I have always been inspired by those who can capture the landscape with a minimum of brushstrokes
HELLO
my name is
Phil Bowsher
MY BACKGROUND

- Shiny
- CS
- Pharma
- Audience
- twitter:@rinpharma
- github:philbowsher
- Speed is the name of the game – fastest way to get you started
- Red is an Action Item for you
GOALS FOR TODAY

- Getting to Know Rstudio
- Getting to Know Python
- Importing data
- Data Viz
- Data Wrangling
- Packages
- Reporting
- R Functions and Creating Packages
THINGS YOU SHOULD KNOW

• There are no stupid questions

• Because we have limited time, please write down questions as they occur to you, and then ask them when the instructor pauses for questions

• You will not learn all of R today

• Don't be afraid to experiment and try things out; you won't blow up anything important if you make a mistake.
Workshop Communication

- Zoom Chat
- Polls & Breakout Rooms
- https://calendly.com/rstudio-phil-bowsher/30min?month=2020-10
Your Turn

Form groups of 2-4 people. Introduce yourself to your group members. Tell them:

1. Who you are
2. What you do with data
3. How long you have been using R
Quick Survey

http://rstd.io/phil-me-out
SETUP IN RTT

- Setup
  - https://github.com/sol-eng/classroom-getting-started
  - http://rstd.io/class
Setup in RTT

- R
- Packages
- IDE
- Projects
- Sessions
- Git/Github
- RSC
- Shiny
Setup & Quick Intro
(Panes & Buttons)
# Built-in datasets

- `data()`
- `data(ToothGrowth)`
- `?ToothGrowth`
- `ToothGrowth`
- `View(ToothGrowth)`
- `summary(ToothGrowth)`
- `plot(ToothGrowth)`
IDE – Let’s explore ... 

- `# getwd()`
- `library(tidyverse)`
- `# let us explore the data set a bit`
- `names(ToothGrowth) # names of the variables`
- `dim(ToothGrowth) # dimension (number of rows and columns)`
- `str(ToothGrowth) # structure of the data set`
- `class(ToothGrowth)`
- `head(ToothGrowth, n = 5)`
- `tail(ToothGrowth, n = 5)`
- `ToothGrowth %>% write_csv('ToothGrowth.csv')`
- `ToothGrowth2 <- read_csv("ToothGrowth.csv")`
Welcome to RStudio Cloud alpha

Do, share, teach and learn data science with R.

Get Started

If you already have an RStudio shinyapps.io account, you can log in using your existing credentials.

THE MISSION

We created RStudio Cloud to make it easy for professionals, hobbyists, trainers, teachers and students to do, share, teach and learn data science using R.

To learn more about RStudio’s other products visit rstudio.com

RStudio.cLOUD

OVERVIEW
After the workshop, go here:

https://rstudio.cloud/spaces/89287/join?access_code=V%2FvqOUv2%2FMCQF0jGlRPJnLAeyN41fegtHS0mpPB
By the Way, Books...

- https://mastering-shiny.org/
- https://bookdown.org/yihui/rmarkdown/
- http://r4ds.had.co.nz & https://rstudio.cloud/
- https://r-graphics.org/
- http://www-bcf.usc.edu/~gareth/ISL/
- http://appliedpredictivemodeling.com/
- https://bookdown.org/yihui/rmarkdown/
- https://www.tidytextmining.com/
- https://adv-r.hadley.nz/
- https://plotly-r.com/
- https://therinspark.com/
- https://www.tidymodels.org/books/
BREAK TIME

- 5-10 Min Break
YOUR TURN

• Form groups of 2-4 people
• Visual Analytics
• Have you used the Tidyverse?
• What data do you import?
• How much time do you spend cleaning data?
Python

- Python Slides are here:

  https://colorado.rstudio.com/rsc/WorkshopRDeepLearningSci/workshopTensorflow.html#11
Import Data with readr
First things First... Data

Read data in

Read in data with `readr`, `haven`, `readxl`

**readr**
- `read_csv()`, `read_tsv()`, `read_delim()`

**haven**
- `read_sas()`, `read_spss()`, `read_stata()`, `read_dta()`

**readxl**
- `read_xls()`, `read_xlsx()`, `read_excel()`
Importing Data
readr

Simple, consistent functions for working with strings.

```r
# install.packages("tidyverse")
library(tidyverse)
```
Compared to read.table and its derivatives, readr functions are:

1. ~ 10 times faster
2. Return tibbles
3. Have more intuitive defaults. No row names, no strings as factors.
## readr functions

<table>
<thead>
<tr>
<th>function</th>
<th>reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>read_csv()</td>
<td>Comma separated values</td>
</tr>
<tr>
<td>read_csv2()</td>
<td>Semi-comma separated values</td>
</tr>
<tr>
<td>read_delim()</td>
<td>General delimited files</td>
</tr>
<tr>
<td>read_fwf()</td>
<td>Fixed width files</td>
</tr>
<tr>
<td>read_log()</td>
<td>Apache log files</td>
</tr>
<tr>
<td>read_table()</td>
<td>Space separated</td>
</tr>
<tr>
<td>read_tsv()</td>
<td>Tab delimited values</td>
</tr>
</tbody>
</table>
readr functions share a common syntax

```r
df <- read_csv("path/to/file.csv", ...)
```

object to save output into

path from working directory to file
### Import Text Data

#### File/URL:
```
~/Abbott Workshop/The-Little-R-Workshop-A-Primer-master/Examples/data/landdata-states.csv
```

#### Data Preview:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AK</td>
<td>West</td>
<td>2010.25</td>
<td>224952</td>
<td>160599</td>
<td>64362</td>
<td>28.6</td>
<td>1.481</td>
<td>1.552</td>
<td>2010</td>
</tr>
<tr>
<td>AK</td>
<td>West</td>
<td>2010.50</td>
<td>225511</td>
<td>160132</td>
<td>65239</td>
<td>28.9</td>
<td>1.454</td>
<td>1.576</td>
<td>2010</td>
</tr>
<tr>
<td>AK</td>
<td>West</td>
<td>2009.75</td>
<td>225820</td>
<td>163791</td>
<td>62029</td>
<td>27.5</td>
<td>1.464</td>
<td>1.464</td>
<td>2009</td>
</tr>
<tr>
<td>AK</td>
<td>West</td>
<td>2010.00</td>
<td>224894</td>
<td>161787</td>
<td>63207</td>
<td>28.1</td>
<td>1.481</td>
<td>1.524</td>
<td>2009</td>
</tr>
</tbody>
</table>

#### Import Options:
- **Name:** landdata_states
- **First Row as Names:** Yes
- **Delimiter:** Comma
- **Escape:** None
- **Trim Spaces:** Yes
- **Quotes:** Default
- **Comment:** Default
- **Locale:** Configure
- **NA:** Default

#### Code Preview:
```r
library(readr)
landdata_states <- read_csv("Abbott Workshop/The-Little-R-Workshop-A-Primer-master/Examples/data/landdata-states.csv")
View(landdata_states)
```
Let’s Chat about Notebooks…

Leonardo da Vinci…Page from the Codex Atlanticus shows notes and images about water wheels and Archimedean Screws
Notebooks

• Number 3: Notebooks are for doing science
• Number 2: R Notebooks have great features
• Number 1: R Notebooks make it easy to create and share reports

https://rviews.rstudio.com/2017/03/15/why-i-love-r-notebooks/
http://r4ds.had.co.nz/r-markdown-workflow.html
Notebooks

Combine in a single document:

- Narrative
- Code
- Output

Then Render to HTML
Setup

The setup chunk is always run once before anything else and is a great place to load packages.
Data Visualization with ggplot2
ggplot2

A package that visualizes data.

ggplot2 implements the grammar of graphics, a system for building visualizations that is built around cases and variables.
R Graphics:

Four Main Graphical Systems in R:
• R’s Base Graphics
• Grid Graphics System
• The lattice Package
• The ggplot2 Package – Created by Hadley Wickham

Why ggplot2?
• Consistent underlying - Grammar of Graphics (Wilkinson, 2005)
• Very flexible
• Mature and complete graphics system
• Many users, active mailing list
ggplot2

`ggplot2` is a powerful and flexible plotting system for R, developed by Hadley Wickham. It provides a grammar of graphics, allowing for the creation of complex statistical plots with ease. The `ggplot2` package is widely used in data visualization and statistical analysis, offering a wide range of features for creating high-quality, publication-ready graphs.

