

R and R Markdown Basics

Spring 23

March 28 2023

Reproducible data science

What does it mean for a data analysis to be "reproducible"?

Short-term goals

- 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?)

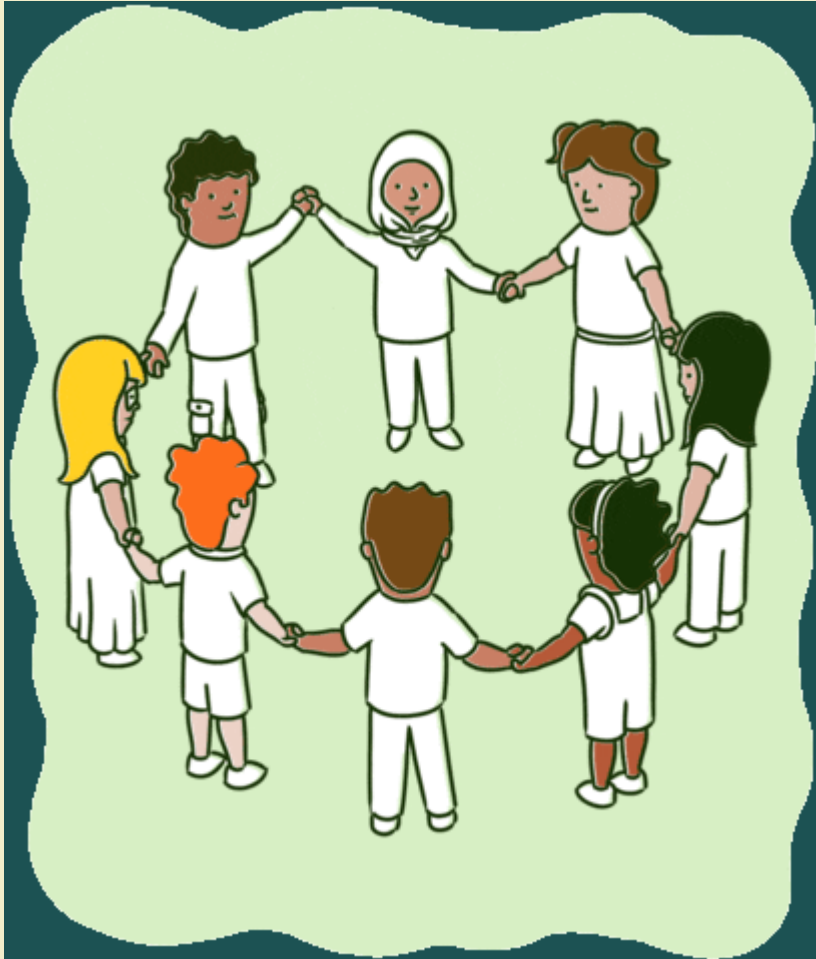
Long-term goals

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

Toolkit for reproducibility

- **Scriptability** → R
- **Literate programming** (code, narrative, output in one place) → R Markdown
- **Version control** → Git / GitHub

Group Activity 1



- Let's go over to maize server/ local Rstudio to connect it with the class Github repository
- Clone hw0-yourusername repository to your course folder as an R project using version control methods
- Make changes to hw and commit those changes
- Tip: push the changes regularly back to Github

Tour: R Markdown

```
1 ---
2 title: "Class Activity 1"
3 author: "Your name here"
4 date: "`r format(Sys.Date(), ' %B %d %Y')`"
5 output:
6   html_document:
7     df_print: paged
8   pdf_document: default
9 ---
10
11 ```{r, include=FALSE}
12 knitr::opts_chunk$set(echo = TRUE,
13                       size = "small",
14                       collapse = TRUE,
15                       comment = NA,
16                       warning = FALSE,
17                       message = FALSE,
18                       error = TRUE) # document will compile even in case of error
19 ```
20
21 { The `R` package `babynames` provides data about the popularity of individual baby names from the US Social Security
22   Administration. Data includes all names used at least 5 times in a year beginning in 1880.
23
24 ```{r}
25 #install.packages("babynames")
26 library(babynames)
```

YAML metadata

Setup Chunk

Code Chunk

Text

Metadata and output types

YAML (yet another markup language)

- data serialization language that is often used for writing configuration files.

