R and R-markdown Basics

STAT 220

Bastola
2022-01-07

Replicability vs. Reproducibility

- Scientific findings should be **replicatable**
    - Asha repeats Bob's lab experiment and gets different data but makes the same conclusions as Bob
- Statistical findings should be **reproducible**
    - Asha takes Bob's data and gets the exact same statistical results as Bob
- Statistical findings should be *easily* **reproducible**
    - Asha only needs to hit one button to reproduce Bob's results.
    - Asha only needs to hit one button to reproduce Bob's analysis on a *new data set*

Reproducible data science

Short-term impact

- Are the tables and figures reproducible from the code and data?
- Does the code work as intended?
- In addition to what was done, is it clear **why** it was done? (e.g., how were parameter settings chosen?)

Long-term impact

- Can the code be used for other data?
- Can you extend the code to do other things?

Making your work reproducible

- You need a scriptable program (e.g. R, Python)
  - forces you to record the linear sequence of events in an analysis
  - avoid point-n-click!
  - avoid any "by hand" actions (e.g. data cleaning in Excel)

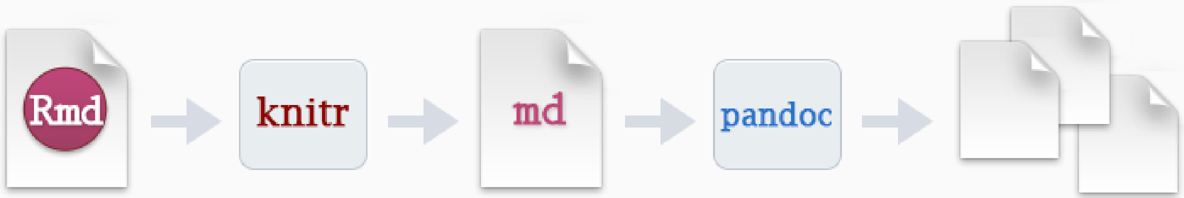
But scriptable doesn't always mean reproducible!!

- You should make your workflow transparent and easily followed, ⟶ R Markdown
  - meaningful file and variable names
  - don't overly complicate code, use packages when only when needed (the fewer dependencies the better)
  - only relevant code included
  - **written description of your analysis process and results alongside your code**

Karl Broman’s Steps toward Reproducible Research

Reproducibility using R markdown

- Karl Broman paraphrasing Mark Holder:
    - **Your closest collaborator is you six months ago, but you don't reply to emails.**
    - You need to document your workflow for both yourself and current/future collaborators
- R Markdown is a literate programming language that integrates R code, results and write-up.
    - literate = it is readable and easy to learn



Assignment-0 Recap

Show 6 entries Search:

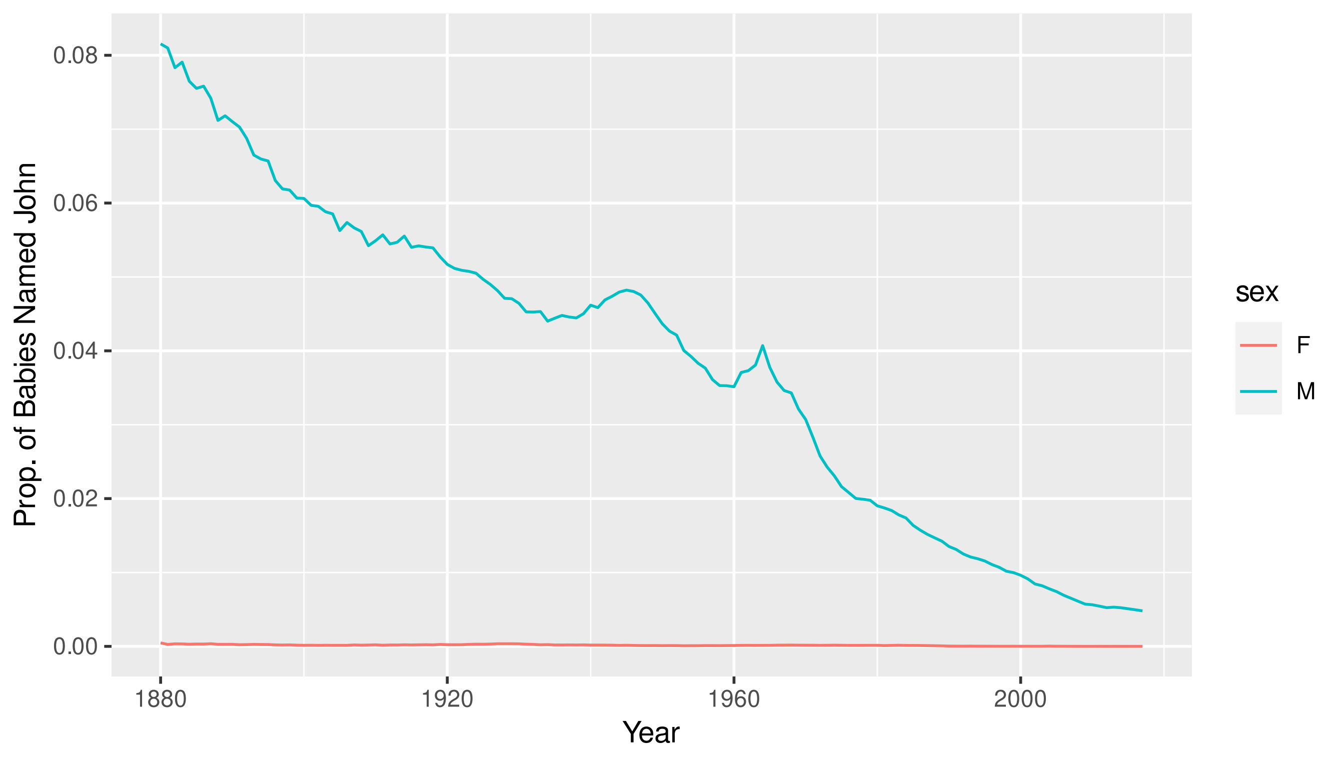
	year	sex	name	n	prop
1	1880	F	Mary	7065	0.07238359
2	1880	F	Anna	2604	0.02667896
3	1880	F	Emma	2003	0.02052149
4	1880	F	Elizabeth	1939	0.01986579
5	1880	F	Minnie	1746	0.01788843
6	1880	F	Margaret	1578	0.0161672

Showing 1 to 6 of 24 entries

Previous 1 2 3 4 Next

```
dim(babynames)
# [1] 1924665       5
```

Communication is important!