The `ggplot2` package is designed to be modular, allowing users to build plots one element at a time. It supports various types of plots, including line plots, bar charts, scatter plots, and more. Additionally, `ggplot2` provides extensive customization options, enabling users to tailor their plots to specific needs and preferences.

In summary, `ggplot2` is an essential tool for anyone working with data visualization in R, offering a robust framework for creating visually appealing and informative graphs.
ggplot2

![ggplot2 Diagram](image-url)
ggplot2

- Scatter plot of `Sepal.Length` vs `Sepal.Width`
- Heatmaps of species distribution with color-coded levels for each species:
  - `setosa`
  - `versicolor`
  - `virginica`

Species levels:
- `0.5`
- `1.0`

CC by RStudio
ggplot(mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
A ggplot2 template

Make any plot by filling in the parameters of this template

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
    stat = <STAT>) +
  <FACET_FUNCTION>
```

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut), stat = "count")
```
"Help" pages

To open the documentation for a function, type

```
?geom_histogram
```

Function name (no parentheses)
global vs. local
Mappings (and data) that appear in `ggplot()` will apply globally to every layer

ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
geom_point() +
geom_smooth()
Mappings (and data) that appear in a geom_function will add to or override the global mappings for that layer only.

```r
library(ggplot2)

ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
geom_point(mapping = aes(color = drv)) +
geom_smooth()
```
```r
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) + geom_point(mapping = aes(color = drv)) + geom_smooth(data = filter(mpg, drv == "f"))
data can also be set locally or globally
```
A ggplot2 template

Make any plot by filling in the parameters of this template

Complete the template below to build a graph.

```r
ggplot (data = <DATA>) +
  <GEOM_FUNCTION> (mapping = aes(<MAPPINGS>),
  stat = <STAT>, position = <POSITION>) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION> +
  <SCALE_FUNCTION> +
  <THEME_FUNCTION>
```

Not required, sensible defaults supplied
Code:

ggplot(data = bankData) +
aes(x = job, y = age, fill = marital) +
geom_boxplot() +
theme_minimal()

Export:
- PNG
- PowerPoint
EXERCISE 7MINS:

• Import some data and build some visualizations using esquisse…Take your data import code and visualization code and insert it into a notebook. Pick any dataset below to work on.

• landdata-states.csv
• starbucks.csv
• adae.csv
• dm.csv
• bank-full.csv
• ad_treatment.xlsx
• dmae.sas7bdat
htmlwidgets
http://www.wthr.com/article/watch-video-shows-tornado-destroying-kokomo-starbucks
Demo Example

- 1_htmlwidgets_tornadoes.R
- Compare R script vs Notebook – what are the differences?
Plotly

Tools for making interactive plots. plot.ly/ggplot2/
htmlwidgets for R:
• R bindings to JavaScript libraries
• Used to create interactive visualizations
• A line or two of R code is all it takes to produce an example

Use htmlwidgets in:
• RStudio viewer pane
• R Markdown files
• Shiny Apps

www.htmlwidgets.org
htmlwidgets gallery

http://gallery.htmlwidgets.org/
EXERCISE 10 MINs:

- 02-Visualize-Exercises.Rmd Beginner
- Run through the chunks
- 2_r4ds_ggplot2_tidyverse.Rmd More Advanced
Break Time

- Fun Video
Masters of the Tidyverse

Art by Dan Mumford
I said Tibbles, not Tribbles.

...Darn Quadrotriticale!
Tidyverse

A collection of R packages that share common philosophies and are designed to work together.

