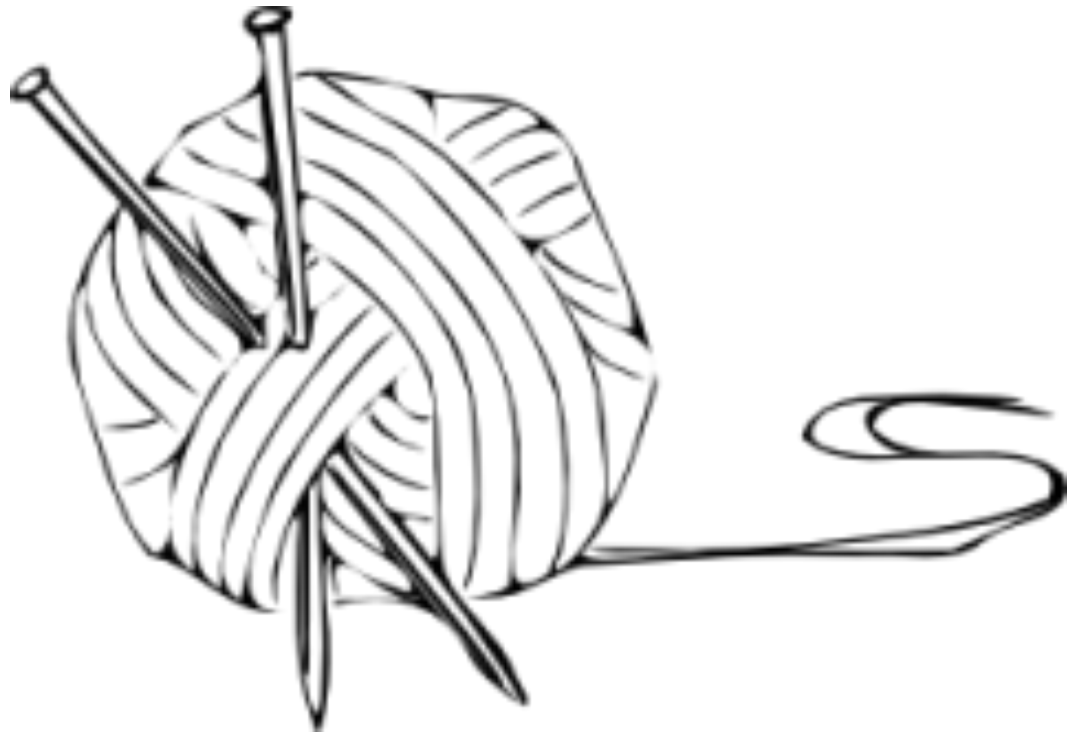
Structuring your analysis with R Markdown

Mine Çetinkaya-Rundel

@minebocek
mine-cetinkaya-rundel
mine@stat.duke.edu

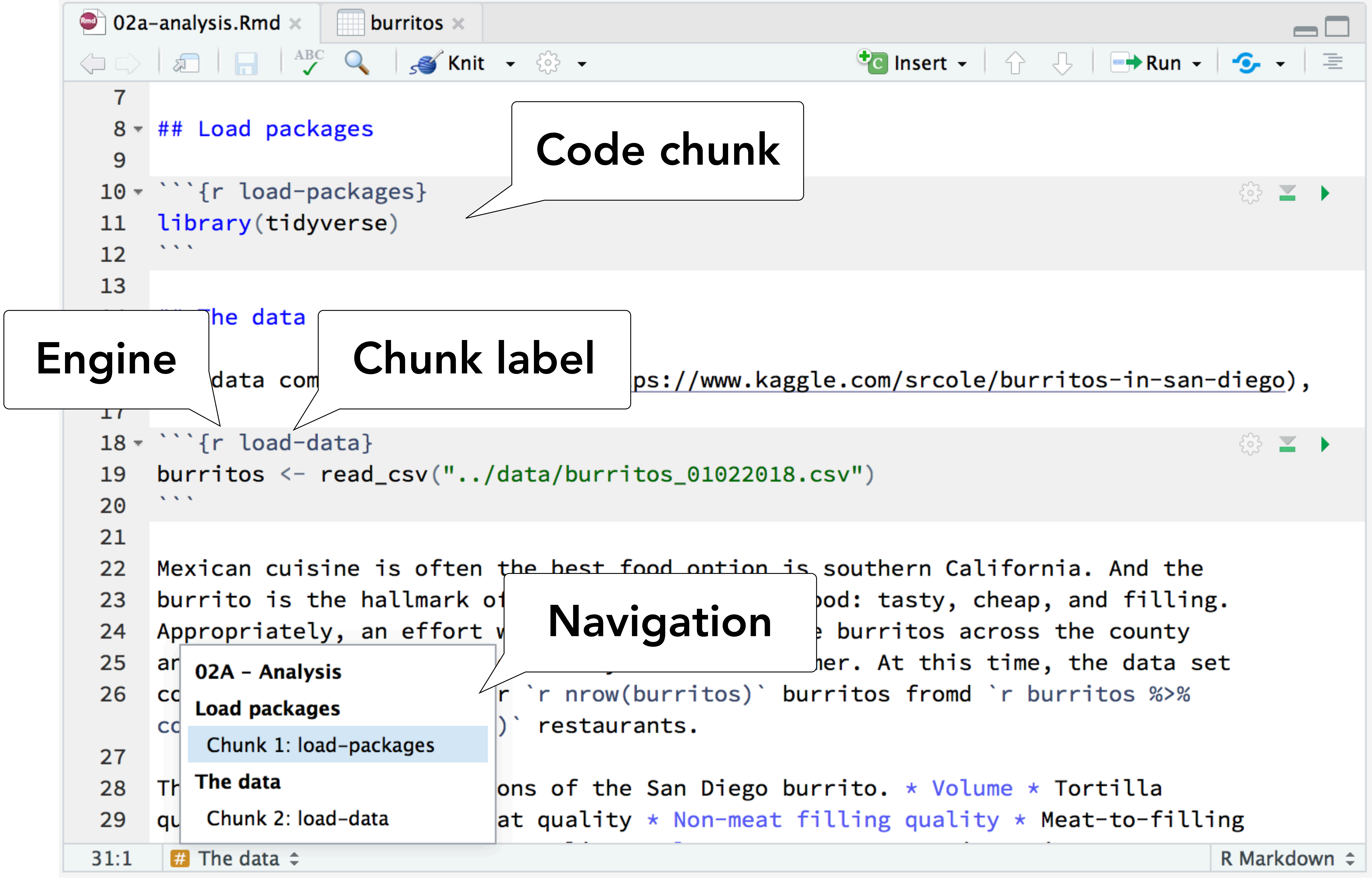
Embedding R code



Code chunks

- Insert new chunks with
  - the Add Chunk button in the editor toolbar
  - typing the chunk delimiters `` ```{r} `` and `` ``` ``
  - the keyboard shortcut **Ctrl + Alt/Option + I**
- When you render your .Rmd file, R Markdown will run each code chunk and embed the results beneath the code chunk in your final report.