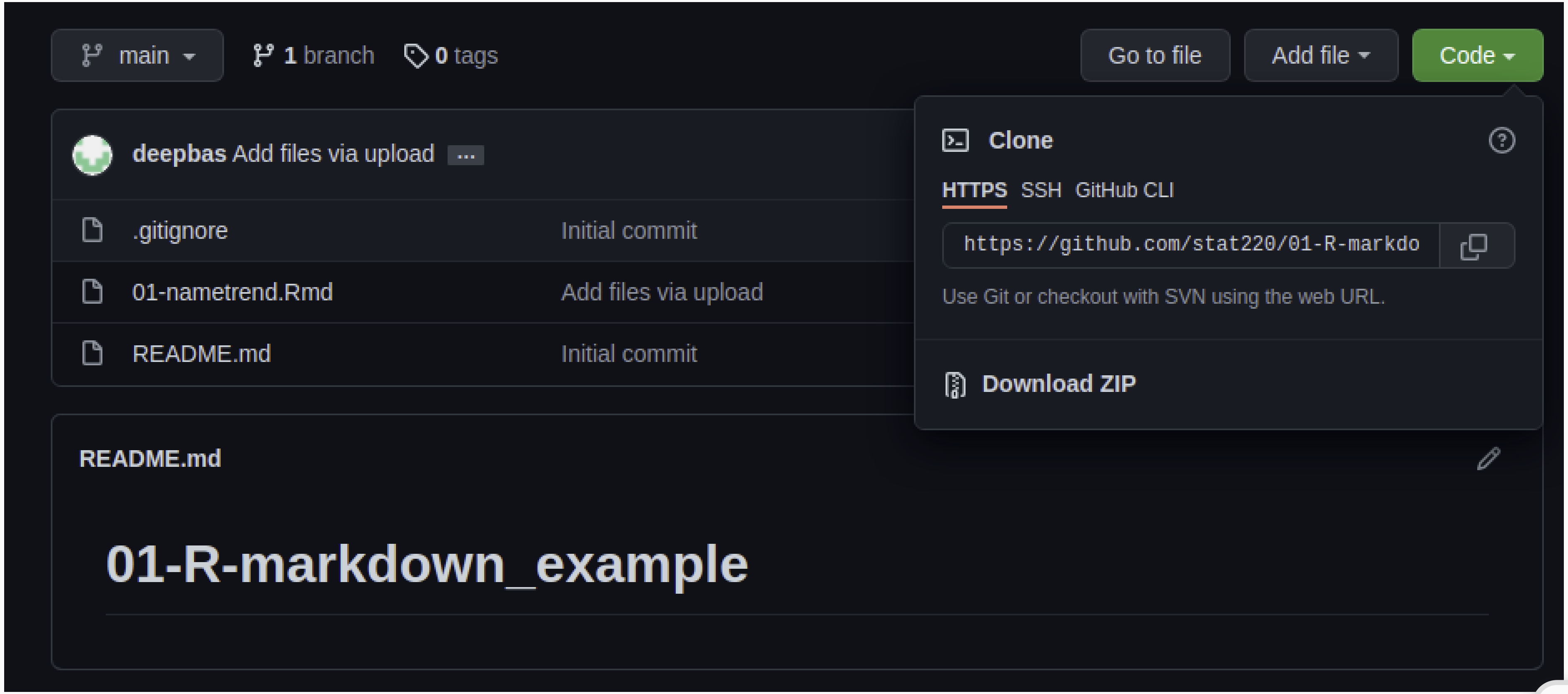
The overall trend of baby name `John` has been on a steady decline over the years.

Basic anatomy of R-markdown

1. The **metadata**
2. The **text**
3. The **code**
4. The **output**

Let's look at the source anatomy

Please git clone this repository to your local folder.



deepbas Add files via upload ...

.gitignore	Initial commit
01-nametrend.Rmd	Add files via upload
README.md	Initial commit

```
https://github.com/stat220/01-R-markdo
```

Use Git or checkout with SVN using the web URL.

README.md

01-R-markdown_example

05:00

Metadata and output types

YAML

Basic recipe:

```
---
key: value
---
```

Example:

```
---
title: My title
output:
  github_document
    toc: true
    theme: flatly
---
```

Output types

- html_document (can’t view in GitHub repo)
- pdf_document (need MikTex or MacTex installed)
- github_document (creates a .md Markdown doc, viewable on GitHub)
- ioslides_presentation, beamer_presentation

```
---
title: "Baby Name Trends"
output: github_document
---
```

```
---
title: "Baby Name Trends"
output: github_document
param:
  attribute: value
---
```

Parameters

```
---
title: "Baby Name Trends"
output: html_document
params:
  name:
  from:
  to:
---
```

To D0:

- Change the parameters and output types!!
- Try 'Knit with Parameters'

06:00

Text

Use markdown to format the text

10 minute tutorial

Text

- Simple rules for
    - section headers (`#`,`##`,etc)
    - lists (need ~2 tabs to create sublists)
    - formatting (bold `**`, italics `*`)
    - tables
    - R syntax (use backward tick `` ` ``)
    - web links `[linked text](url)`
    - latex math equations $\beta_1 + \beta_2$
    - in HTML docs, you can use HTML commands (in pdf, latex commands)

For further help, look at R Markdown Cheatsheet

Code chunks

Code goes in **chunks**, defined by three backticks

~~~
```{r}

filtered_names <- babynames %>% filter(name=="Amiee", year < max(year), year > min(year))

ggplot(data=filtered_names, aes(x=year, y=prop)) +
  geom_line(aes(colour=sex)) +
  xlab('Year') +
  ylab('Prop. of Babies Named Aimee')

```
~~~

Adding/running chunks

Let's

1. Add chunks with button or:

   Command (or Cmd) `⌘` + Option (or Alt) `⌥` + `i` (Mac)

   Ctrl + Alt + `i` (Windows/Linux)

2. Run chunks by:

   Run current chunk button (interactive)

   Knit button / run all chunks

Inline code

How many babies were born with name 'Aimee'?

```
`r sum(filtered_names$n)`
```

There are a total of 53228 babies.