install.packages("tidyverse")
library(tidyverse)
Data Types

- R has a wide variety of data types...
- Vectors
- Lists
- Matrix
- Factors
- Data frame
- Tibble
- Is ToothGrowth a Tibble? Hint: class(ToothGrowth)
Toy data

```r
storms <- tribble(
  ~storm, ~wind, ~pressure, ~date,
  "Alberto", 110,  1007, "2000-08-12",
  "Alex",   45,   1009, "1998-07-30",
  "Allison", 65,   1005, "1995-06-04",
  "Ana",    40,   1013, "1997-07-01",
  "Arlene",  50,   1010, "1999-06-13",
  "Arthur",  45,   1010, "1996-06-21"
)
```

```
<table>
<thead>
<tr>
<th>storm</th>
<th>wind</th>
<th>pressure</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alberto</td>
<td>110</td>
<td>1007</td>
<td>2000-08-12</td>
</tr>
<tr>
<td>Alex</td>
<td>45</td>
<td>1009</td>
<td>1998-07-30</td>
</tr>
<tr>
<td>Allison</td>
<td>65</td>
<td>1005</td>
<td>1995-06-04</td>
</tr>
<tr>
<td>Ana</td>
<td>40</td>
<td>1013</td>
<td>1997-07-01</td>
</tr>
<tr>
<td>Arlene</td>
<td>50</td>
<td>1010</td>
<td>1999-06-13</td>
</tr>
<tr>
<td>Arthur</td>
<td>45</td>
<td>1010</td>
<td>1996-06-21</td>
</tr>
</tbody>
</table>
```
**Tibbles – Quick Intro**

- `as_tibble()`
- `as_tibble(ToothGrowth)`
- This will work for reasonable inputs that are already data.frames, lists, matrices, or tables.
- There are two main differences in the usage of a tibble vs. a classic data.frame: printing and subsetting
- Tibbles show only the first 10 rows
- Each column reports its type, a nice feature borrowed from `str()`
- `package?tibble`
Tibbles – What to Know

- `tibble()` does much less than `data.frame()`:
  - A. it never changes the type of the inputs (e.g. it never converts strings to factors!)
  - B. it never changes the names of variables, and it never creates row names

- `data.frame()` vs. modern `data_frame()`:
  - Base R has a burning desire to turn character information into factor...via `read.table()` & `data.frame()` and other functions are also eager
  - To shut this down, use `stringsAsFactors = FALSE` in `read.table()` and `data.frame()` or – even better – use the tidyverse!

- `readr::read_csv()`, `readr::read_tsv()`, etc. For data frame creation, use `tibble::tibble()`
install.packages("tidyverse")

does the equivalent of

install.packages("ggplot2")
install.packages("dplyr")
install.packages("tidyr")
install.packages("readr")
install.packages("purrr")
install.packages("tibble")
install.packages("hms")
install.packages("stringr")
install.packages("lubridate")
install.packages("forcats")
install.packages("DBI")
install.packages("haven")
install.packages("httr")
install.packages("jsonlite")
install.packages("readxl")
install.packages("rvest")
install.packages("xml2")
install.packages("modelr")
install.packages("broom")
install.packages("tidyverse") does the equivalent of

install.packages("ggplot2")
install.packages("dplyr")
install.packages("tidyr")
install.packages("readr")
install.packages("purrr")
install.packages("tibble")
install.packages("hms")
install.packages("stringr")
install.packages("lubridate")
install.packages("forcats")
install.packages("DBI")
install.packages("haven")
install.packages("httr")
install.packages("jsonlite")
install.packages("readxl")
install.packages("rvest")
install.packages("xml2")
install.packages("modelr")
install.packages("broom")
tidyverse Core Principles

- Built around data - usually as a *data.frame* or *tibble*
- Built around tidy data
  - Each variable in its own column
  - Each observation or case in its own row
  - Each type of observational units forms a table
babynames

Names of male and female babies born in the US from 1880 to 2008. 1.8M rows.

```r
# install.packages("babynames")
library(babynames)
```
View(babynames)

<table>
<thead>
<tr>
<th>year</th>
<th>sex</th>
<th>name</th>
<th>n</th>
<th>prop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>F</td>
<td>Mary</td>
<td>7065</td>
<td>0.0723835869</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>Anna</td>
<td>2604</td>
<td>0.0266789611</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>Emma</td>
<td>2003</td>
<td>0.0205214897</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>Elizabeth</td>
<td>1939</td>
<td>0.0198657856</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>Minnie</td>
<td>1746</td>
<td>0.0178884278</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>Margaret</td>
<td>1578</td>
<td>0.0161672045</td>
</tr>
<tr>
<td>7</td>
<td>F</td>
<td>Ida</td>
<td>1472</td>
<td>0.0150811946</td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td>Alice</td>
<td>1414</td>
<td>0.0144869628</td>
</tr>
<tr>
<td>9</td>
<td>F</td>
<td>Bertha</td>
<td>1320</td>
<td>0.0135238973</td>
</tr>
<tr>
<td>10</td>
<td>F</td>
<td>Sarah</td>
<td>1288</td>
<td>0.0131960453</td>
</tr>
<tr>
<td>11</td>
<td>F</td>
<td>Annie</td>
<td>1258</td>
<td>0.0128886840</td>
</tr>
</tbody>
</table>
We call this data **Tidy**.

