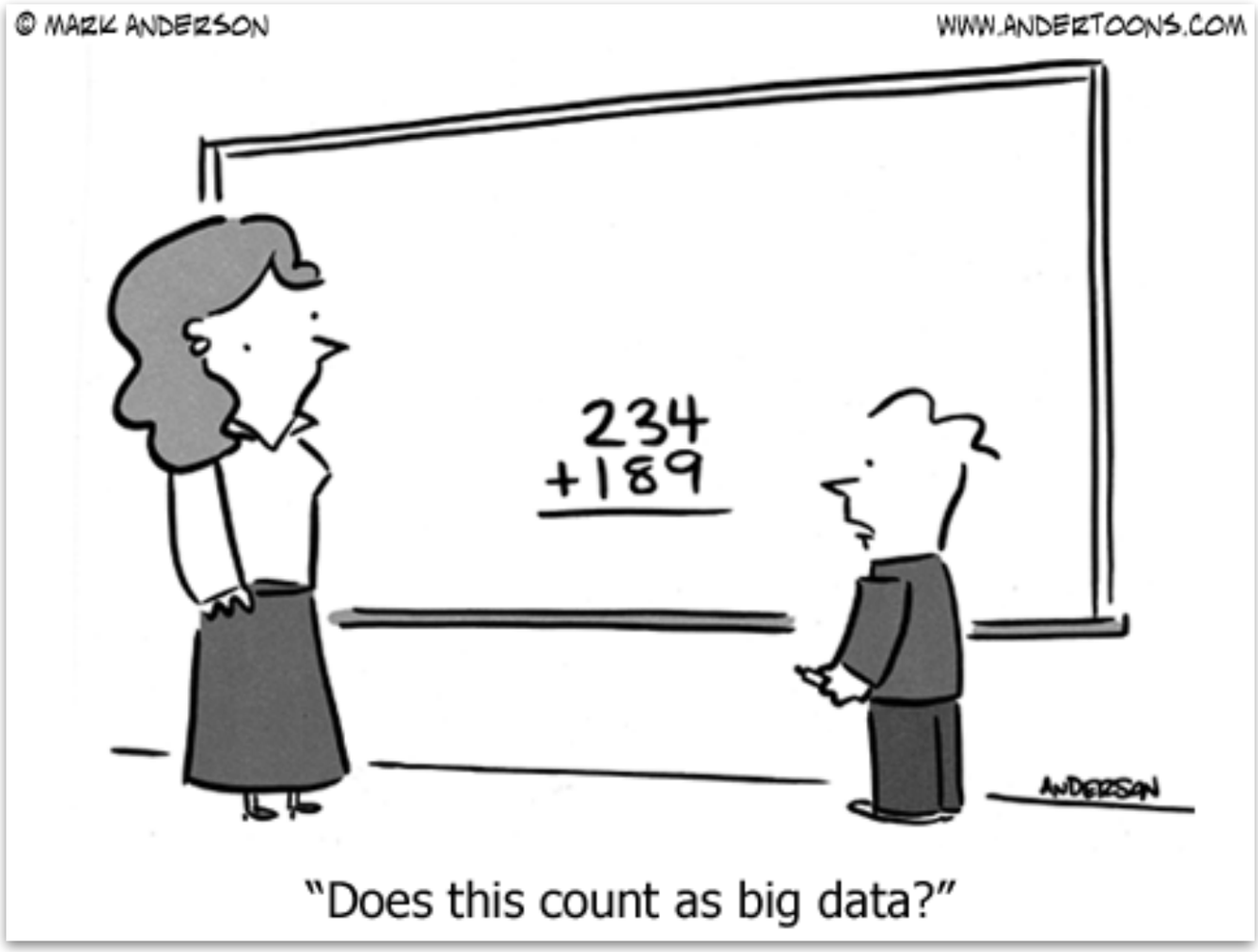
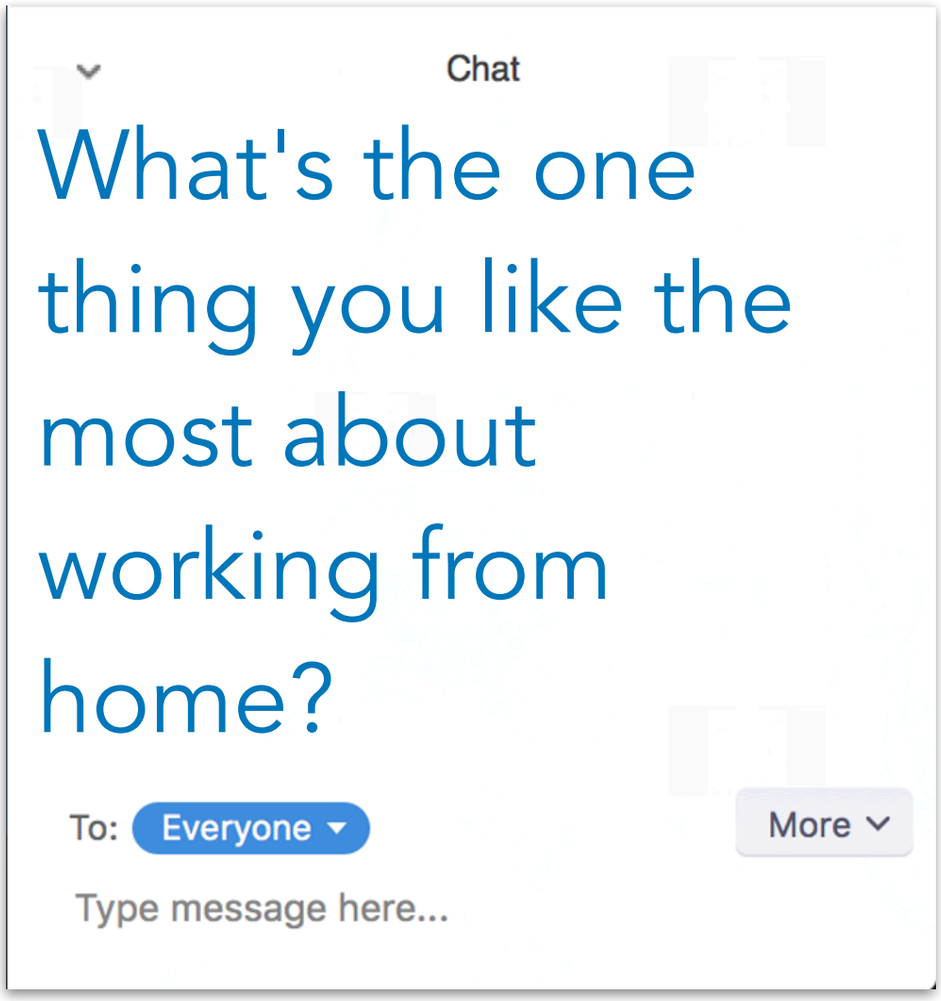
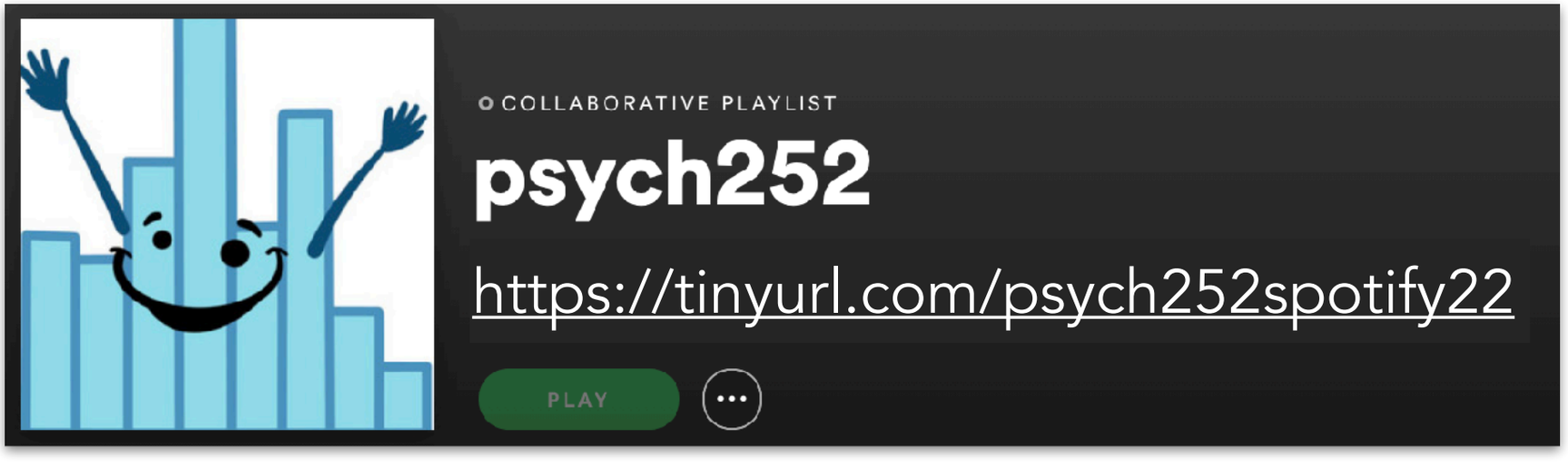
PSYCH 252: Statistical Methods for Behavioral and Social Sciences