DEMO



~~~
## Load packages

```{r load-packages}
library(tidyverse)
~~~

The data

data com   ps://www.kaggle.com/srcole/burritos-in-san-diego),

```{r load-data}
burritos <- read_csv("../data/burritos_01022018.csv")
```

Mexican cuisine is often the best food option is southern California. And the
burrito is the hallmark of   od: tasty, cheap, and filling.
Appropriately, an effort w   e burritos across the county
ar   her. At this time, the data set
co   r `r nrow(burritos)` burritos fromd `r burritos %>%
co   )` restaurants.

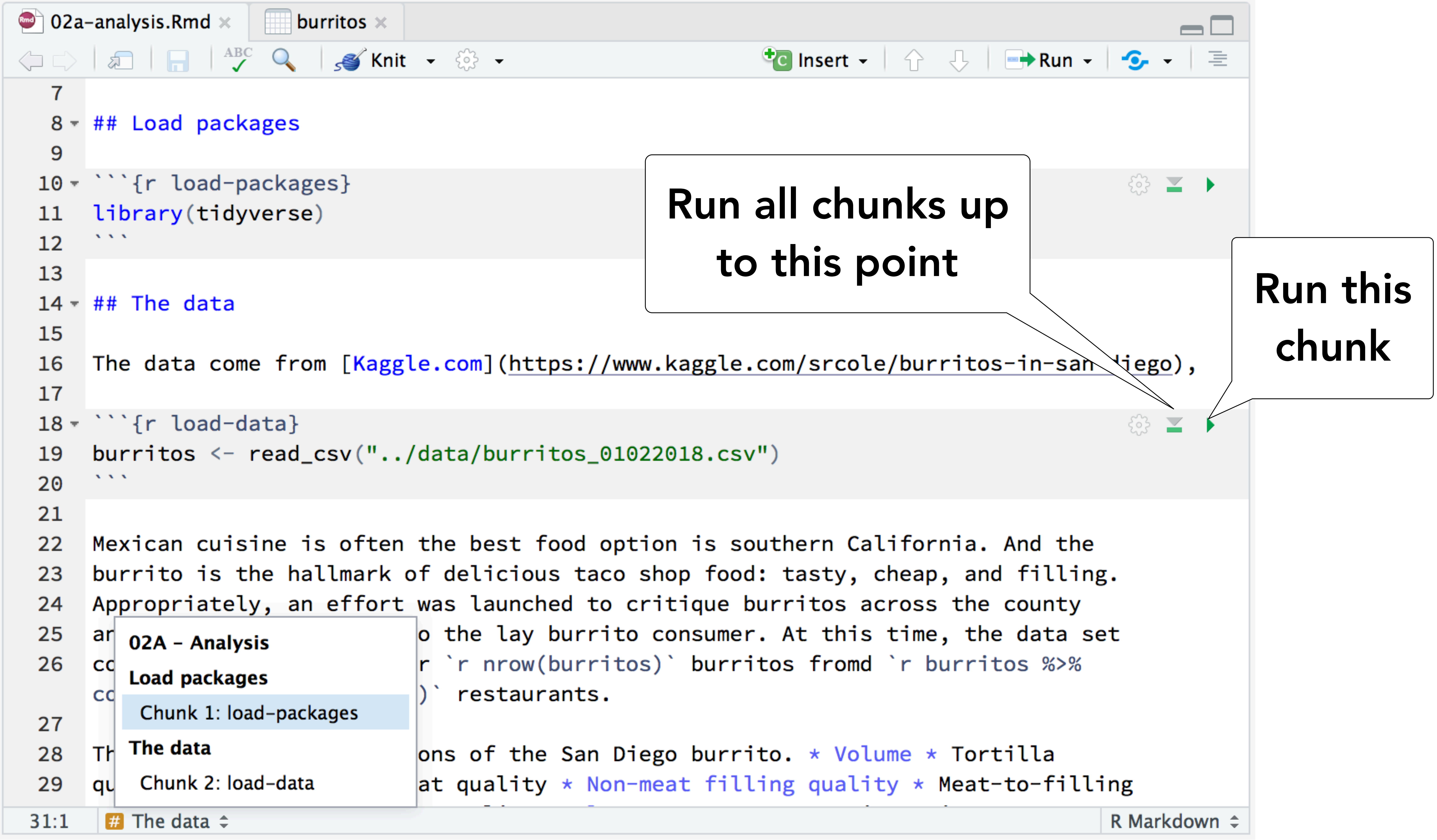
Th   ons of the San Diego burrito. * Volume * Tortilla
qu   at quality * Non-meat filling quality * Meat-to-filling
```

Code chunk

Engine

Chunk label

Navigation
```



~~~
## Load packages

```{r load-packages}
library(tidyverse)
```

The data

The data come from [Kaggle.com](https://www.kaggle.com/srcole/burritos-in-san-diego),

```{r load-data}
burritos <- read_csv("../data/burritos_01022018.csv")
```

Mexican cuisine is often the best food option is southern California. And the
burrito is the hallmark of delicious taco shop food: tasty, cheap, and filling.
Appropriately, an effort was launched to critique burritos across the county
ar   o the lay burrito consumer. At this time, the data set
co   r `r nrow(burritos)` burritos fromd `r burritos %>%
co   )` restaurants.

Th   ons of the San Diego burrito. * Volume * Tortilla
qu   at quality * Non-meat filling quality * Meat-to-filling
~~~

Run all chunks up to this point

Run this chunk

Tips

- Avoid spaces in chunk labels, even though technically they are “allowed”, especially if you work with GitHub — *more on this later!*
- If you’re having a hard time coming up with a short label that describes what the chunk is doing, consider breaking it down into shorter chunks ➡ especially useful for troubleshooting!

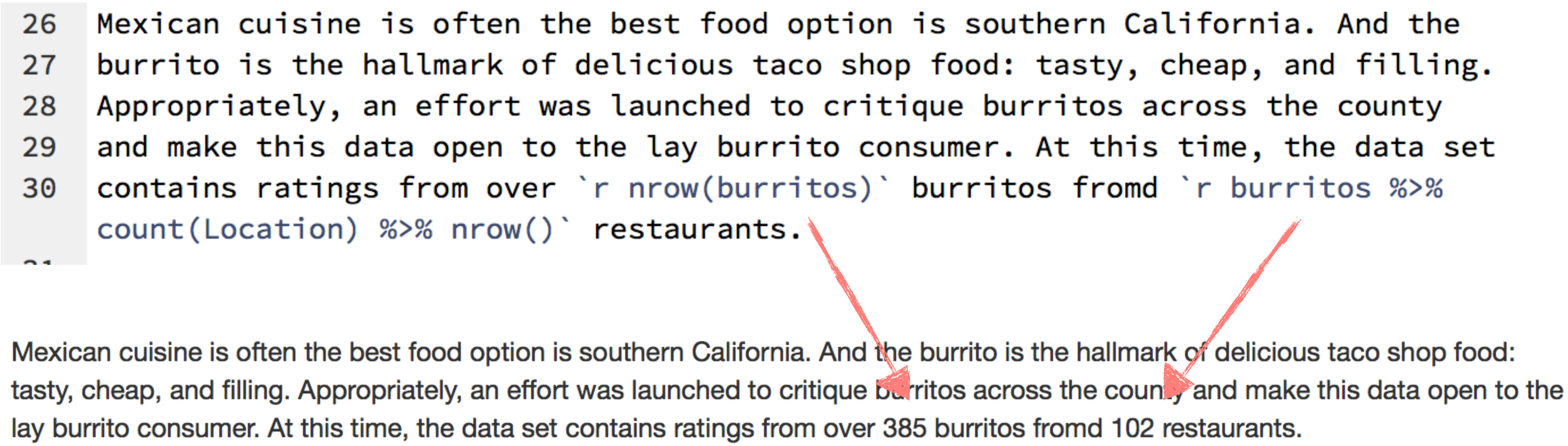
What happens when there’s an error in the R code in a chunk? What feedback/error does R give?

What about when there are multiple chunks with errors?

?

Inline code

Code results can be inserted directly into the text of a R Markdown file by enclosing the code with `` `r  ` ``:



```
Mexican cuisine is often the best food option is southern California. And the
burrito is the hallmark of delicious taco shop food: tasty, cheap, and filling.
Appropriately, an effort was launched to critique burritos across the county
and make this data open to the lay burrito consumer. At this time, the data set
contains ratings from over `r nrow(burritos)` burritos fromd `r burritos %>%
count(Location) %>% nrow()` restaurants.
```

Mexican cuisine is often the best food option is southern California. And the burrito is the hallmark of delicious taco shop food: tasty, cheap, and filling. Appropriately, an effort was launched to critique burritos across the county and make this data open to the lay burrito consumer. At this time, the data set contains ratings from over 385 burritos fromd 102 restaurants.

Inline code styling

- R Markdown will always
  - display the results of inline code, but not the code
  - apply relevant text formatting to the results
- As a result, inline output is indistinguishable from the surrounding text
- Inline expressions do not take knitr options

Your turn

- Open `sd-burritos.Rmd`, knit the document, and view the rendered file to get a sense of the data and the analysis.

- Add an inline R chunk to your document. This should be something that results in a numerical value or a character string.

- Add another inline R chunk and put some code in here that would result in a plot. What does a plot defined in an inline code chunk look like? Compare your answer with your neighbors'.

- **Stretch goal:** Update the date field in the YAML so that the date at the time of knitting the document is printed.

5m 00s

Chunk options

Safe to play with:

- `echo = FALSE`: code runs, code doesn’t appear in rendered file, results do ➡ for when your audience doesn’t need to see the code
- `include = FALSE`: code runs but neither code nor results appear in rendered file ➡ results can be used by other chunks
- `eval = FALSE`: code doesn’t run but they appear in the rendered file ➡ when you want to show/teach code

Requires care:

- `message = FALSE`: hides messages ➡ especially useful/harmful for loading packages and data
- `warning = FALSE`: hides messages ➡ especially useful/harmful for functions that throw many warnings

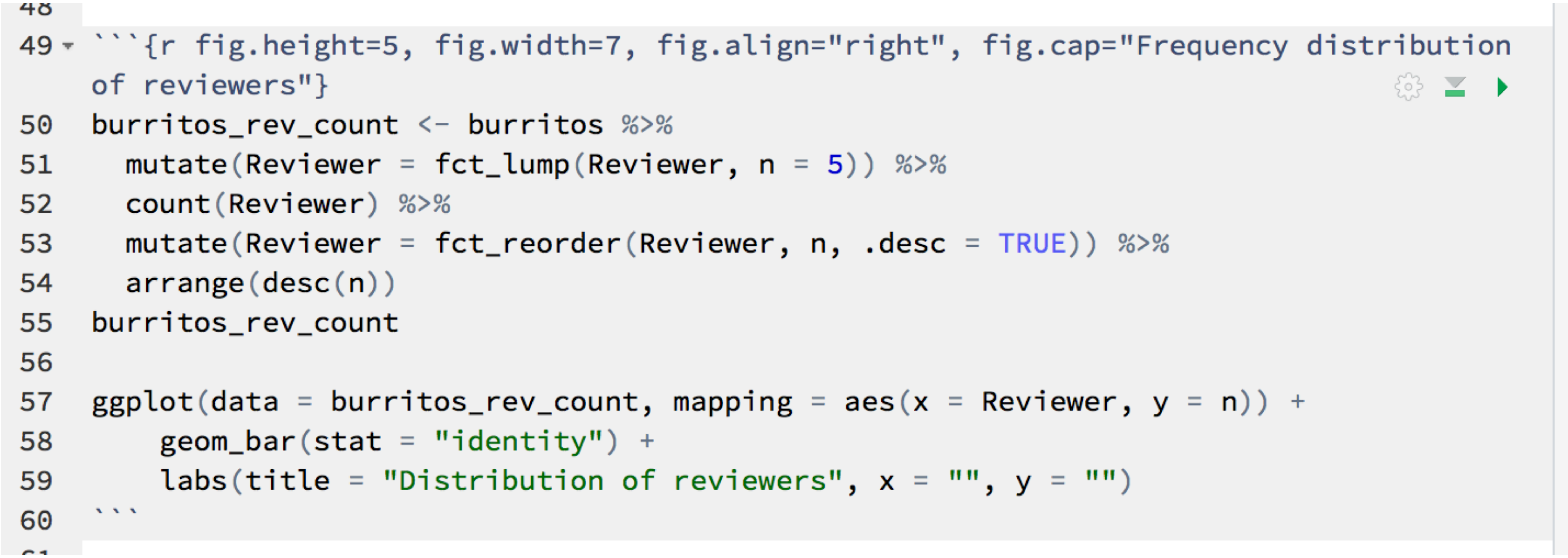
Your turn

- Add labels to each chunk.
- Change the chunk options (`echo`, `eval`, `include`, `message`, `warning`) to explore what changes in the output. Then, decide on an appropriate option for each of the chunks. Compare your choices to your neighbors'.
- **Stretch goal:** What does the option `collapse` (set to `TRUE` or `FALSE`) do? What is the default setting for this option? Which code chunk(s) does changing this option affect?

10m 00s

Chunk options for figures

- `fig.height`: Height in inches
- `fig.width`: Width in inches
- `fig.align`: “`left`”, “`right`”, or “`center`”
- `fig.cap`: Figure caption as character string

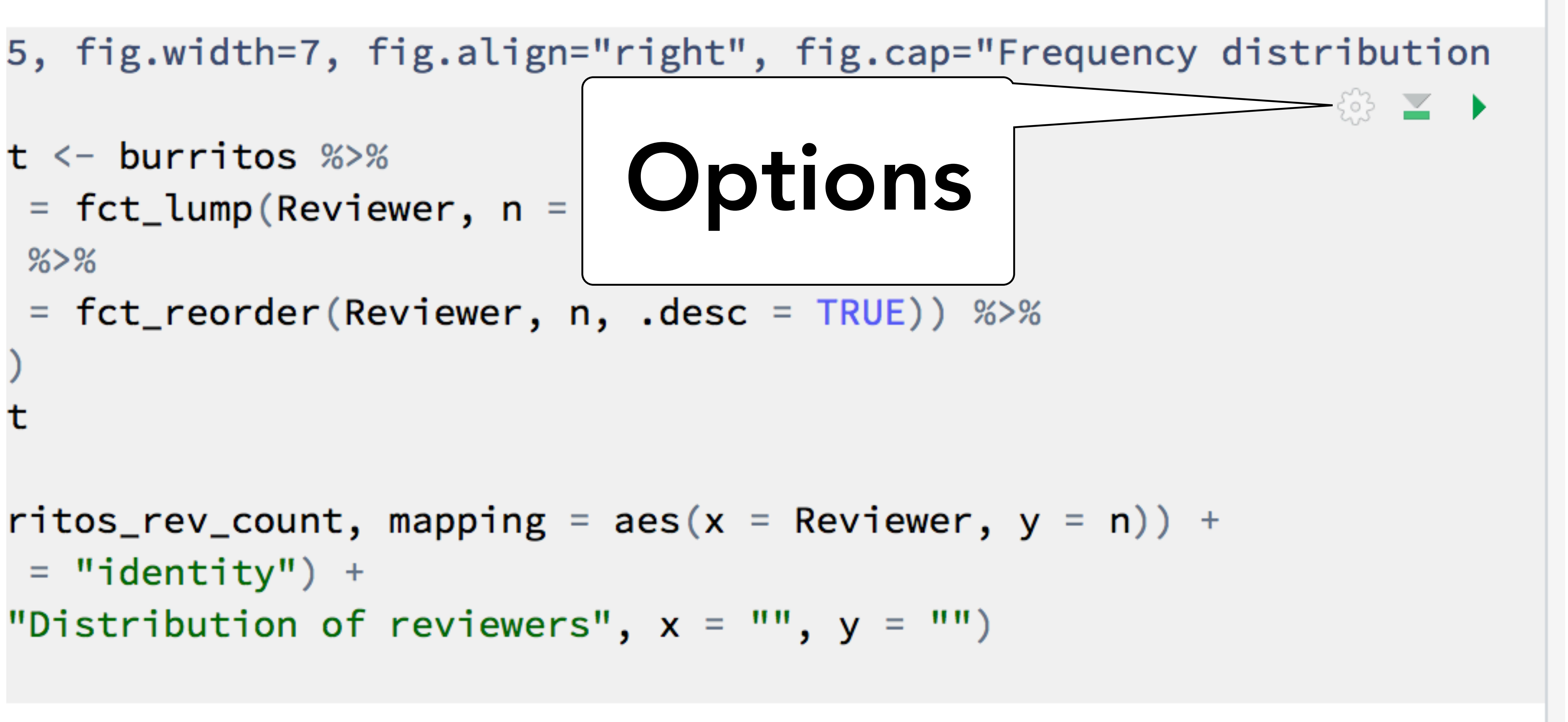