Data are organized in **rows** and **columns**.

Each **variable** has its own column.

Each **observation** or record has its own row.

Every cell has only one value in it.
## Untidy data

```r
untidy_df
```

```r
## A tibble: 5 x 7
##  age_group     male_2016 female_2016 male_2017 female_2017 male_2018
##       <chr>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
## 1    1 < 18      22000    20000     22000     20000     22000
## 2   2 18-30      36000    35000     36000     35000     36000
## 3   3 31-50      50000    40000     50000     40000     50000
## 4   4 51-60      62000    60000     62000     60000     62000
## 5   5 > 60       75000    72000     75000     72000     75000
```

```r
## ... with 1 more variable: female_2018 <dbl>
```
## Tidy data

```r
tidy_df

## # A tibble: 30 x 4
## #  age_group gender year income
## #  <chr>    <chr> <chr>   < dbl>
##   1 < 18    male 2016  22000
##   2 18-30   male 2016  36000
##   3 31-50   male 2016  50000
##   4 51-60   male 2016  62000
##   5 > 60    male 2016  75000
##   6 < 18    female 2016 20000
##   7 18-30   female 2016 35000
##   8 31-50   female 2016 40000
##   9 51-60   female 2016 60000
##  10 > 60    female 2016 72000
##   # ... with 20 more rows
```
BREAK TIME

- Fun Video
Transforming Data & Data Visualization

Art by Lou Pimentel
A package that transforms data.
dplyr implements a *grammar* for transforming tabular data.

Data transformation toolbox
Sparklyr
Makes it easy to use R with Spark

Spark with AWS EMR
Spark and the data lake
**dplyr**

**6 Main verbs**
- `filter()`
- `arrange()`
- `select()`
- `mutate()`
- `group_by()`
- `summarise()`

**Simple use**
- `pull()`
- `n()`/`count()`
- `glimpse()`

**Advanced iterations**
- `summarize_at()`
- `mutate_at()`

**More info**
- `dplyr.tidyverse.org`
- *R for Data Science*
babynames %>%
group_by(year, sex) %>%
summarise(total = sum(n)) %>%
ggplot(aes(x = year, y = total, color = sex)) +
  geom_line()
%>%% IS CALLED THE PIPE OPERATOR

• Connects the output of one operation to the input of the next.
• Allows us to see the flow of data from left to right

```r
employees %>%
  filter(Department == "Engineering") %>%
  count()

# A tibble: 1 x 1
   n
<int> 71
```
The %>% == and then

Rather than multiple assignment or nesting functions

```r
did_something <- do_something(data)

did_anotherThing <- do_another_thing(did_something)

final_thing <- do_last_thing(did_another_thing)

final_thing <- do_last_thing(
  do_another_thing(
    do_something(
      data
    )
  )
)

final_thing <- data %>%
  do_something() %>%
  do_another_thing() %>%
  do_last_thing()
```
Phil <- filter(babynames, name == "Phil", sex == "M")
summarise(Phil, min = min(prop), mean = mean(prop),
          max = max(prop))

filter(babynames, name == "Phil", sex == "M") %>%
summarise(min = min(prop), mean = mean(prop),
          max = max(prop))
Phil <- filter(babynames, name == "Phil", sex == "M")
summarise(Phil, min = min(prop), mean = mean(prop),
          max = max(prop))

babynames %>%
  filter(name == "Phil", sex == "M") %>%
  summarise(min = min(prop), mean = mean(prop),
            max = max(prop))
Shortcut to type %>%

- **Cmd** + **Shift** + **M**  (Mac)
- **Ctrl** + **Shift** + **M**    (Windows)
Your Turn – Pick One of Interest: 10min

- 03-Transform-Exercises.Rmd Beginner
- 1_dplyr_tidyr_r4ds_tidyverse.Rmd Advanced
- In folder RMD_Clinical_Tidyverse, A gentle guide to Tidy statistics in R.rmd Clinical
Tidy Data with tidyr
tidyr

A package that reshapess the layout of tabular data.
tidyr

The goal of tidyr is to help you create tidy data. Tidy data is data where:

- Each variable is in a column.
- Each observation is a row.
- Each value is a cell.

