep, fig.lp, fig.path, fig.pos, fig.process, fig.retina, fig.scap, fig.show, fig.showtext, fig.subcap, interval, out.extra, out.height, out.width, prompt, purl, ref.label, render, size, split, tidy.opts

.rmd Structure

YAML Header
Optional section of render (e.g. pandoc) options written as key:value pairs (YAML).
At start of file
Between lines of ---

Text
Narration formatted with markdown, mixed with:

Code Chunks
Chunks of embedded code. Each chunk:
Begins with ```{r}
ends with ```
R Markdown will run the code and append the results to the doc.
It will use the location of the .Rmd file as the **working directory**

Parameters

Parameterize your documents to reuse with different inputs (e.g., data, values, etc.)

1. **Add parameters** - Create and set parameters in the header as sub-values of params

```
---
params:
  n: 100
  d: !r Sys.Date()
---
```

2. **Call parameters** - Call parameter values in code as params$<name>

Today's date is `r params$d`

3. **Set parameters** - Set values with Knit with parameters or the params argument of render():

render("doc.Rmd", params = list(n = 1, d = as.Date("2015-01-01"))

Interactive Documents

Turn your report into an interactive Shiny document in 4 steps

1. Add runtime: shiny to the YAML header.
2. Call Shiny input functions to embed input objects.
3. Call Shiny render functions to embed reactive output.
4. Render with rmarkdown::run or click Run Document in RStudio IDE.

```
---
output: html_document
runtime: shiny
---

```{r, echo = FALSE}
numericInput("n",
  "How many cars?", 5)

renderTable({
  head(cars, input$n)
})
```
```

How many cars? 5

	speed	dist
1	4.00	2.00
2	4.00	10.00
3	7.00	4.00
4	7.00	22.00
5	8.00	16.00

Embed a complete app into your document with shiny::**shinyAppDir()**.

NOTE: *Your report will rendered as a Shiny app, which means you must choose an html output format, like* ***html_document****, and serve it with an active R Session.*

RStudio

R Markdown Reference Guide

Learn more about R Markdown at rmarkdown.rstudio.com
Learn more about Interactive Docs at shiny.rstudio.com/articles

Contents:

Syntax	Becomes
`Plain text`	Plain text
`End a line with two spaces to start a new paragraph.`	End a line with two spaces to start a new paragraph.
`*italics* and _italics_`	*italics* and *italics*
`**bold** and __bold__`	**bold** and **bold**
`superscript^2^`	superscript^2
`~~strikethrough~~`	~~strikethrough~~
`[link](www.rstudio.com)`	link
`# Header 1`	**Header 1**
`## Header 2`	**Header 2**
`### Header 3`	**Header 3**
`#### Header 4`	**Header 4**
`##### Header 5`	**Header 5**
`###### Header 6`	**Header 6**
`endash: --`	endash: –
`emdash: ---`	emdash: —
`ellipsis: ...`	ellipsis: …
`inline equation: $A = \pi * r^{2}$`	inline equation: $A = \pi * r^2$
`image: ![](path/to/smallorb.png)`	image: 
`horizontal rule (or slide break):`	horizontal rule (or slide break):
`***`	
`> block quote`	> block quote