~~~
```{r fig.height=5, fig.width=7, fig.align="right", fig.cap="Frequency distribution of reviewers"}
burritos_rev_count <- burritos %>%
  mutate(Reviewer = fct_lump(Reviewer, n = 5)) %>%
  count(Reviewer) %>%
  mutate(Reviewer = fct_reorder(Reviewer, n, .desc = TRUE)) %>%
  arrange(desc(n))
burritos_rev_count

ggplot(data = burritos_rev_count, mapping = aes(x = Reviewer, y = n)) +
    geom_bar(stat = "identity") +
    labs(title = "Distribution of reviewers", x = "", y = "")
```
~~~

Setting chunk options via GUI

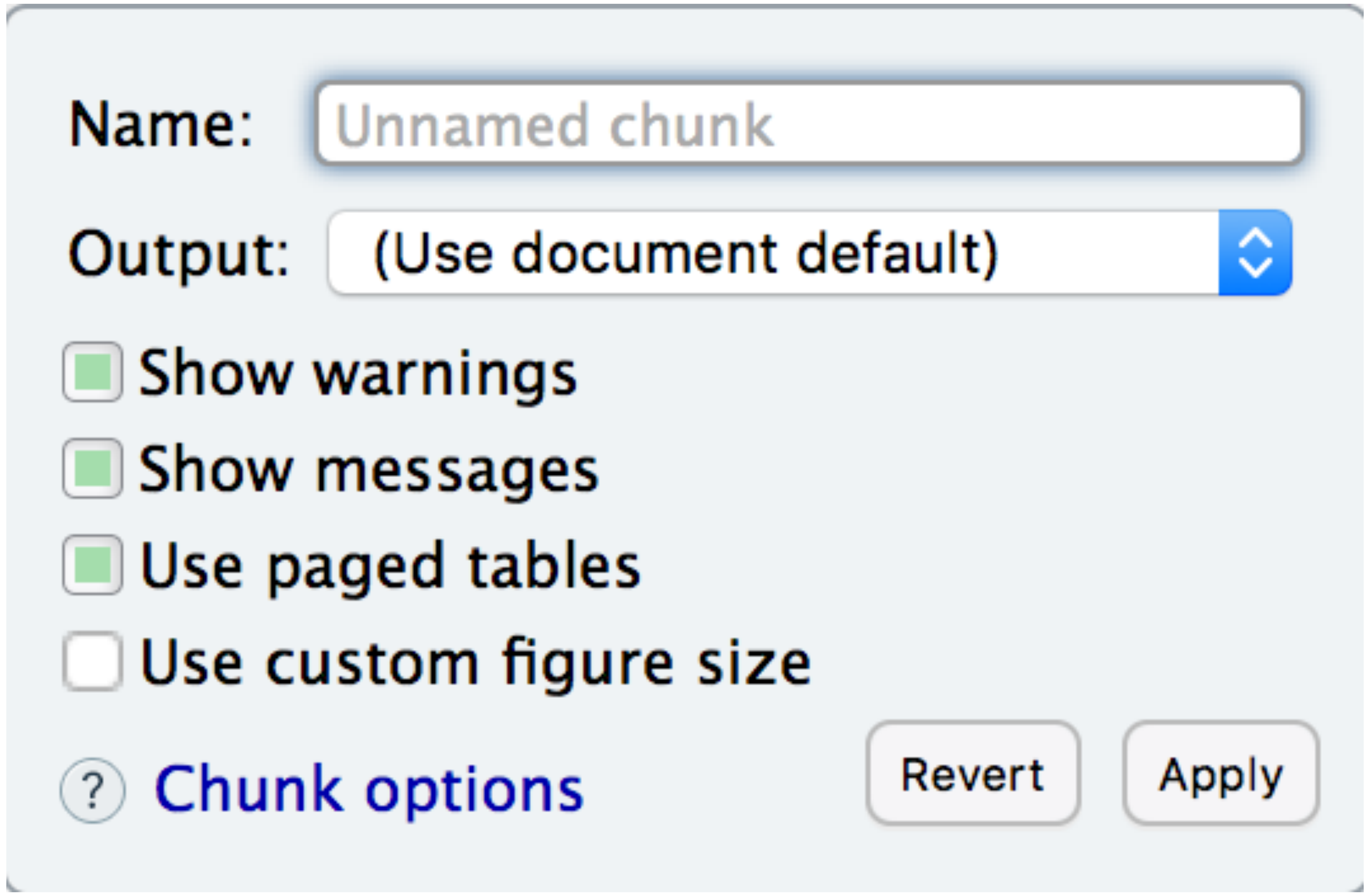
Some of the chunk options can be set via a handy GUI that you can access by clicking on the gear icon on a given chunk.