Wrangling data 1



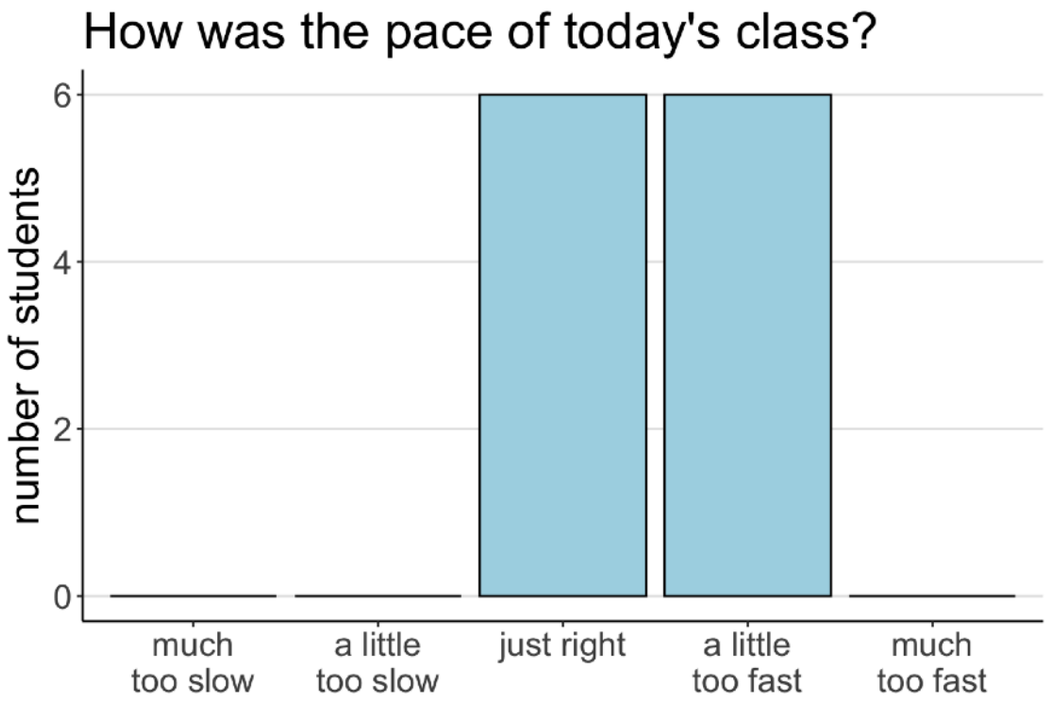


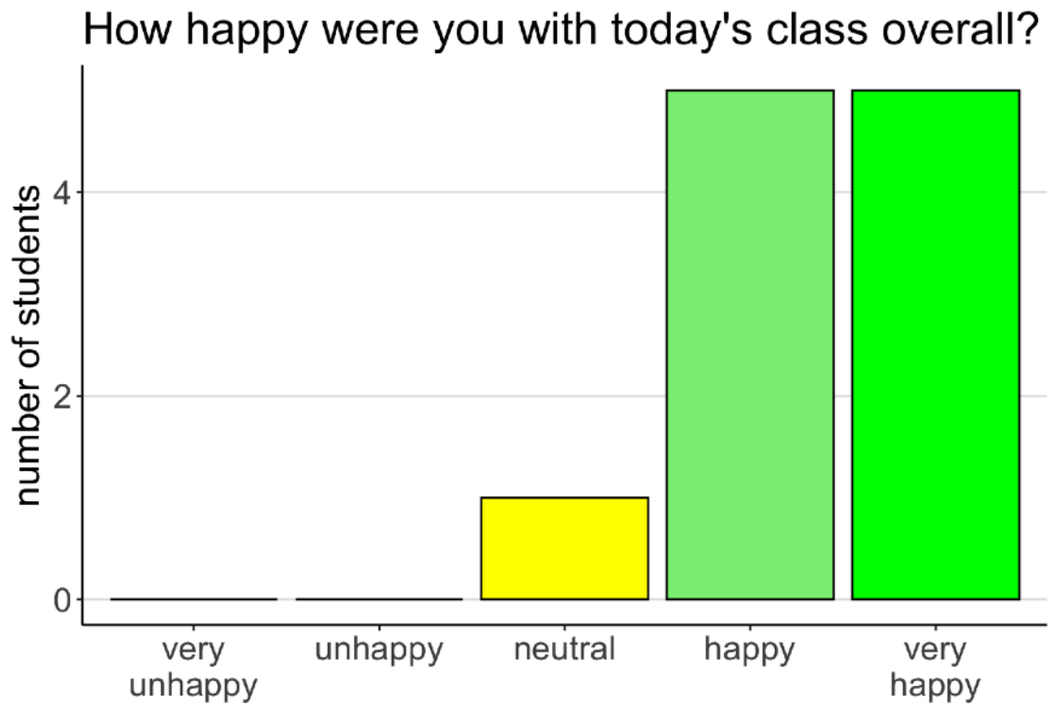


01/10/2022

Your feedback

Your feedback





liked that we dived into actual material, thought it was well-paced. if you could post the rmds with solutions after class (which I think you said you would) that would be super helpful to go back to when we forget something

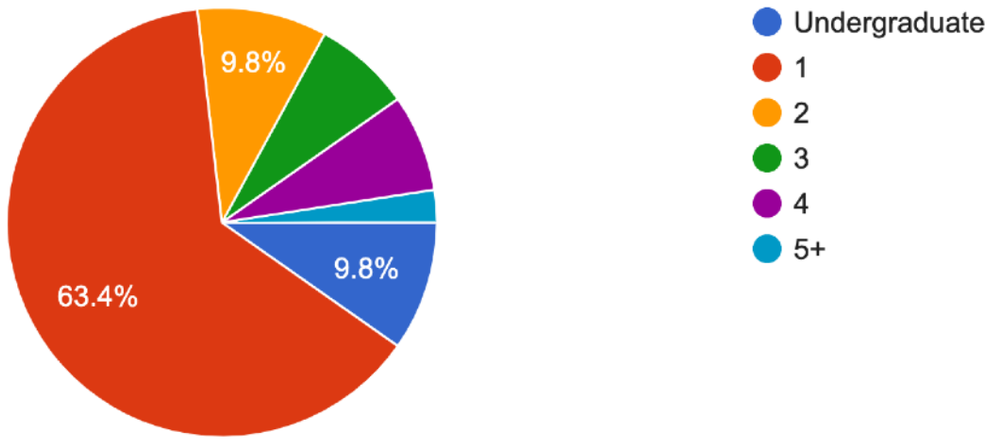
I liked everything. Would be great to have a cheatsheet that we can reference for some common graph types.

Really like the breakout discussion & practice. Learned good stuffs.