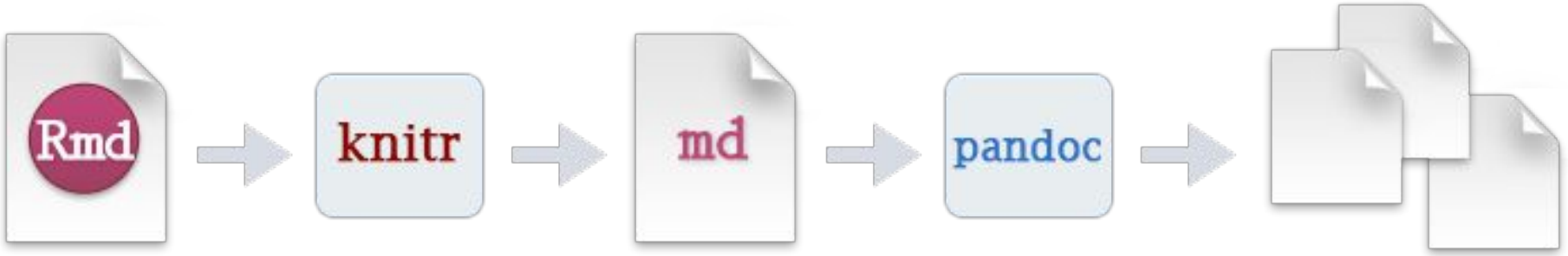
`* unordered list`
`* item 2`
`  + sub-item 1`
`  + sub-item 2`
	• unordered list
	• item 2
	  ◦ sub-item 1
	  ◦ sub-item 2

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

`Table Header | Second Header`
`------------ | -------------`
`Table Cell | Cell 2`
`Cell 3 | Cell 4`
	Table Header / **Second Header**
	Table Cell / Cell 2
	Cell 3 / Cell 4

Rmd files have three important types of content:

1. An (optional) **YAML header** surrounded by `---`.
2. **Chunks** of R code surrounded by ```.
3. Text mixed with simple text formatting like `#` heading and `_italics_`.


Rmd
knitr
md
pandoc

Chunk Details

- Names
  - ```` ```{r by-name} ````
- Options (defaults)
  - `eval = FALSE`
  - `include = FALSE`
  - `echo = FALSE`
  - `message = FALSE` or `warning = FALSE`
  - `results = "hide"` or `fig.show = "hide"`
  - `error = TRUE`

Option	Run code	Show code	Output	Plots	Messages	Warnings
`eval = FALSE`	-		-	-	-	-
`include = FALSE`		-	-	-	-	-
`echo = FALSE`		-				
`results = "hide"`			-			
`fig.show = "hide"`				-		
`message = FALSE`					-	
`warning = FALSE`						-

Chunk Details

- Tables
  - `kable()`
- Caching
  - `cache = TRUE`
  - `dependson = “other chunk name”`
  - `knitrcache.extra = file.info(“filename.csv”)`
  - `clean_cache()`
- Global Options
  - `opts_chunk$set()`

Inline Code

- `` `r ` ``
- `format()`

YAML Header

- Set parameters that can be used in the document with `params`

````
---
output: html_document
params:
  my_class: "suv"
---

```{r setup, include = FALSE}
library(ggplot2)
library(dplyr)

class <- mpg %>% filter(class == params$my_class)
```
````

YAML Header

- Use R code in parameters