```
5, fig.width=7, fig.align="right", fig.cap="Frequency distribution

t <- burritos %>%
 = fct_lump(Reviewer, n =
 %>%
 = fct_reorder(Reviewer, n, .desc = TRUE)) %>%
)
t

ritos_rev_count, mapping = aes(x = Reviewer, y = n)) +
 = "identity") +
"Distribution of reviewers", x = "", y = "")
```



Name: Unnamed chunk

Output: (Use document default)

☑ Show warnings

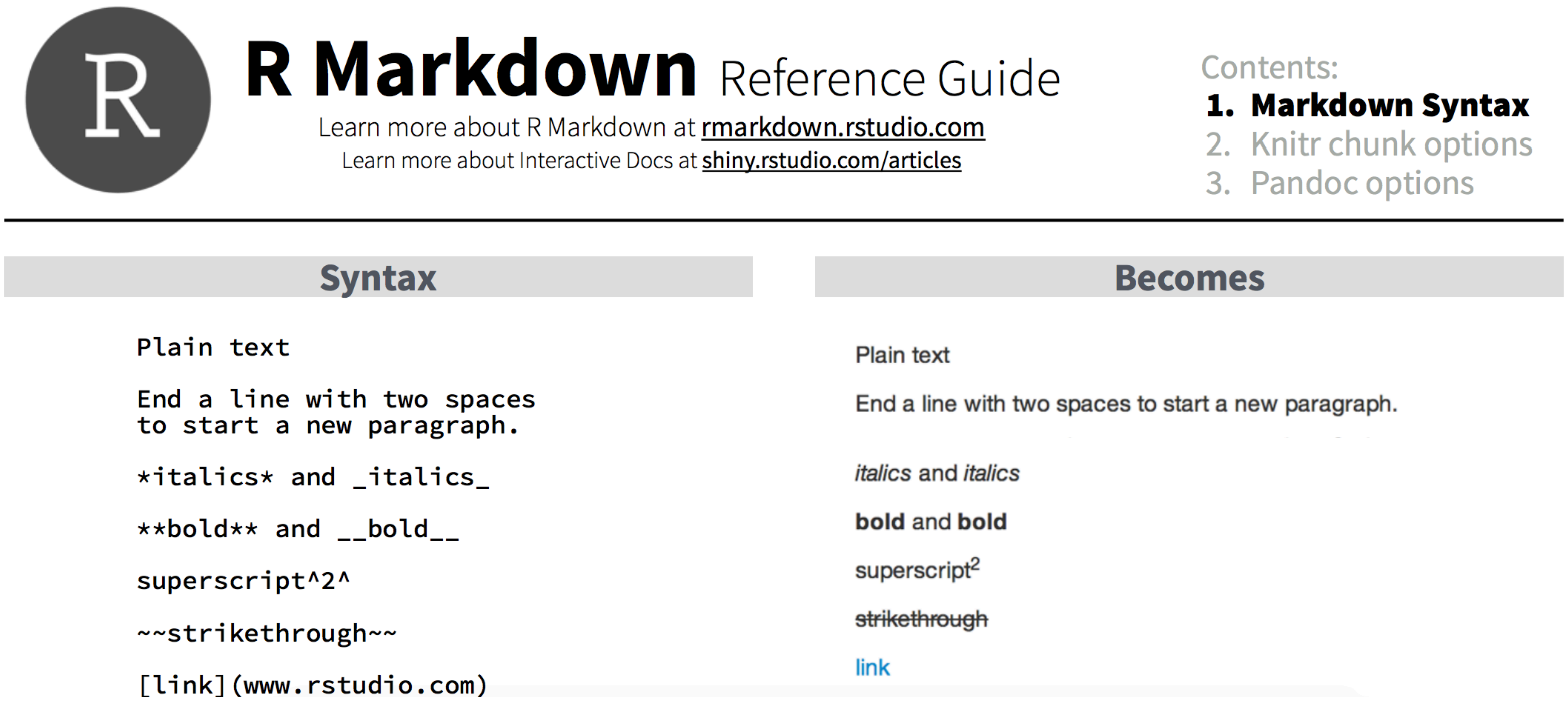
☑ Show messages

☑ Use paged tables

□ Use custom figure size

So many more chunk options!

https://www.rstudio.com/resources/cheatsheets/



R Markdown Reference Guide

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

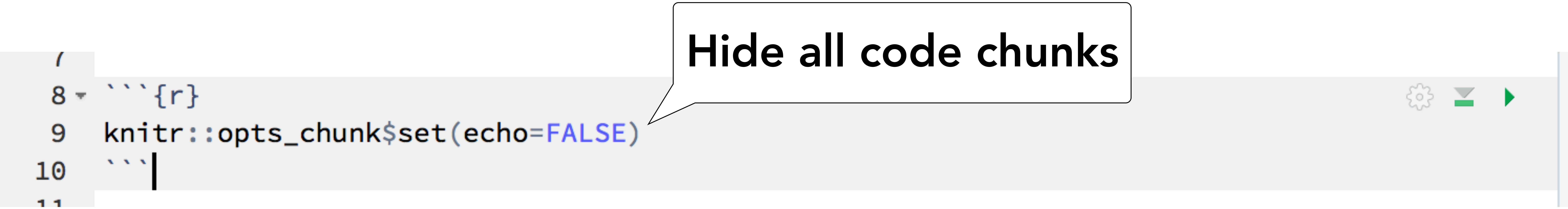
Contents:

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

rmarkdown

Global options

- To set global options that apply to every chunk in your file, call `knitr::opts_chunk$set` in a code chunk
- Knitr will treat each option that you pass to `knitr::opts_chunk$set` as a global default that can be overwritten in individual chunk headers