Introductory survey

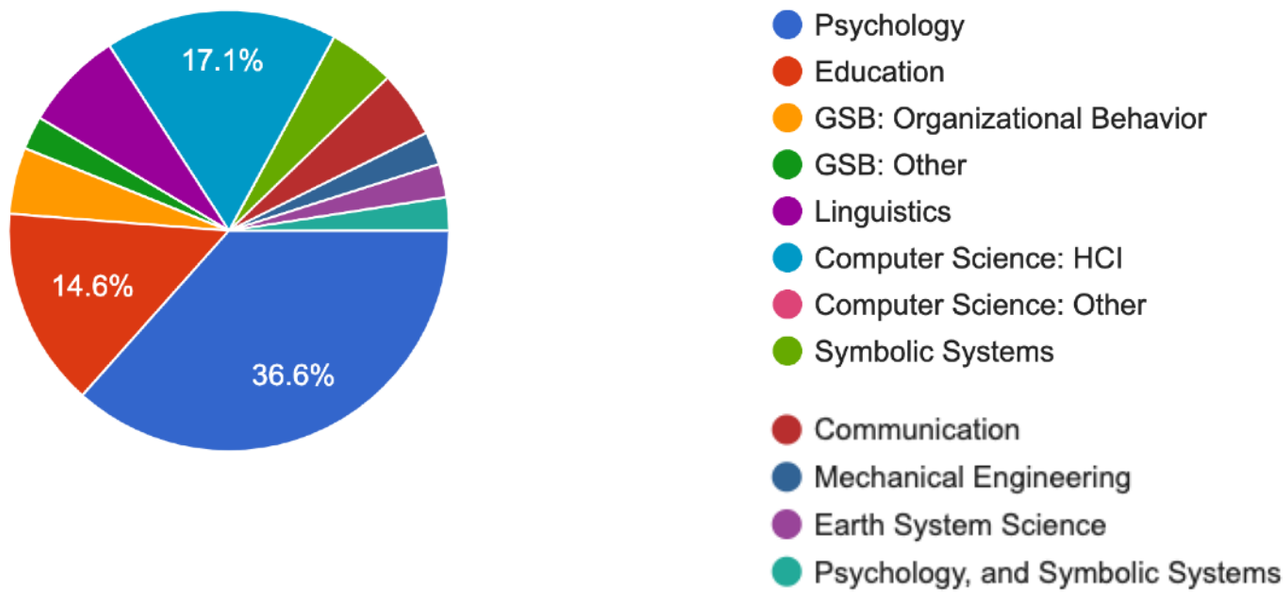
What year of graduate school are you in?

41 responses



What department are you in?

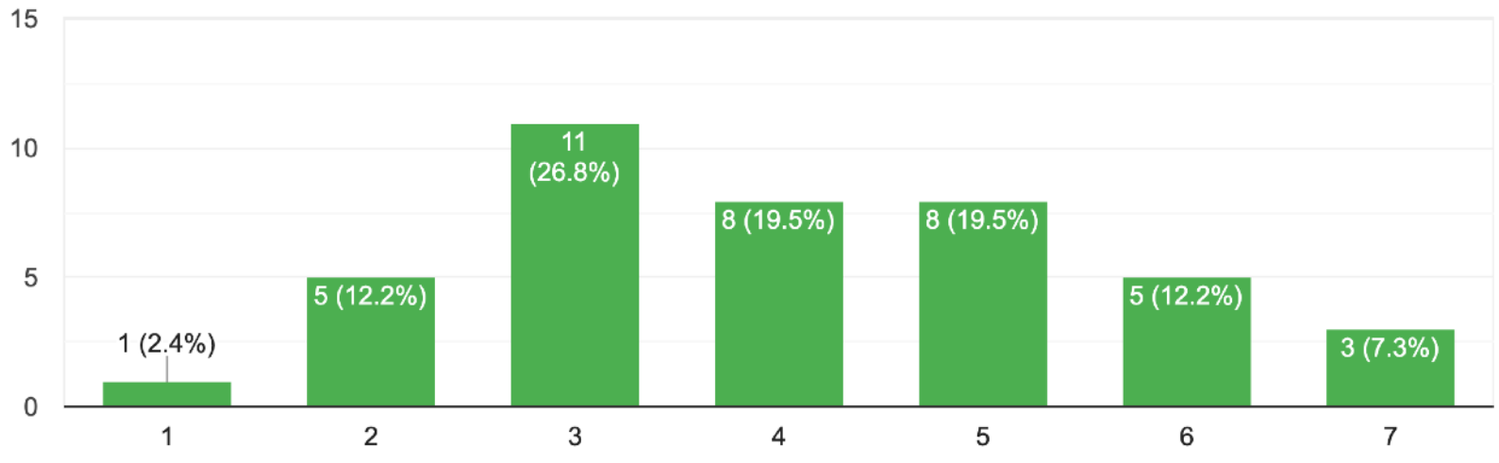
41 responses



Introductory survey

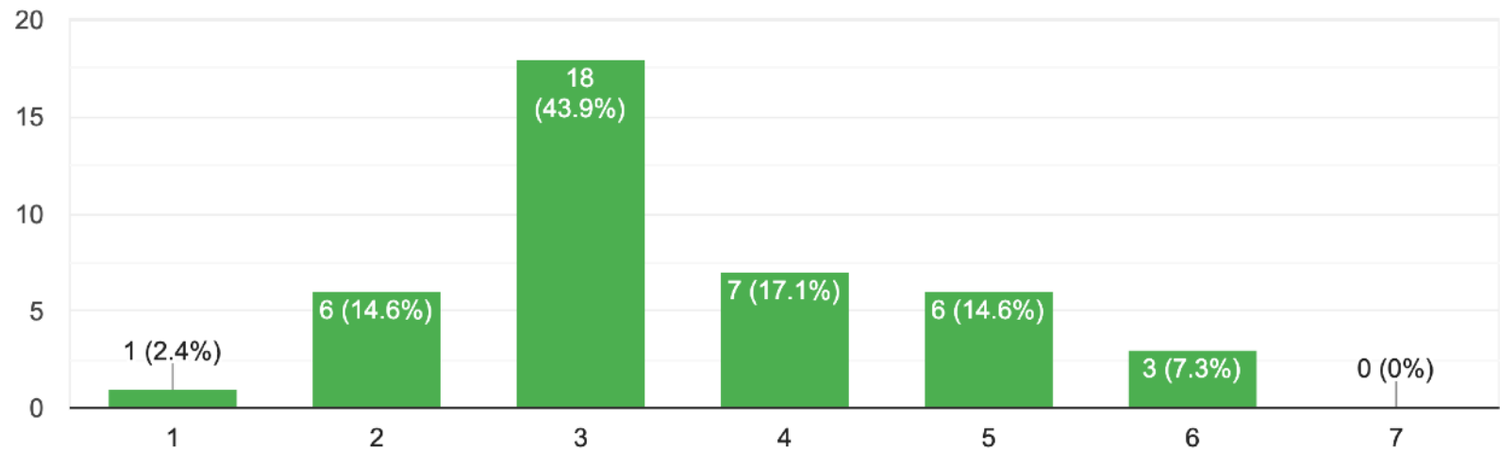
Please rate your level of experience with computer programming

41 responses



Please rate your level of experience with statistics

41 responses



Some tips and tricks

Reprex

this is even better!

- best way to get help is by posting a **reprex**
- **reprex =** reproducible example

reprex

Overview

Prepare reprexes for posting to GitHub issues, StackOverflow, or Slack snippets. What is a `reprex`? It's a reproducible example, as coined by Romain Francois.

Given R code on the clipboard, selected in RStudio, as an expression (quoted or not), or in a file ...

- run it via `rmarkdown::render()`,
- with deliberate choices re: arguments and setup chunk.