In what year were there highest number of babies born with the name `Aimee`?

```
`r filtered_names$year[which.max(filtered_names$prop)]`
```

`Amiee` name was the most popular in 1973.

Chunk options

~~~
```{r}
glimpse(filtered_names)
```
~~~

```
dplyr::glimpse(filtered_names)
## Rows: 148
## Columns: 5
## $ year <dbl> 1881, 1882, 1883, 1884, 1885, 1886, 18
## $ sex  <chr> "F", "F", "F", "F", "F", "F", "F", "F",
## $ name <chr> "Aimee", "Aimee", "Aimee", "Aimee", "A
## $ n    <int> 11, 13, 11, 15, 17, 17, 18, 12, 16, 18,
## $ prop <dbl> 0.00011127, 0.00011236, 0.00009162, 0.
```

echo

~~~
```{r echo=FALSE}
glimpse(filtered_names)
```
~~~

```
## Rows: 148
## Columns: 5
## $ year <dbl> 1881, 1882, 1883, 1884, 1885, 1886, 1887, 1888, 1889, 1890, 1891,…
## $ sex  <chr> "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", …
## $ name <chr> "Aimee", "Aimee", "Aimee", "Aimee", "Aimee", "Aimee", "Aimee", "A…
## $ n    <int> 11, 13, 11, 15, 17, 17, 18, 12, 16, 18, 14, 15, 17, 13, 13, 23, 1…
## $ prop <dbl> 0.00011127, 0.00011236, 0.00009162, 0.00010902, 0.00011976, 0.000…
```

eval

~~~
```{r eval=FALSE}
glimpse(filtered_names)
```
~~~

```
glimpse(filtered_names)
```

include

~~~
```{r include=FALSE}
glimpse(filtered_names)
```
~~~

results

~~~
```{r echo=TRUE, results='hide'}
glimpse(filtered_names)
```
~~~

```
glimpse(filtered_names)
```

Chunk option take-aways

- Place between curly braces
  `{r option=value}`
- Multiple options separated by commas
  `{r option1=value, option2=value}`
- Careful! The `r` part is the **code engine** (other engines possible)

Chunk labels

~~~
```{r peek, echo=FALSE, results='hide'}
glimpse(filtered_names)
```
~~~

- Place between curly braces
  `{r label}`
- Separate options with commas
  `{r label, option1=value}`
- Careful! Don't duplicate labels

~~~
```{r peek}
head(filtered_names)
```
~~~

```
Error in parse_block(g[-1], g[1], params.src) :
  duplicate label 'peek'
Calls: <Anonymous> ... process_file -> split_file -> lapply -> FUN -> parse_block
Execution halted
```

The setup chunk

~~~
```{r setup, include=FALSE}
knitr::opts_chunk$set(
  collapse = TRUE,
  comment = "#>",
  out.width = "100%"
)
```
~~~

- A special chunk label: `setup`
- Typically the first
- All following chunks will use these options (i.e., sets global chunk options)
- **Tip**: set `include=FALSE`
- You can (and should) use individual chunk options too

Exercise

Customize this report

1. Label the code chunk with your plot in it.

2. Add your setup chunk.

3. Add `fig.path = "figs/"` as a knitr code chunk option for a single plot. What happened? What happens if you don't include the backslash?

4. Knit and behold

5. Add it to a global setup chunk instead

05:00

Acknowledgments

Parts of these slides were adapted from previous works of Adam Loy and Katie St. Clair.