~~~
```{r}
knitr::opts_chunk$set(echo=FALSE)
```
~~~

Hide all code chunks

Your turn

- Add a new code chunk to `sd-burritos.Rmd` and set relevant options for that particular chunk. You create a plot, calculate summary statistics, or if you prefer, just do some basic calculation (without using the data).

- Remove the figure height and width options from individual chunks and set them as global options.

5m 00s

Caching

> There are only two hard things in Computer Science: cache invalidation and naming things.

–Phil Karlton

Martin Fowler, Two Hard Things. https://martinfowler.com/bliki/TwoHardThings.html.

Caching

- If document rendering becomes time consuming due to long computations you can use caching to improve performance
- If `cache = TRUE` is set:
  - Cached chunks are skipped, but objects created in these chunks are (lazy-) loaded from previously saved databases (`.rdb` and `.rdx`) files
  - These files are saved when a chunk is evaluated for the first time, or when cached files are not found
  - Results of the code will still be included in the output even when cache is used, because knitr also caches the printed output of a code chunk as a character string

Chunk options for caching

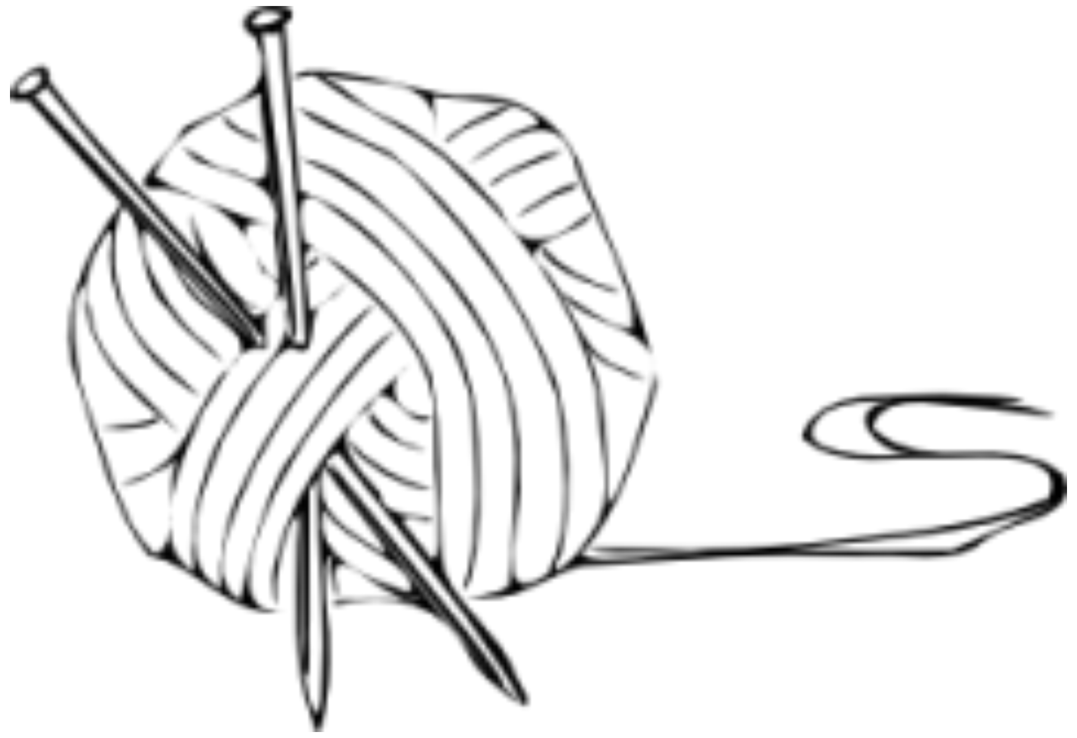
- `cache.path`: Directory to save cached results in (default = “`cache/`”)
- `dependson`: Chunk dependencies for caching (default = `NULL`)
- `...`

Your turn

- **Before you start:** Make sure all of your code chunks are labeled!
- Add the following to the chunk that creates the bar plot: `Sys.time(60)`
- Turn on caching for the code chunk that creates the bar plot by setting the relevant chunk option. Knit the document (this is going to take a while, a bit more than 60 seconds). Take a look at the folder where `sd-burritos.Rmd` lives. What else is new there?
- Knit the document again without making any changes to this particular code chunk. How long did knitting the document take the second time around?
- Add a code chunk before this one and slice and overwrite the data so that `burritos` is now just the first 50 observations in the original burritos dataset. Knit the document again. What is the problem?
- **Stretch goal:** How do you fix this?

10m 00s

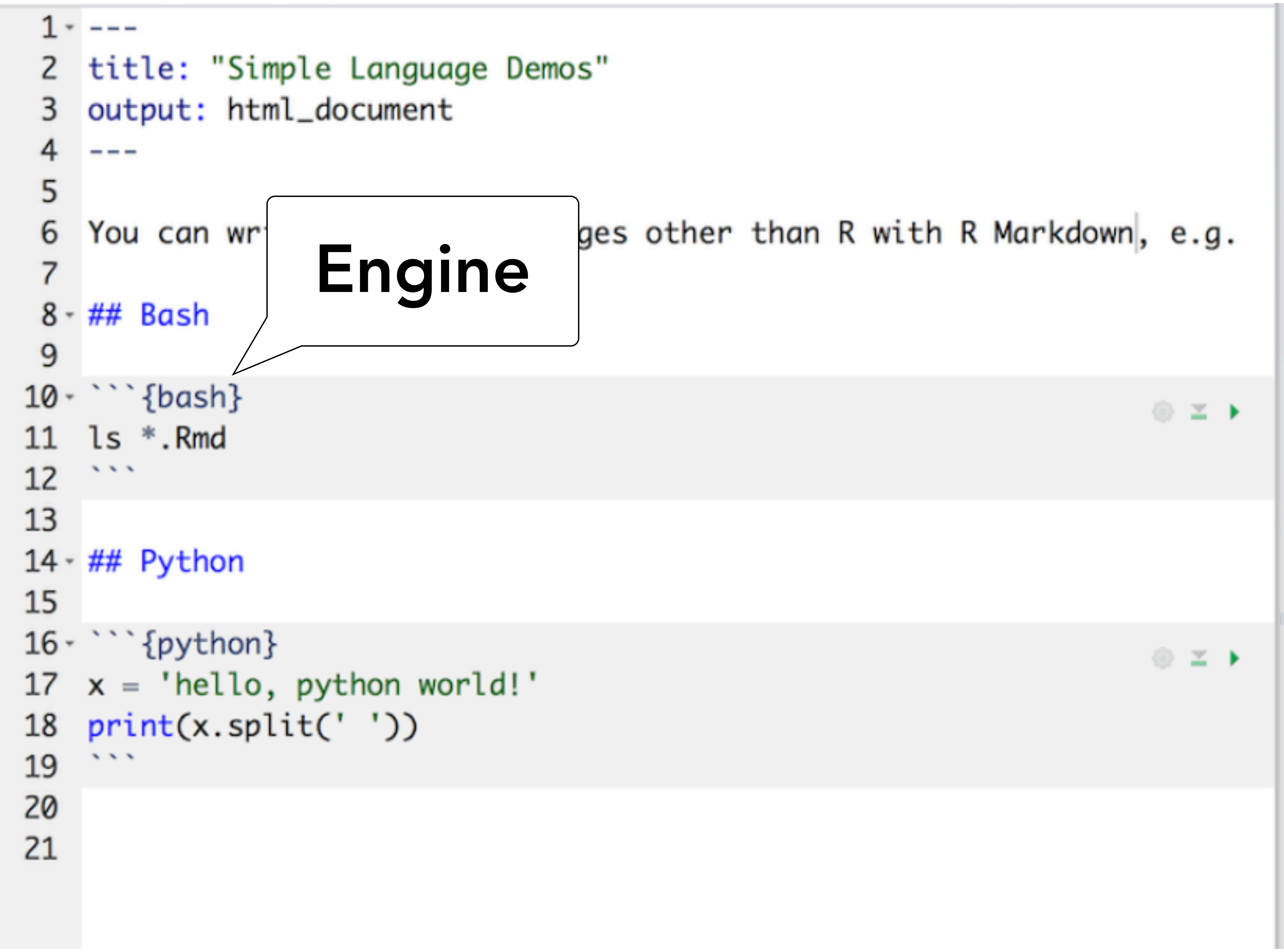
Other languages



Other languages

knitr can execute code in many languages besides R. Some of the available language engines include:

- Python
- SQL
- Bash
- Rcpp
- Stan
- JavaScript
- CSS



~~~
---
title: "Simple Language Demos"
output: html_document
---

You can wr                    ges other than R with R Markdown, e.g.