Get resulting runnable code + output as

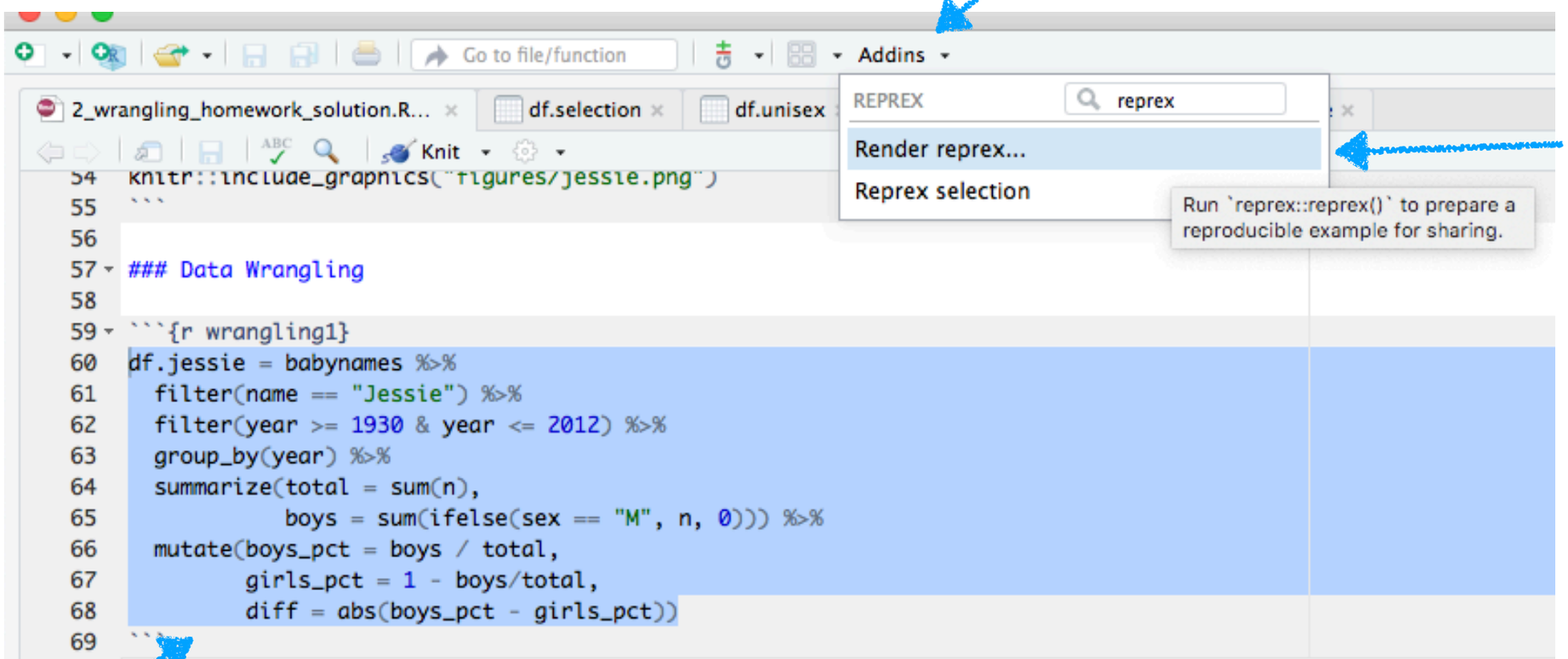
- Markdown, formatted for target venue, e.g. `gh` or `so`, or as
- R code, augmented with commented output.

Result is returned invisibly, placed on the clipboard, and written to a file. Preview an HTML version in RStudio viewer or default browser.