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, MacTeX, or TinyTeX installed)
- `github_document` (creates a .md Markdown doc, viewable on GitHub)

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

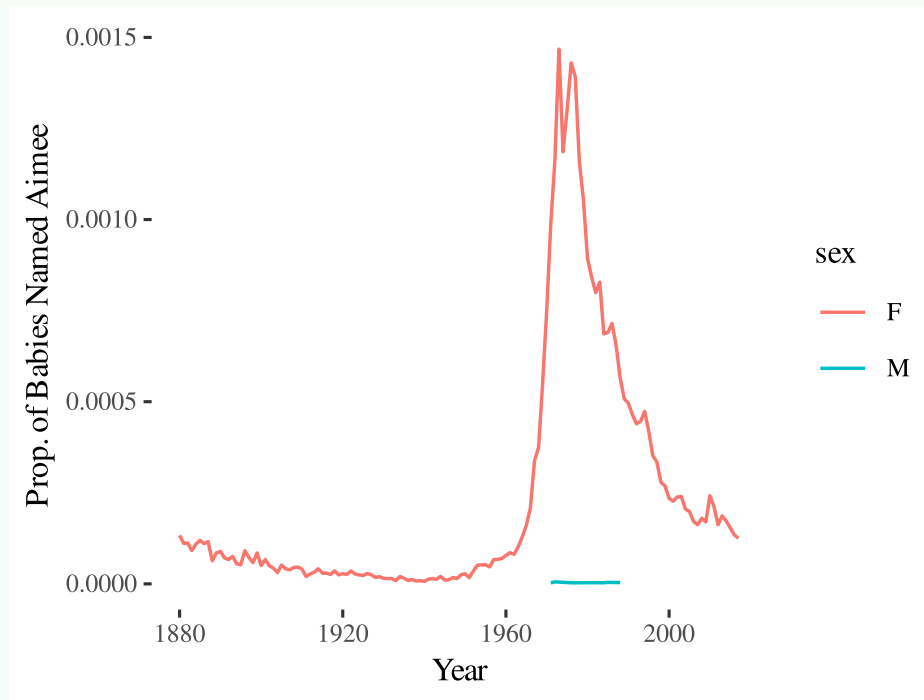
```
---  
title: "Baby Name Trends"  
output: github_document  
params:  
  attribute: value  
---
```

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 $\beta_1 + \beta_2$

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

```
```{r}
filtered_names <- babynames %>% filter(name=="Aimee", year < max(year), year > min(year)) # filter the rows
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

Add chunks with button or:

- Command (or Cmd) ⌘ + Option (or Alt) ⌥ + i (Mac)
- Ctrl + Alt + i (Windows/Linux)

Run chunks by:

- Run current chunk button (interactive)
- Knit button / run all chunks

Inline code

How many babies were born with name 'Aimee'?

```
`r filtered_names %>% summarise(total = sum(n))`
```

There are a total of 53476 babies.

In what year were there highest proportion of babies born with the name **Aimee**?

```
`r filtered_names %>% filter(prop == max(prop)) %>% pull(year)`
```

Aimee name was the most popular in 1973.

Chunk options: echo

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

```
Rows: 150  
Columns: 5  
$ year <dbl> 1880, 1881, 1882, 1883, 1884, 1885, 1886, 1887, 1888, 1889, 1890, ...  
$ sex <chr> "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", ...  
$ name <chr> "Aimee", "Aimee", "Aimee", "Aimee", "Aimee", "Aimee", "Aimee", "Aimee", "A...  
$ n <int> 13, 11, 13, 11, 15, 17, 17, 18, 12, 16, 18, 14, 15, 17, 13, 13, 2...  
$ prop <dbl> 0.00013319, 0.00011127, 0.00011236, 0.00009162, 0.00010902, 0.000...
```

Chunk options: eval

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

```
> glimpse(filtered_names)
```

Chunk options: eval

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

```
> glimpse(filtered_names)
```

Chunk options: include

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

Chunk options: eval

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

```
> glimpse(filtered_names)
```

Chunk options: include

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

Chunk options: results

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

```
> glimpse(filtered_names)
```

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}`

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

# Math in R

```
> 10^2
[1] 100
```

```
> 3 ^ 7
[1] 2187
```

```
> 6/9
[1] 0.6666667
```

```
> 9-43
[1] -34
```

- Rules for order of operations are followed
- Spaces between numbers and characters are ignored

```
> 4^3-2* 7+9 /2
[1] 54.5
```

The equation above is computed as

$$4^3 - (2 \cdot 7) + \frac{9}{2}$$

# Variables

*Variables are used to store data, figures, model output, etc.*

Assign just one value:

```
> x <- 5
> x
[1] 5
```

Concatenate a string of numbers

```
> b <- c(5, 12, 2, 100, 8)
> b
[1] 5 12 2 100 8
```

Assign a **vector** of values:

```
> a <- 3:10
> a
[1] 3 4 5 6 7 8 9 10
```

Concatenate a string of characters

```
> names <- c("Amy", "Dee", "Lux")
> names
[1] "Amy" "Dee" "Lux"
```

## A few things to remember

- Do not use special characters such as `$` or `%`. Common symbols that are used in variable names include `.` or `_`.
- Remember that `R` is case sensitive.
- To assign values to objects, we use the assignment operator `<-`. Recommend to use `<-` to assign values to objects and `=` within functions.
- The `#` symbol is used for commenting and demarcation. Any code following `#` will not be executed.