Make Taller and Make Wider

- `gather()` - takes multiple columns, and gathers them into key-value pairs: it makes “wide” data longer.
- `spread()` - takes two columns (key & value), and spreads into multiple columns: it makes “long” data wider.

Separate and unite columns

- `separate()` - Separate one column into multiple columns.
- `unite()` - Unite multiple columns into one.
Your Turn: 3min

- 1_dplyr_tidy_r4ds_tidyverse.Rmd
- At Bottom
Break Time

- Fun Video
Shiny Slides are Here

https://colorado.rstudio.com/rsc/content/3437/#67
R Markdown
R Markdown

An authoring format for Data Science.

```
---
title: "R Notebook"
output: html_notebook
---

Text written in **markdown**

```{r}
# code written in R
(x <- rnorm(7))
```

```
[1] -1.2  1.0 -0.5  0.9 -0.6 -1.1 -1.5
```
R Markdown
The Definitive Guide

bookdown.org/yihui/rmarkdown/

ONLINE, FREE
rmardown

- Starts as a notebook style interface
  - Mixes code with prose
- Knits to dozens of different formats
  - HTML
  - PDF
  - Handouts
  - Books
  - Reports
  - dashboards
  - Interactive shiny apps
  - Articles
  - Websites
Your Turn: 3mins

• Go to 3_Report.
• Open 01-RMarkdown-Exercises.Rmd.
• Read through the file and do everything it tells you to do.
KNITR IS MULTI-LINGUAL!

- SAS
- PYTHON
- MORE
Your Turn

- demo-notebook.Rmd in python folder
- Which section is changing the language engine via knitr?
Parameters

A list of values that you can call in R code chunks

```r
params list
params: 
  filename: "data.csv"
  symbol: "GOOG"
```
Your Turn: 5mins

• Part 1 - Easy
• 1_RMD_Stocks.Rmd
• Pick a new stock and generate a new report
• Now Make it a PDF

• Part 2 - Harder
• Go to 05-Report-Exercise.Rmd
• See if you can make it parameterized for your name
R Markdown

Render

Function
rmarkdown::render

Render at the command line with YAML options

```r
> render("doc.Rmd")
```

Render at the command line, override output format.

```r
> render("doc.Rmd", "html_document")
```

Render at the command line to multiple formats.

```r
> render("doc.Rmd", c("html_document", "pdf_document"))
```
rmarkdown::render

Render at the command line with YAML options

> render("doc.Rmd")

Render at the command line, set parameters.

> render("doc.Rmd", params = list(  
  filename = "other_data.csv",  
  symbol = "AAPL")

Your Turn: 2mins

• 3_RMD_stock_Flex_CSS
• Render the report programatically using the render function.
• Now do it manually using the “Knit with Parameters” button
What about many reports?

Say Hi to purrr
Demo Example

- 6_RMD_Report_Versions
- Don't forget to set the WD
- Review the airplane-report.Rmd
- Notice how it is connecting to a DB?
- build_airplane_report, run this function as well as the list above it
- Then run the purr command at the bottom
- See how the reports generate dynamically?
- How would you automate this? Ask your neighbor
What if I Start to Have a Collection of RMDs or If I Want a Website?
Other R Markdown Output Types

- Blogdown
- RMD Websites - Example
- Bookdown
- Presentations
- Package Documentation
Your Turn: 5mins

- Go to the 11_RMD_IL_Home_Prices and knit the portfolio.Rmd
- Now go to 12_RMD_Stocks_RMarkdown_Website
- Knit the Index.RMD file…
- How is this different than what we have done so far? How are they similar?
- Now Knit index.Rmd in 8_RMD_Immunogenicity_RMarkdown_Distill
- How is this different? What about Blogdown and Bookdown? Same or different?
But all of my data are in DBs?

DB - Three ways to write queries
1. DBI code
2. dplyr syntax
3. R Notebook SQL language engine
Your Turn

- 1_DB_Examples
- quick_db_demo.Rmd
- Review 5_RMD_Flex_Database to see a report built on data in a DB
Clear your mind of questions.