Reprex

```
install.package("reprex")
```

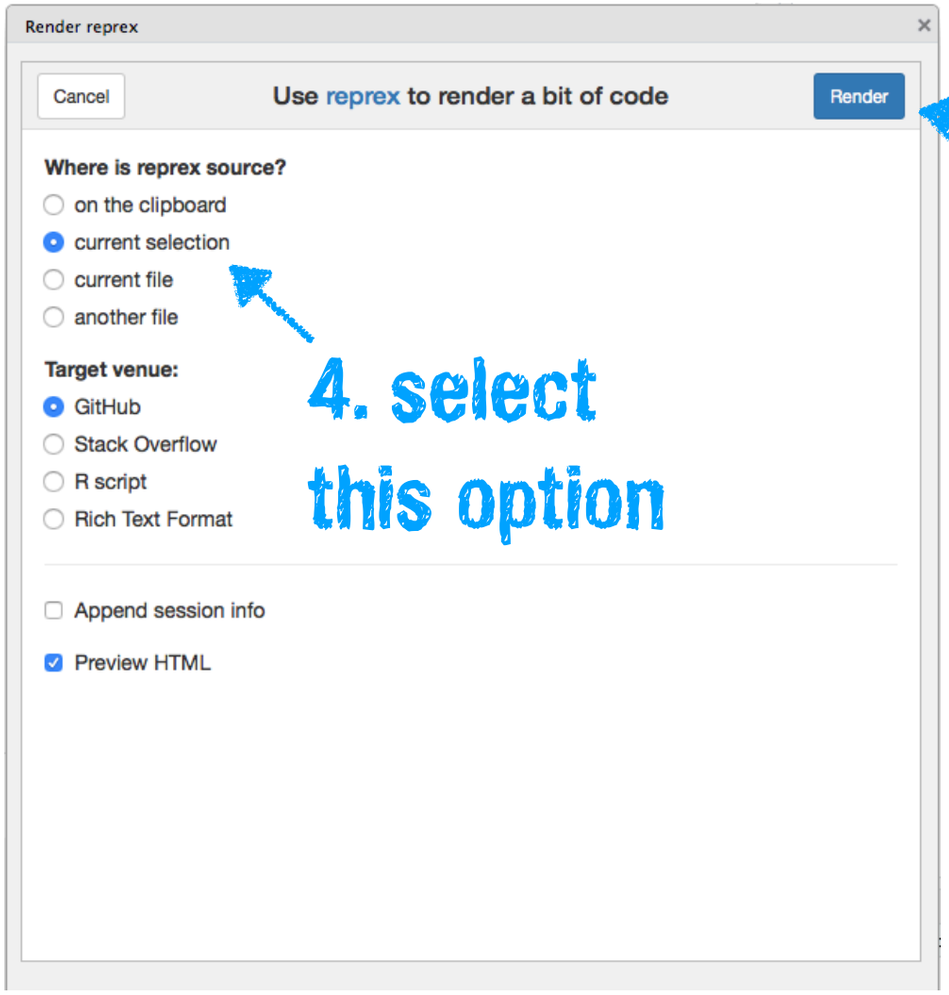
2. click on Addins



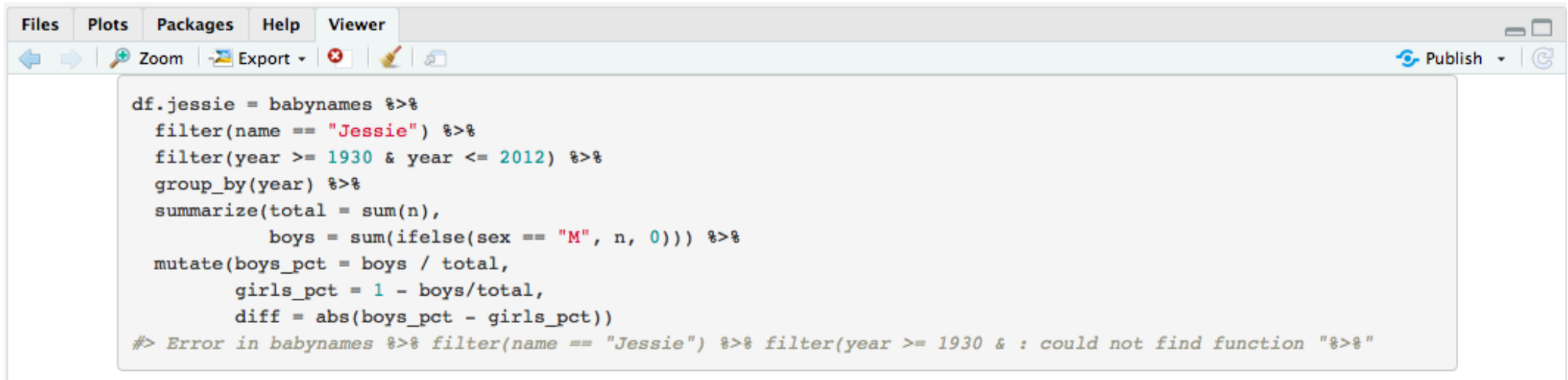
3. Render reprex

1. select the text

Reprex



6. copy and paste from the viewer



Reprex

Files | Plots | Packages | Help | Viewer

Zoom | Export | Publish

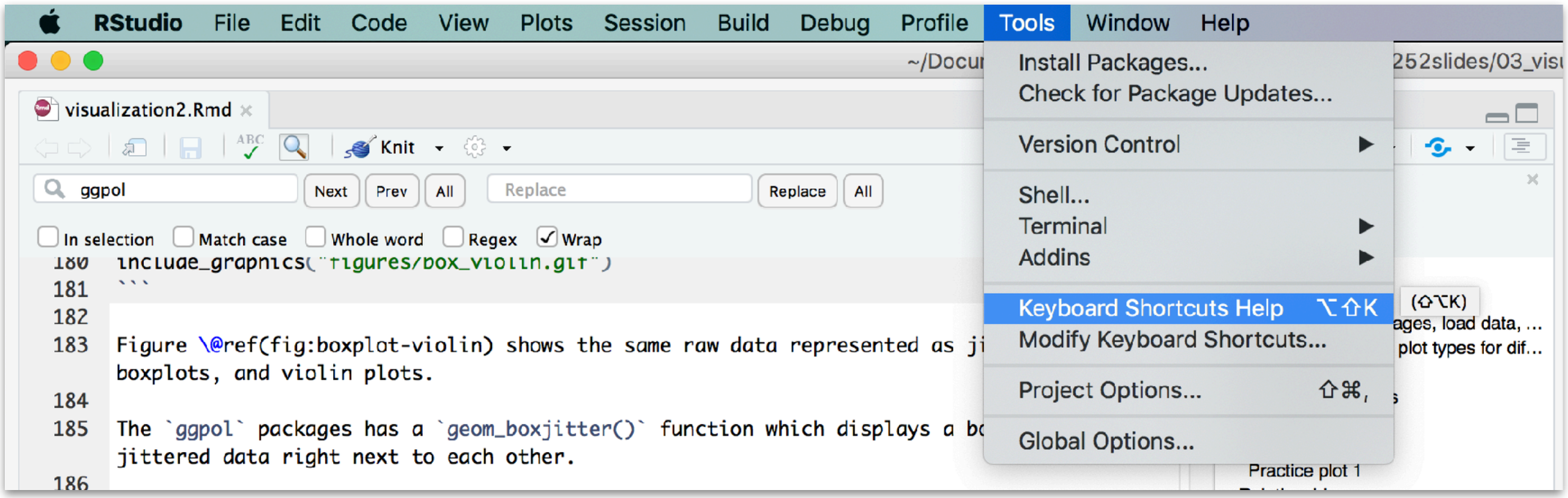
```
library("babynames")
library("tidyverse")
df.jessie = babynames %>%
  filter(name == "Jessie") %>%
  filter(year >= 1930 & year <= 2012) %>%
  group_by(year) %>%
  summarize(total = sum(n),
            boys = sum(ifelse(sex == "M", n, 0))) %>%
  mutate(boys_pct = boys / total,
         girls_pct = 1 - boys/total,
         diff = abs(boys_pct - girls_pct)) %>%
  print()
#> # A tibble: 83 x 6
#>     year total  boys boys_pct girls_pct  diff
#>    <dbl> <int> <dbl>    <dbl>     <dbl> <dbl>
#>  1  1930  3525  1329    0.377     0.623 0.246
#>  2  1931  3196  1267    0.396     0.604 0.207
#>  3  1932  3178  1282    0.403     0.597 0.193
#>  4  1933  2886  1079    0.374     0.626 0.252
#>  5  1934  2883  1090    0.378     0.622 0.244
#>  6  1935  2721  1103    0.405     0.595 0.189
#>  7  1936  2599  1012    0.389     0.611 0.221
#>  8  1937  2589  1041    0.402     0.598 0.196
#>  9  1938  2446   970    0.397     0.603 0.207
#> 10  1939  2454  1058    0.431     0.569 0.138
#> # … with 73 more rows
```

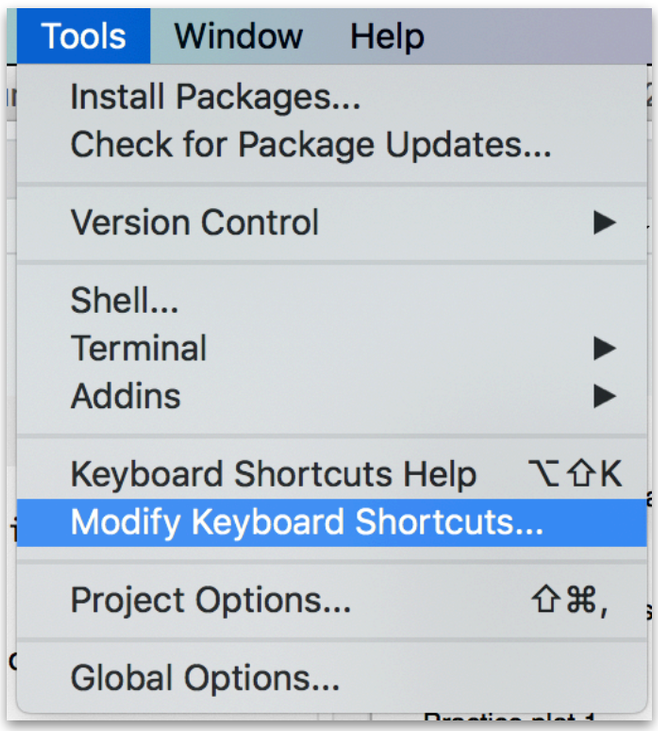
7. make sure to load necessary packages in your reprex

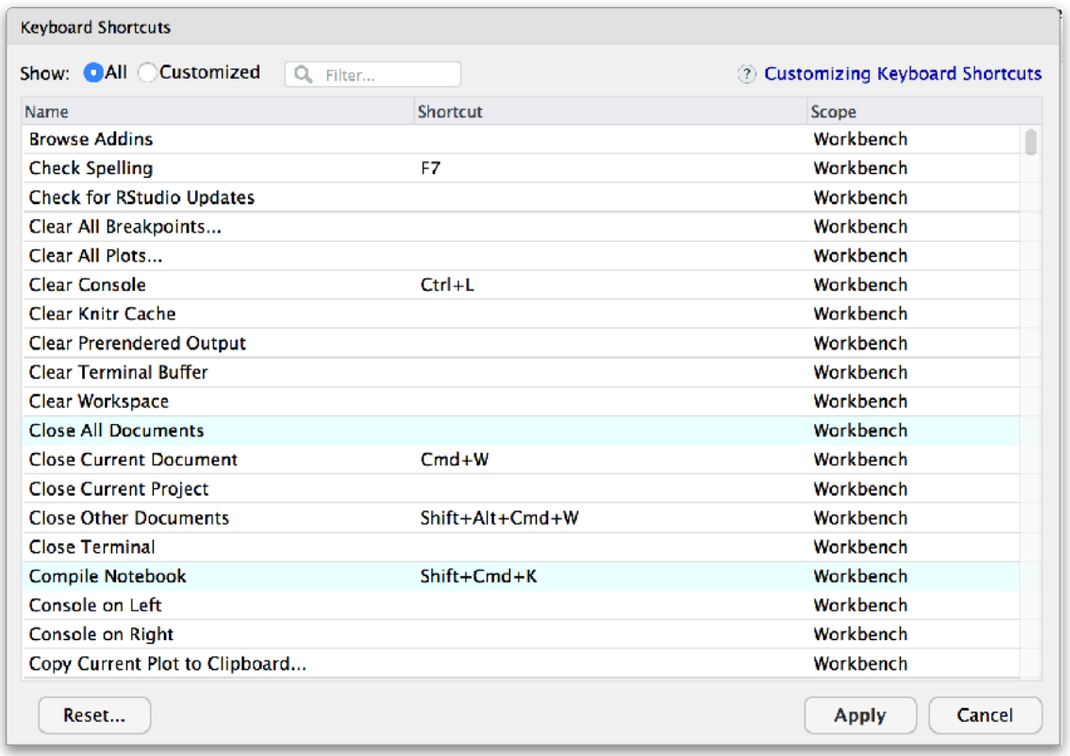
Created on 2019-01-24 by the reprex package (v0.2.1)

Learn the keyboard shortcuts!

... and make your own







Reformatting code

```
ggplot(data = df.diamonds[1:150,], mapping = aes(x = color, y = price)) +
  # individual data points (jittered horizontally)
      geom_point(alpha = 0.2,
                 position = position_jitter(width = 0.1, height = 0),
             size = 2) +
  # error bars
  stat_summary(fun.data = "mean_cl_boot",
               geom = "linerange",
      color = "black",
                   size = 1) +
        # means
stat_summary(fun.y = "mean",
               geom = "point",
shape = 21,
               fill = "red",
               color = "black",
               size = 4)
```

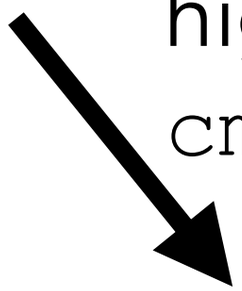
highlight code and press

`cmd + i`

```
ggplot(data = df.diamonds[1:150,], mapping = aes(x = color, y = price)) +
  # individual data points (jittered horizontally)
  geom_point(alpha = 0.2,
             position = position_jitter(width = 0.1, height = 0),
             size = 2) +
  # error bars
  stat_summary(fun.data = "mean_cl_boot",
               geom = "linerange",
               color = "black",
               size = 1) +
  # means
  stat_summary(fun.y = "mean",
               geom = "point",
               shape = 21,
               fill = "red",
               color = "black",
               size = 4)
```