# Data frames (aka "tibbles" in tidyverse)

**Vectors** vs. **data frames**: a data frame is a collection (or array or table) of vectors

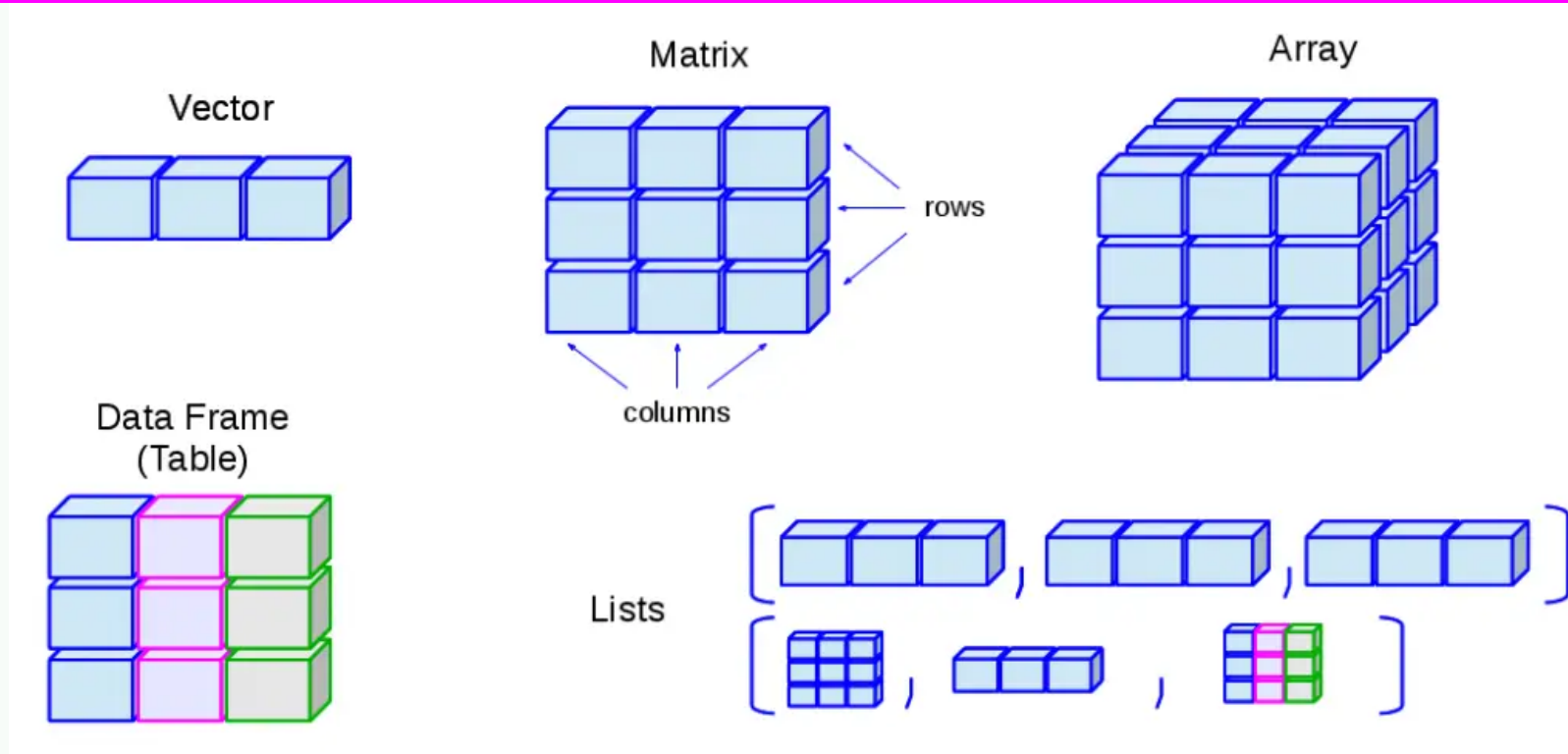
```
> df <- tibble(
+ IDs=1:3,
+ gender=c("Male", "Female", "Male"),
+ age=c(28, 36, 23),
+ trt = c("control", "treatment", "treatment"),
+ Diabetes = c(FALSE, TRUE, TRUE)
+)
> df
A tibble: 3 × 5
 IDs gender age trt Diabetes
 <int> <chr> <dbl> <chr> <lgl>
1 1 Male 28 control FALSE
2 2 Female 36 treatment TRUE
3 3 Male 23 treatment TRUE
```

- Allows different columns to be of different data types (i.e. numeric vs. text)
- Both numeric and text can be stored within a column (stored together as *text*).

