

Literate Programming

Reproducible Computing

@ JSM 2019

Colin Rundel

July 27, 2019

Literate Programming

Donald Knuth "Literate Programming (1983)"

"Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do."

"The practitioner of literate programming [...] strives for a program that is comprehensible because its concepts have been introduced in an order that is best for human understanding, using a mixture of formal and informal methods that reinforce each other."

- These ideas have been around for years!
- And tools for putting them to practice have also been around
- But they have never been as accessible as the current tools: R Markdown, Jupyter, etc.

What is Markdown?

- Markdown is a lightweight markup language for creating HTML (or XHTML) documents.
- Markup languages are designed to produce documents from human readable text (and annotations).
- Some of you may be familiar with LaTeX. This is another (less human friendly) markup language for creating pdf documents.
- Why I love Markdown:
 - Simple syntax means easy to learn and use.
 - Focus on **content**, rather than **coding** and **debugging**.
 - Allows for easy web authoring.
 - Once you have the basics down, you can get fancy and customize everything (via HTML, JavaScript, and CSS).

Sample Markdown document

```
---
_Advertisement :)_

- __[pica](https://nodeca.github.io/pica/demo/)_ - high quality and
fast image
  resize in browser.
- __[babelfish](https://github.com/nodeca/babelfish/)_ - developer
friendly
  i18n with plurals support and easy syntax.

You will like those projects!

---

# h1 Heading 8-)
## h2 Heading
### h3 Heading
#### h4 Heading
##### h5 Heading
##### h6 Heading

## Horizontal Rules

---

---

***

## Typographic replacements
```

Advertisement 😊

- [pica](#) - high quality and fast image resize in browser.
- [babelfish](#) - developer friendly i18n with plurals support and easy syntax.

You will like those projects!

h1 Heading 😎

h2 Heading

h3 Heading

h4 Heading

h5 Heading

h6 Heading

Horizontal Rules

Typographic replacements

```
## Code

Inline `code`

Indented code

    // Some comments
    line 1 of code
    line 2 of code
    line 3 of code

Block code "fences"
```
Sample text here...
```

Syntax highlighting
``` js
var foo = function (bar) {
 return bar++;
};

console.log(foo(5));
```

## Tables

Option	Description
data	path to data files to supply the data that will be passed
```

58. bar

Code

Inline `code`

Indented code

```
// Some comments
line 1 of code
line 2 of code
line 3 of code
```

Block code "fences"

```
Sample text here...
```

Syntax highlighting

```
var foo = function (bar) {
  return bar++;
};

console.log(foo(5));
```

Tables

| Option | Description |
|--------|---|
| data | path to data files to supply the data that will be passed into templates. |

What is R Markdown?

Well, it's R + Markdown:

- Ease of Markdown syntax
- Execution, rendering, and embedding of R code to produce output and plots
- Ability to include typeset mathematical expressions via LaTeX syntax: e.g.
$$\hat{y} = \beta_0 + \beta_1 \times x$$

Sample R Markdown document

```
---
_Avertisement :)_
- __[pica](https://nodeca.github.io/pica/demo/)__ - high quality and
fast image
  resize in browser.
- __[babelfish](https://github.com/nodeca/babelfish/)__ - developer
friendly
  i18n with plurals support and easy syntax.

You will like those projects!

---

# h1 Heading 8-)
## h2 Heading
### h3 Heading
#### h4 Heading
##### h5 Heading
##### h6 Heading

## Horizontal Rules

---

***

## Typographic replacements
```

Advertisement 😊

- **pica** - high quality and fast image resize in browser.
- **babelfish** - developer friendly i18n with plurals support and easy syntax.

You will like those projects!

h1 Heading 😎

h2 Heading

h3 Heading

h4 Heading

h5 Heading

h6 Heading

Horizontal Rules

Typographic replacements

Another R Markdown document

This presentation!

R Markdown

It's your lucky day!

You got some data.

- You are given a data file: `WorldCupMatches-01.csv`, it contains results for each match in World Cups before 2000.
- A codebook is included in `data/README.md`
- Goal: Visualize the total number of goals for each World Cup over time.

Open `world-cup-goals.Rmd`. Knit the document. Then, update the **yaml** with your information, and knit again.

The YAML

YAML: Yet another Markdown language

- Fields like `title`, `subtitle`, `author`, `date`
- You can also change output formats:
 - `html_document` for web authoring,
 - `github_document` for markdown documents which can be viewed on GitHub,
 - `pdf_document` for PDF (requires TeX),
 - `word_document` for MS Word (requires Word)
- Can use inline R code in values (see `date`)

Chunk options

- Turn off messages with `message = FALSE`
- Turn off warnings with `warning = FALSE`
- Hide code with `echo = FALSE`
- Exclude chunk from doc with `include = FALSE` to prevent code and results from appearing in the finished file. Code in the chunk will still be ran, and the results can be used by other chunks.
- Display error messages in document with `error = TRUE`, as opposed to stopping render when errors occur `error = FALSE`, which is the default
- Set these per chunk or globally in a `setup` chunk on top of the document with `knitr::opts_chunk$set(...)`

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`

Not so lucky after all

Turns out there is an error in the data you received: The number of `home_team_goals` in 1998 by Brazil (in the game vs. Denmark played on 03 Jul 1998) should be 3, not 0. Implement a fix and redo the analysis.

More data!

And now you received more data: World Cup matches post-2000. The data are in `data/WorldCupMatches-02.csv`. Redo the analysis combining data from both files.

Tips

- Make sure RStudio and the `rmarkdown` package (and its dependencies) are up-to-date.
- Get rid of your `.Rprofile`, especially if you have anything in there relating to `knitr`, `markdown`, `rmarkdown`, and RStudio.
- Set a global option for `error = TRUE` (or for a given chunk) so that your document renders even when there are errors.
- Don't try to change working directory within an R Markdown document. (If you do still decide to use `setwd` in a code chunk, beware that the new working directory will only apply to that specific code chunk, and any following code chunks will revert back to use the original working directory.)