Commenting code

```
1 ggplot(data = df.diamonds,
2        mapping = aes(x = color, y = price)) +
3   stat_summary(fun.y = "mean", geom = "bar")
```



highlight code and press

`cmd + shift + c`

```
1 # ggplot(data = df.diamonds,
2 #        mapping = aes(x = color, y = price)) +
3 #   stat_summary(fun.y = "mean", geom = "bar")
```

Quickly copying code

```
1 ggplot(mapping = aes(x = color, y = price), data = df.diamonds) +
2   stat_summary(fun.y = "mean", geom = "point")
```

put cursor anywhere in line 1

`cmd + shift + d`

```
1 ggplot(mapping = aes(x = color, y = price), data = df.diamonds) +
2 ggplot(mapping = aes(x = color, y = price), data = df.diamonds) +
3   stat_summary(fun.y = "mean", geom = "point")
```

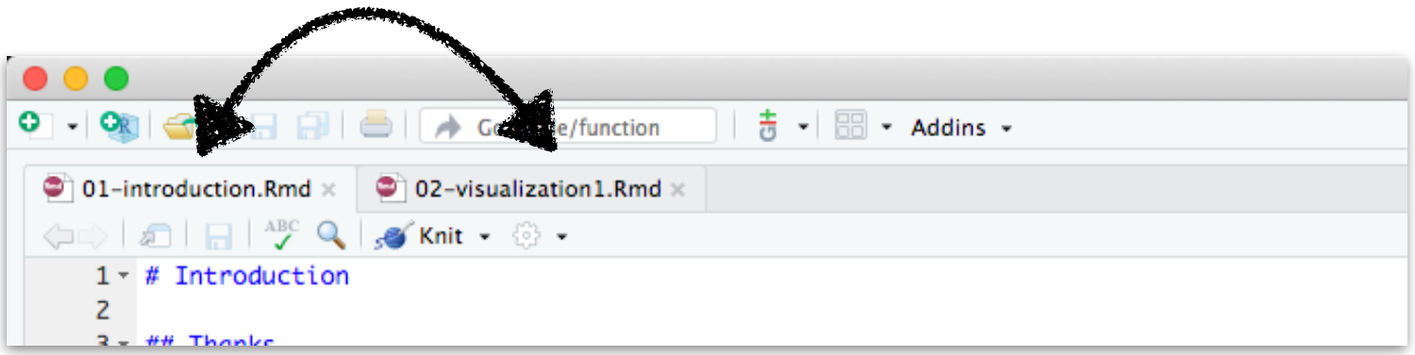
comment

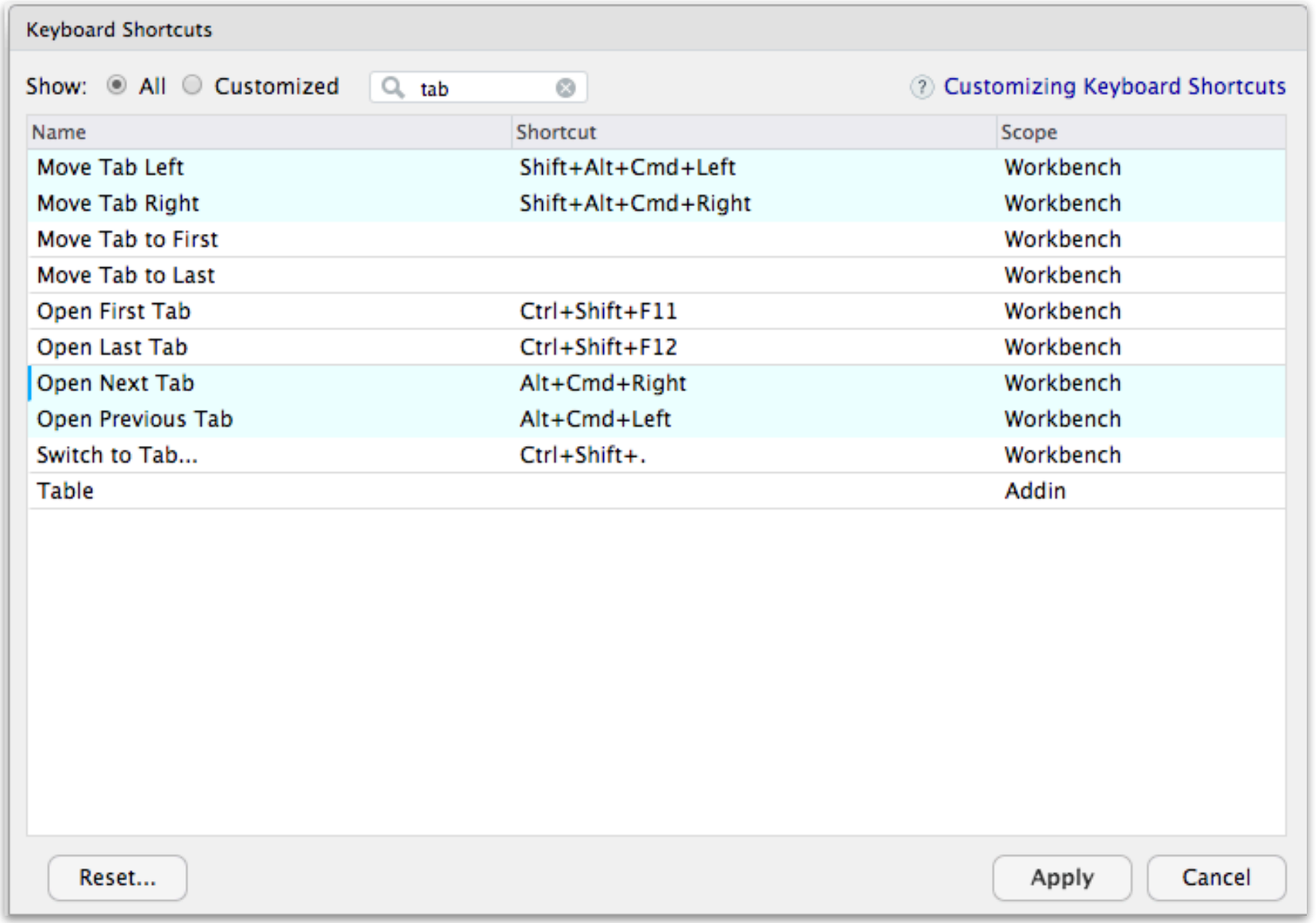
`cmd + shift + c`

```
1 # ggplot(mapping = aes(x = color, y = price), data = df.diamonds) +
2 ggplot(mapping = aes(x = cut, y = price), data = df.diamonds) +
3   stat_summary(fun.y = "mean", geom = "point")
```

change

Navigating between tabs





Jumping between code chunks

````
190  Here is the help file for the `print()` function:
191
192  ```{r visualization1-17, echo=FALSE, fig.cap="Help file for the print()
     function.", out.width="95%"}
193  include_graphics("figures/help_print.png")
194  ```
195
196  ## Data visualization using `ggplot2`
197
198  We will use the `ggplot2` package to visualize data. By the end of next class,
     you'll be able to make a figure like this:
199
200  ```{r visualization1-18, echo=FALSE, fig.cap="What a nice figure!",
     out.width="95%"}
201  include_graphics("figures/combined_plot.png")
202  ```
````