## Bash

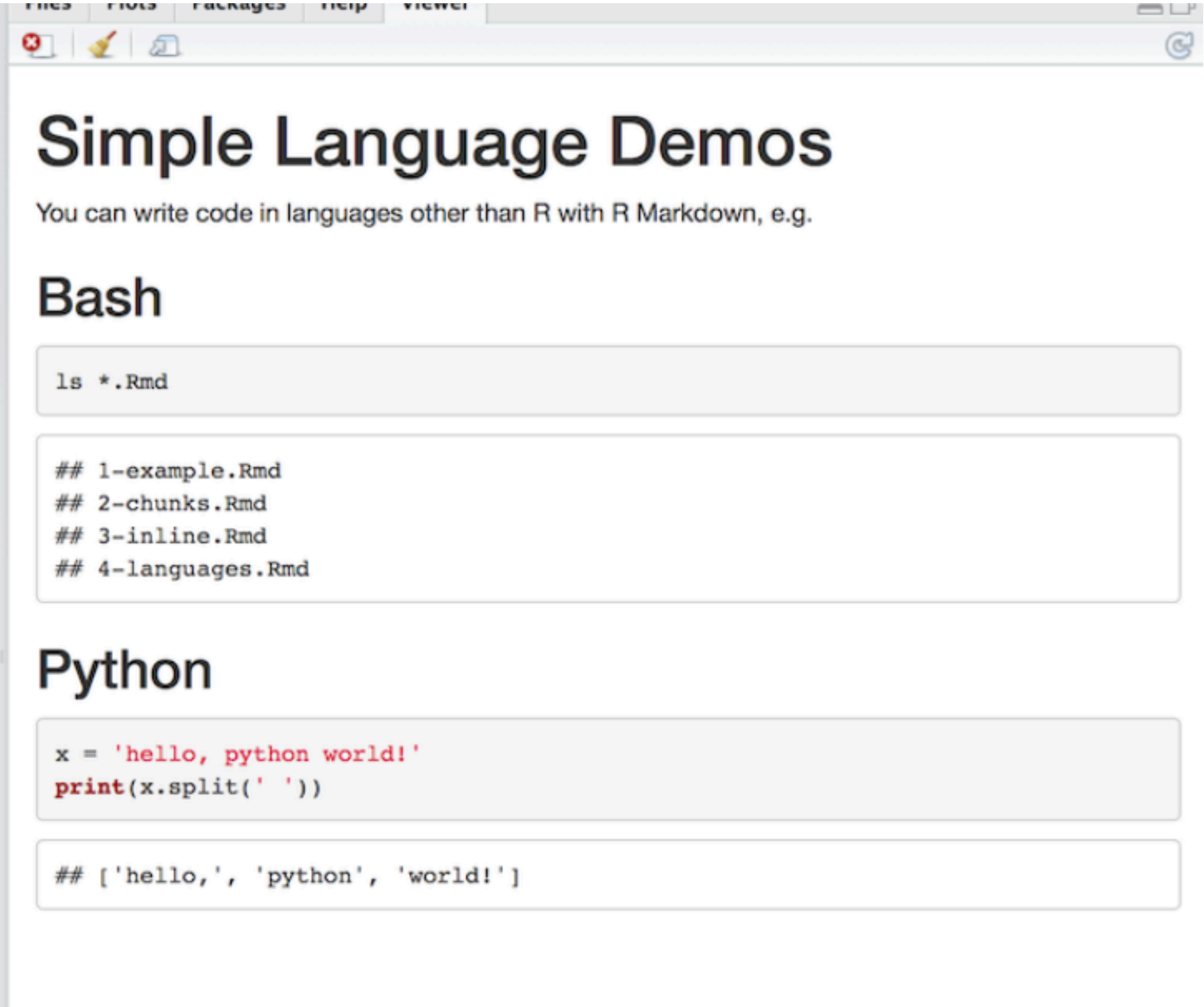
```{bash}
ls *.Rmd
~~~

Python

```{python}
x = 'hello, python world!'
print(x.split(' '))
```

```

Engine



Files
Plots
Packages
Help
Viewer

# Simple Language Demos

You can write code in languages other than R with R Markdown, e.g.

## Bash

```
ls *.Rmd
```

```
## 1-example.Rmd
## 2-chunks.Rmd
## 3-inline.Rmd
## 4-languages.Rmd
```

Python

```
x = 'hello, python world!'
print(x.split(' '))
```

```
## ['hello,', 'python', 'world!']
```
```

Output options

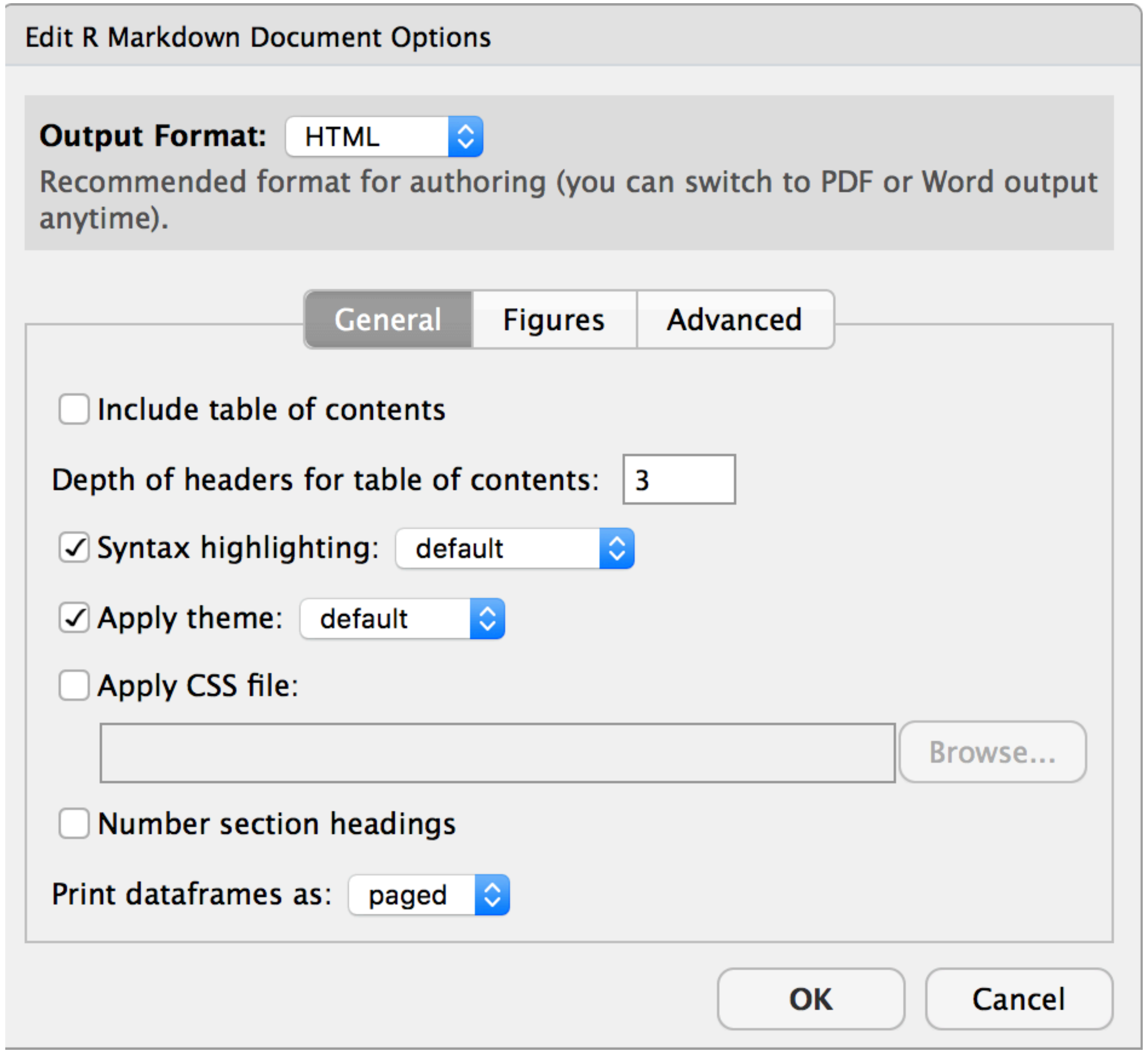
Output options

- Output options are defined in the YAML
- They are similar to setting global knitr options
- To learn which arguments a format takes, read the format's help page in R, e.g. `?html_document`

```
html_notebook(toc = FALSE, toc_depth = 3, toc_float = FALSE,
  number_sections = FALSE, fig_width = 7, fig_height = 5,
  fig_retina = 2, fig_caption = TRUE, code_folding = "show",
  smart = TRUE, theme = "default", highlight = "textmate",
  mathjax = "default", extra_dependencies = NULL, css = NULL,
  includes = NULL, md_extensions = NULL, pandoc_args = NULL,
  output_source = NULL, self_contained = TRUE, ...)
```

Setting output options via GUI

Some of the output options for some of the output types can be set via a handy GUI that you can access by clicking on the gear icon on the toolbar



Edit R Markdown Document Options

Output Format: HTML

Recommended format for authoring (you can switch to PDF or Word output anytime).