## R Objects

Vectors and data frames are examples of *objects* in R.

- There are other types of R objects to store data, such as matrices, arrays, lists.



## Variable (column) types

type	description
integer	integer-valued numbers
numeric	numbers that are decimals
factor	categorical variables stored with levels (groups)
character	text, "strings"
logical	boolean (TRUE, FALSE)

- View the **structure** of our data frame to see what the variable types are:

```
> str(df)
tibble [3 × 5] (S3: tbl_df/tbl/data.frame)
 $ IDs : int [1:3] 1 2 3
 $ gender : chr [1:3] "Male" "Female" "Male"
 $ age : num [1:3] 28 36 23
 $ trt : chr [1:3] "control" "treatment" "treatment"
 $ Diabetes: logi [1:3] FALSE TRUE TRUE
```

# Data frame cells, rows, or columns

Show whole data frame

```
> df
A tibble: 3 × 5
 IDs gender age trt Diabetes
<int> <chr> <dbl> <chr> <lgl>
1 1 Male 28 control FALSE
2 2 Female 36 treatment TRUE
3 3 Male 23 treatment TRUE
```

Specific cell: `DataSetName[row#, column#]`

```
> # Second row, Third column
> df[2, 3]
A tibble: 1 × 1
 age
<dbl>
1 36
```

Entire col: `DataSetName[, column#]`

```
> # Third column
> df[, 3]
A tibble: 3 × 1
 age
<dbl>
1 28
2 36
3 23
```

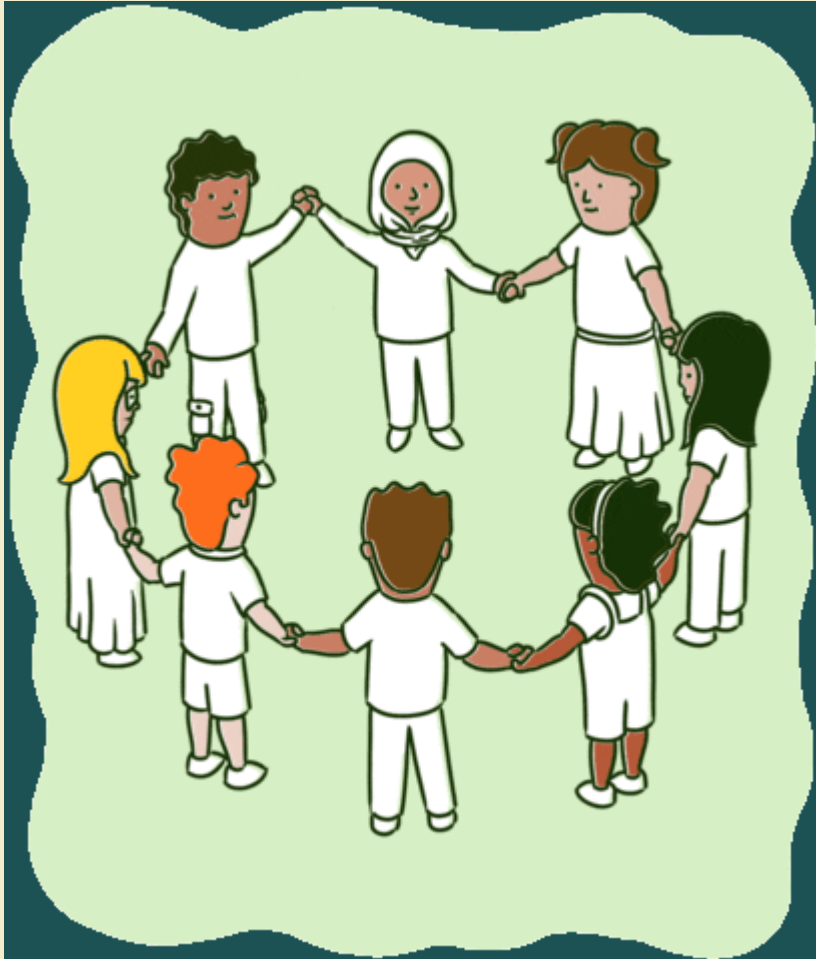
Entire row: `DataSetName[row#, ]`

```
> # Second row
> df[2,]
A tibble: 1 × 5
 IDs gender age trt Diabetes
<int> <chr> <dbl> <chr> <lgl>
1 2 Female 36 treatment TRUE
```



## Group Activity 2

10:00



Go to `class_activity_2.Rmd` in **moodle**

1. Read through the activity answering any questions asked
2. Submit to moodle when done.