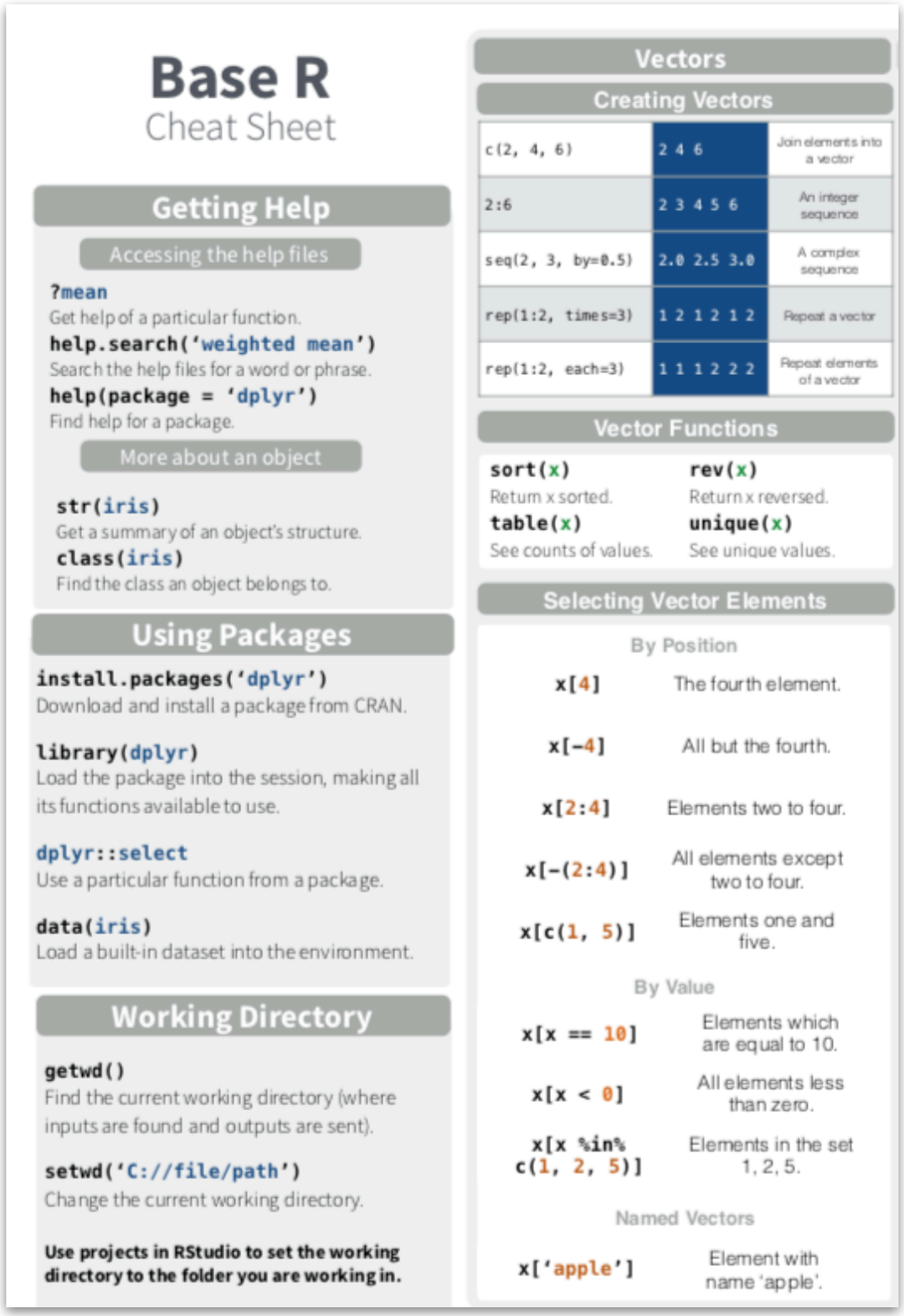
Keyboard Shortcuts

Show: All Customized chunk Customizing Keyboard Shortcuts

Name	Shortcut	Scope
Restart R Session and Clear Chunk Output		Workbench
Restart R Session and Run All Chunks		Workbench
Go to Next Chunk	Alt+Cmd+Down	Editor
Go to Previous Chunk	Alt+Cmd+Up	Editor
Insert Chunk	Alt+Cmd+I	Editor

Data wrangling 1

Two styles of coding in R



Base R
Cheat Sheet

Getting Help

Accessing the help files

?mean
Get help of a particular function.
help.search('weighted mean')
Search the help files for a word or phrase.
help(package = 'dplyr')
Find help for a package.

More about an object

str(iris)
Get a summary of an object's structure.
class(iris)
Find the class an object belongs to.

Using Packages

install.packages('dplyr')
Download and install a package from CRAN.

library(dplyr)
Load the package into the session, making all its functions available to use.

dplyr::select
Use a particular function from a package.

data(iris)
Load a built-in dataset into the environment.

Working Directory

getwd()
Find the current working directory (where inputs are found and outputs are sent).

setwd('C://file/path')
Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

Vectors

Creating Vectors

c(2, 4, 6)	2 4 6	Join elements into a vector
2:6	2 3 4 5 6	An integer sequence
seq(2, 3, by=0.5)	2.0 2.5 3.0	A complex sequence
rep(1:2, times=3)	1 2 1 2 1 2	Repeat a vector
rep(1:2, each=3)	1 1 1 2 2 2	Repeat elements of a vector

Vector Functions

sort(x) Return x sorted.
rev(x) Return x reversed.
table(x) See counts of values.
unique(x) See unique values.

Selecting Vector Elements

By Position

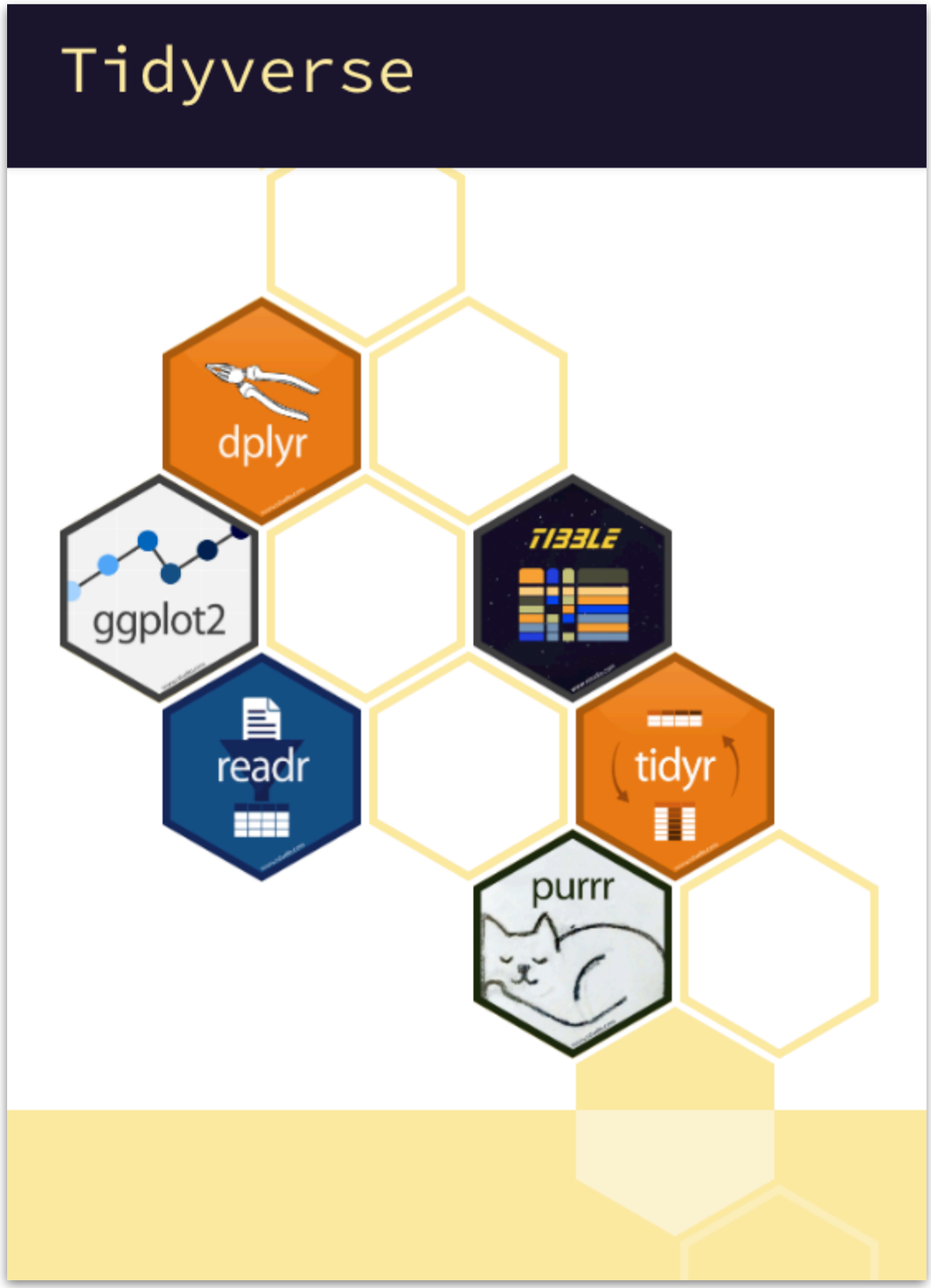
x[4]	The fourth element.
x[-4]	All but the fourth.
x[2:4]	Elements two to four.
x[-(2:4)]	All elements except two to four.
x[c(1, 5)]	Elements one and five.

By Value

x[x == 10]	Elements which are equal to 10.
x[x < 0]	All elements less than zero.
x[x %in% c(1, 2, 5)]	Elements in the set 1, 2, 5.

Named Vectors

x['apple']	Element with name 'apple'.