□ Include table of contents

Depth of headers for table of contents: 3

☑ Syntax highlighting: default

☑ Apply theme: default

□ Apply CSS file:

□ Number section headings

Print dataframes as: paged

Your turn

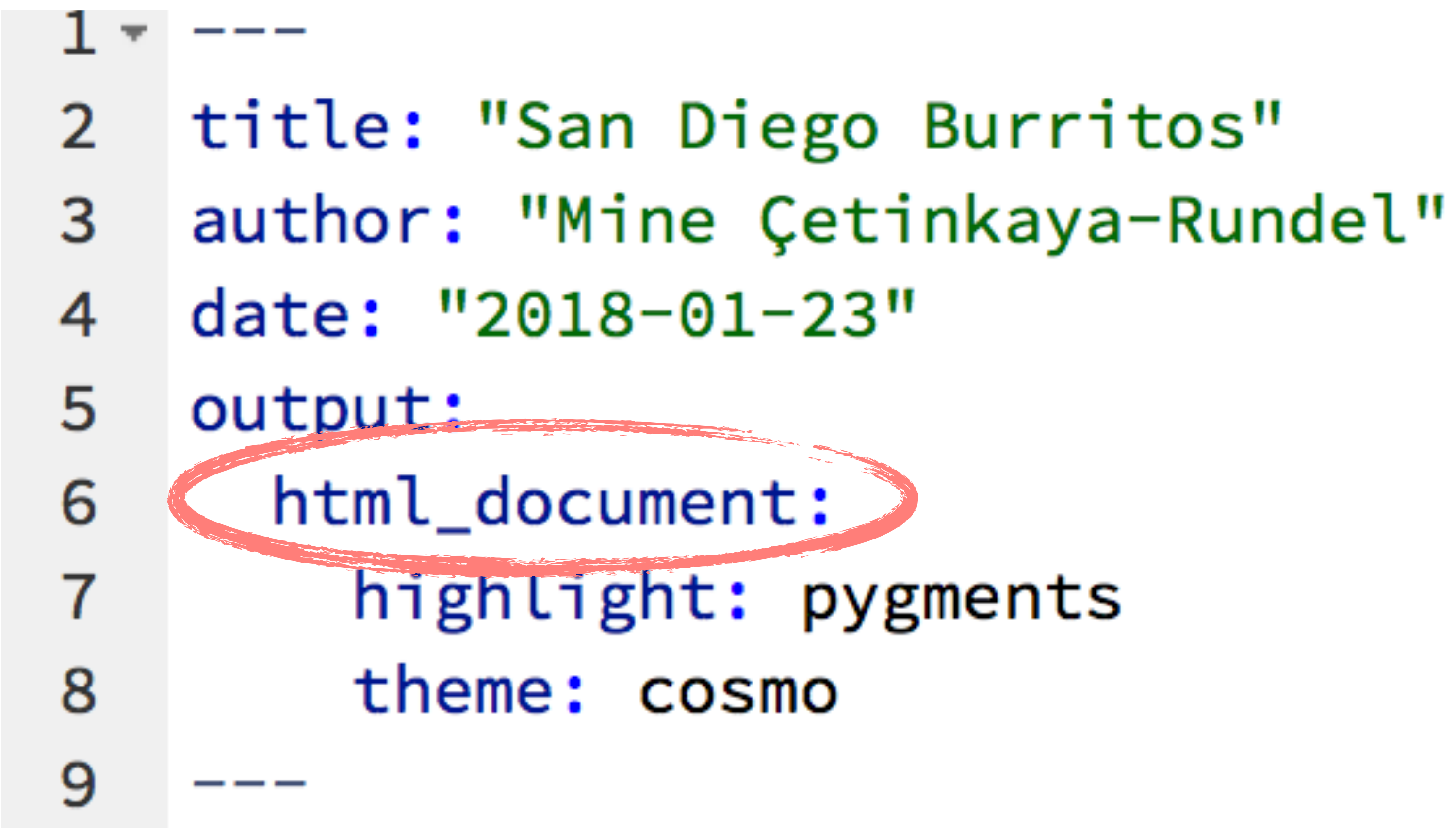
- Add a floating table of contents to the html document output.
- Also set all figures to be 4 x 7.

3m 00s

Output formats

Documents

- Most commonly used output is html_document — HTML document w/ Bootstrap CSS
- Other document options are as follows:
  - `html_notebook` - Interactive R Notebooks
  - `pdf_document` - PDF document (via LaTeX template)
  - `word_document` - Microsoft Word document (docx)
  - `odt_document` - OpenDocument Text document
  - `rtf_document` - Rich Text Format document
  - `md_document` - Markdown document (various flavors)



```
---
title: "San Diego Burritos"
author: "Mine Çetinkaya-Rundel"
date: "2018-01-23"
output:
  html_document:
    highlight: pygments
    theme: cosmo
---
```

Your turn

Convert `html_document` output to `html_notebook`. What changed?

Presentations

- `ioslides_presentation` - HTML presentation with ioslides
- `revealjs::revealjs_presentation` - HTML presentation with reveal.js
- `slidy_presentation` - HTML presentation with W3C Slidy
- `beamer_presentation` - PDF presentation with LaTeX Beamer
- `xaringan::moon_reader` - remark.js slides

Other output formats

- `flexdashboard::flex_dashboard` - Interactive dashboards
- `tufte::tufte_html` - HTML handouts in the style of Edward Tufte
- `html_vignette` - R package vignette (HTML)
- `github_document` - GitHub Flavored Markdown document

Your turn

- **Before you start:** Delete any files and folders created during the caching exercise. Turn off caching, and remove the `Sys.sleep(60)` command.

- Change output to `github_document`. Knit the document.

- What changed, other than cosmetic changes in the output? Discuss with your neighbors.

- **Stretch goal (if you are a git/GitHub user):** Push this document, and any other necessary files, so that it can be previewed on Github (with figures and output).

3m 00s

Your turn

- Decide which you want to work on: `xaringan` slides or `tufte_html`
- Make sure the package for the one you choose (xaringan or tufte) is installed and loaded
- Go to New File ➡ R Markdown… ➡ From Template, and then choose either Ninja Presentation (xaringan) or Tufte Handout (tufte)
- Convert `sd-burritos.Rmd` into one of these format