```
params:
  start: !r lubridate::ymd("2015-01-01")
  snapshot: !r lubridate::ymd_hms("2015-01-01 12:30:00")
```

Bibliographies and Citations

- Add to YAML header `bibliography: rmarkdown.bib`
- Formats such as BibLaTeX, BibTeX, endnote, medline accepted
- Parenthetical citation: `As seen before [@smith04].`
- In-text citation citation: `As seen before by @smith04.`
- List of citations added to end of report

Chapter 28 Graphics for communication

ggplot2::labs()

- **`labs()`** - code variables labels (title, x, y)
- Annotations within `labs()`
  - **`title`** - text at top of plot
  - **`subtitle`** - adds additional detail in a smaller font beneath the title
  - **`caption`** - adds text at the bottom right of the plot, often used to describe the source of the dat
  - **`x`** - text displayed on the x-axis
  - **`y`** - text displayed on the y-axis
  - **`color`** / **`fill`** - the title of the legend

ggrepel

- **`geom_label_repeal`** - automatically adjusts labels do they don’t overlap

ggplot2::scales_*()

- There are **`scales_x_*()`**, **`scales_y_*()`**, and **`scales_color_*()`** among others
- Types of scales are (with example full functions):
  - `scale_x_`**`continuous`**`()`
  - `scale_x_`**`discrete`**`()`
  - `scale_x_`**`datetime`**`()`
  - `scale_x_`**`date`**`()`
  - `scale_x_`**`log10`**`()`
- To modify which numbers are ticks use the **`breaks`** argument
- To modify the text associated with ticks use the **`labels`** argument
- There are also several functions to choose colors, e.g **`scale_color_brewer()`**

Zooming

- “There are three ways to control the plot limits:
  - Adjusting what data are plotted
  - Setting the limits in each scale
  - Setting **`xlim`** and **`ylim`** in **`coord_cartesian()`**”

Themes

- Eight themes by default:
  - **`theme_bw()`** - white background with grid lines
  - **`theme_classic()`** - classy theme, axes but no grid lines
  - **`theme_dark()`** - dark background for contrast
  - **`theme_gray()`** - grey background (default theme)
  - **`theme_light()`** - light axes and grid lines
  - **`theme_linedraw()`** - only black lines
  - **`theme_minimal()`** - minimal theme, no background
  - **`theme_void()`** - empty theme, only geoms are visible

Chapter 29 R Markdown formats

Document Types

- **`pdf_document`** makes a PDF with LaTeX
- **`word_document`** for Microsoft Word documents (`.docx`)
- **`odt_document`** for OpenDocument Text documents (`.odt`)
- **`rtf_document`** for Rich Text Format (`.rtf`) documents
- **`md_document`** for a Markdown document
- **`github_document`** this is a tailored version of `md_document` designed for sharing on GitHub

Other Types of Renders

- **`html_notebook`** - a variation on a `html_document`
- Presentations:
  - **`ioslides_presentation`** - HTML presentation with ioslides
  - **`slidy_presentation`** - HTML presentation with W3C Slidy
  - **`beamer_presentation`** - PDF presentation with LaTeX Beamer
- **`flexdashboard::flex_dashboard`** - creates a dashboard for sharing

Interactivity and Other Formats

- Interactivity:
  - **`htmlwidgets`** - embed interactive HTML visualizations
  - **`shiny`** - create interactive R code hosted on the web
- Other formats:
  - **`bookdown`** - write books
  - **`prettydoc`** - lightweight document formats with a range of attractive themes
  - **`rticles`** - compiles a selection of formats tailored for specific scientific journals

Chapter 30 R Markdown workflow

Advice on Lab Notebooks / Workflow

- “Ensure each notebook has a descriptive title, an evocative filename, and a first paragraph that briefly describes the aims of the analysis.

- “Use the YAML header date field to record the date you started working on the notebook:

  ```
  date: 2016-08-23
  ```

  Use ISO8601 YYYY-MM-DD format so that’s there no ambiguity. Use it even if you don’t normally write dates that way!

Advice on Lab Notebooks / Workflow

- “If you spend a lot of time on an analysis idea and it turns out to be a dead end, don’t delete it! Write up a brief note about why it failed and leave it in the notebook. That will help you avoid going down the same dead end when you come back to the analysis in the future.

- “Generally, you’re better off doing data entry outside of R. But if you do need to record a small snippet of data, clearly lay it out using `tibble::tribble()`.

- “If you discover an error in a data file, never modify it directly, but instead write code to correct the value. Explain why you made the fix.

Advice on Lab Notebooks / Workflow

- “Before you finish for the day, make sure you can knit the notebook (if you’re using caching, make sure to clear the caches). That will let you fix any problems while the code is still fresh in your mind.

- “If you want your code to be reproducible in the long-run (i.e. so you can come back to run it next month or next year), you’ll need to track the versions of the packages that your code uses. A rigorous approach is to use **packrat**, which stores packages in your project directory, or **checkpoint**, which will reinstall packages available on a specified date. A quick and dirty hack is to include a chunk that runs `sessionInfo()` — that won’t let you easily recreate your packages as they are today, but at least you’ll know what they were.

Advice on
Lab Notebooks / Workflow

- “You are going to create many, many, many analysis notebooks over the course of your career. How are you going to organise them so you can find them again in the future? I recommend storing them in individual projects, and coming up with a good naming scheme.”