Software can be chaotic, but we make it work

Expert

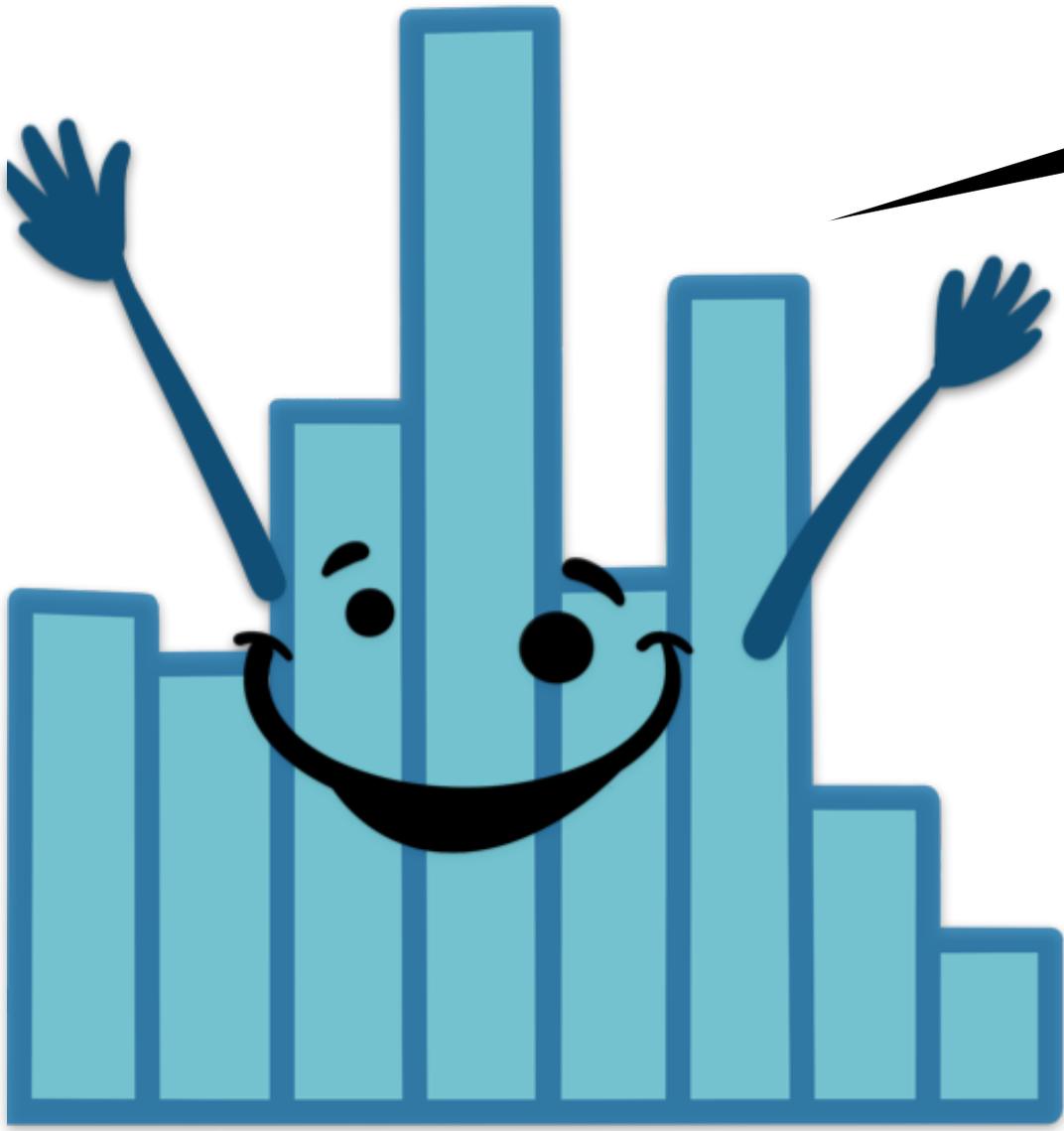
Trying Stuff Until it Works

O RLY?

The Practical Developer

@ThePracticalDev

RStudio time



01:00

stretch break!

Using functions with `tidyverse` verbs

```
1 df.starwars %>%
2   select(where(fn = is.numeric))
```

my recommendation

- `fn = is.numeric`
- `fn = "is.numeric"`
- `fn = function(x){is.numeric(x)}`
- `fn = ~ is.numeric(.)`

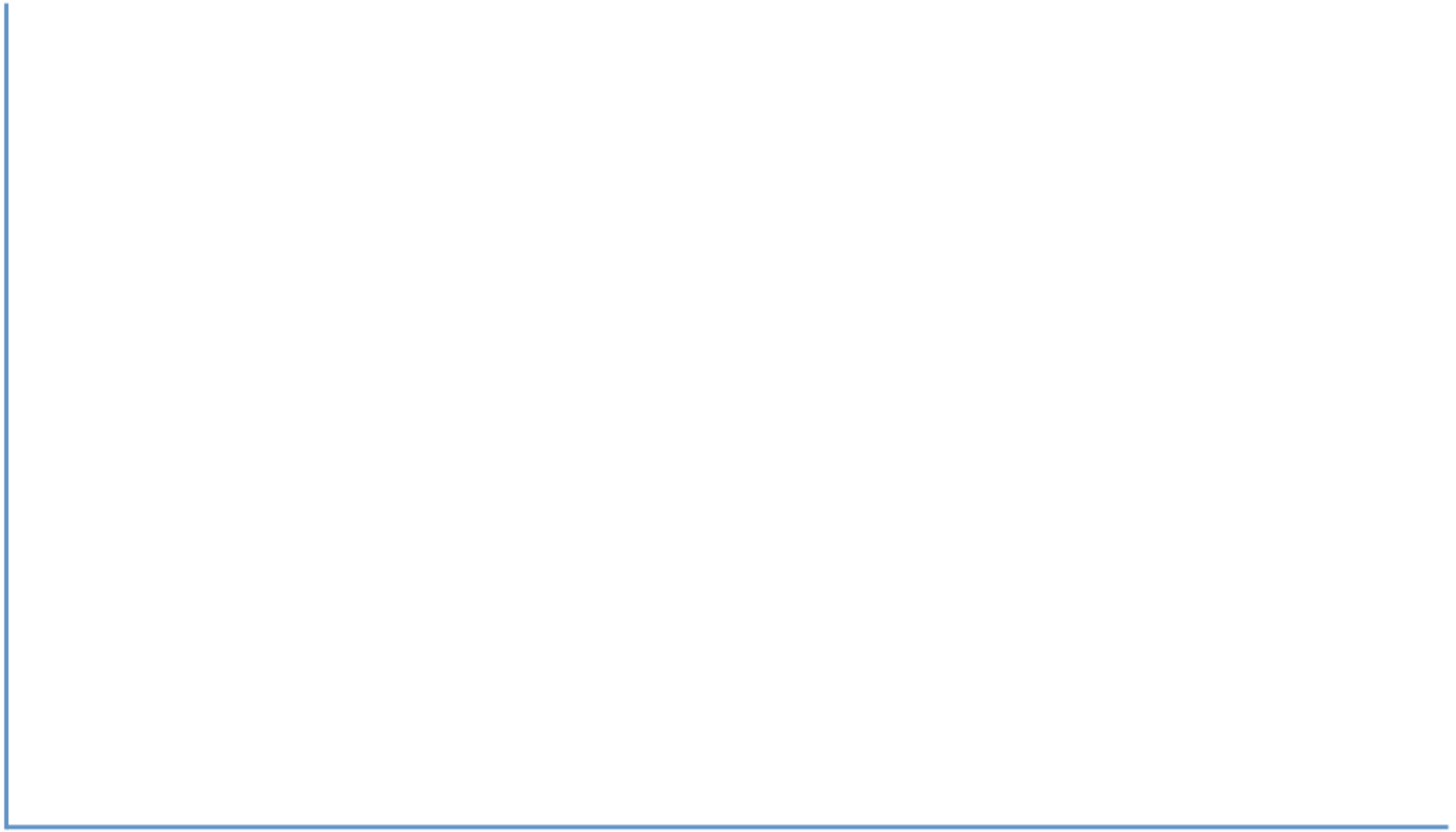
flexible, short, works well with other verbs we'll learn about later

- `fn = ~ !is.numeric(.)`

select all variables that are not numeric

Feedback

How was the pace of today's class?



How happy were you with today's class overall?

What did you like about today's class? What could be improved next time?