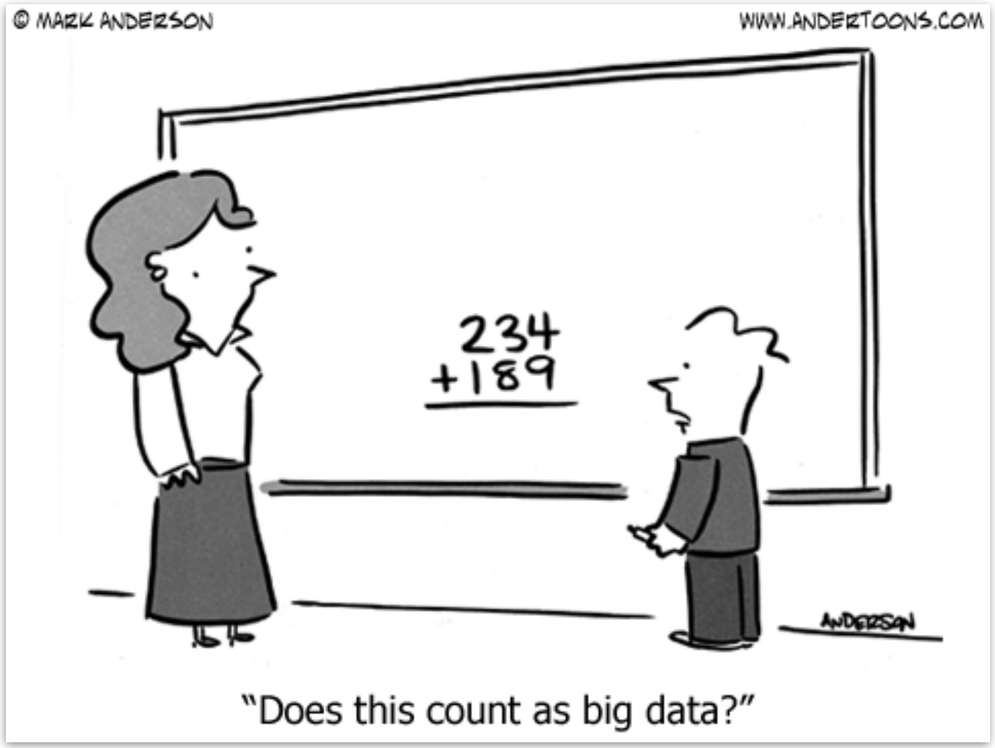
PSYCH 252: Statistical Methods for Behavioral and Social Sciences

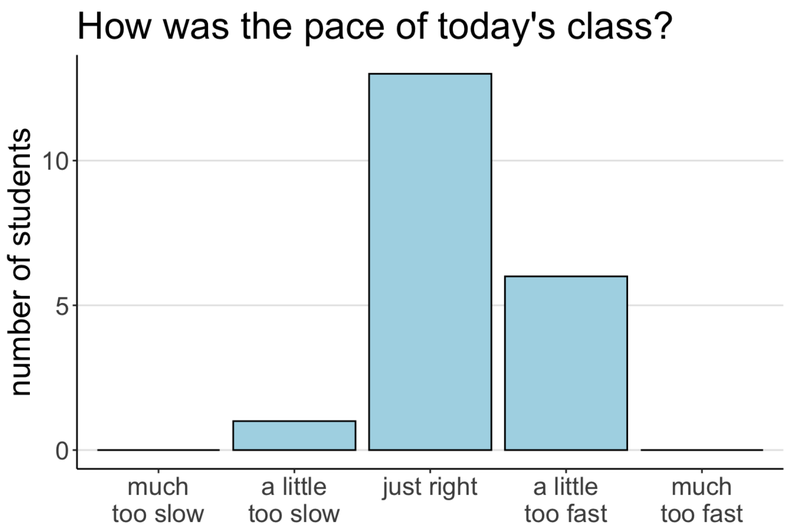
Wrangling data 1

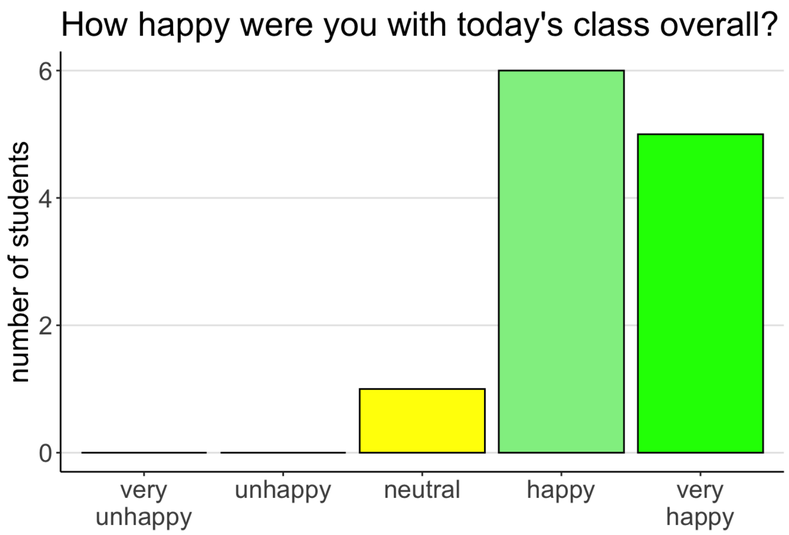


1/13/2025

Your feedback

Your feedback





I appreciate how the instruction is tailored to the skill and experience level of the class, and I think Tobi does a nice job of transitioning from more advanced to more basic topics as needed. It would be helpful to have perhaps an extra minute to work on the practice plots during class

Sometimes you finish giving the answer for an activity but does not allow for time so we can fix our own. Would be helpful to just have a few seconds before we move on. Thanks :)

Things that came up

Coding = learning a new language

Martin Hebart
@martinhebart.bsky.social

+ Follow

As academics, we spend much of our time on data analysis, writing code that allows us to analyze complex data patterns. As we are at the brink of AI getting better at programming than us, do you still incentivize your students to learn to write complex code from scratch? Or should the focus shift?

January 13, 2025 at 1:58 AM · Everybody can reply

Lisa DeBruine @debruine.bsky.social · 4h

That's my argument when teaching the intro coding class for our social science masters students. I like the analogy of those translation fail memes — you need to at least know enough to spot the gibberish.

Timothy O'Leary
@timothyoleary.bsky.social

+ Follow

Suppose there's no AI to write your code, and instead you contract it out to a human. Do you trust it at face value? Or do you do a few sanity checks?

Here's the point: how would you know *how to* sanity check without knowing the fundamentals yourself? How would you know what *could* go wrong?

> **Martin Hebart** @martinhebart.bsky.social · 7h
> As academics, we spend much of our time on data analysis, writing code that allows us to analyze complex data patterns. As we are at the brink of AI getting better at programming than us, do you still incentivize your students to learn to write complex code from scratch? Or should the focus shift?

Anatomy of a nice `ggplot`

```
# ggplot call with global aesthetics
ggplot(data = data,
       mapping = aes(x = cause,
                     y = effect)) +
  # add geometric objects (geoms)
  geom_point() +
  stat_summary(fun.y = "mean", geom = "point") +
  ... +
  # add text objects
  geom_text() +
  annotate() +
  # adjust axes and coordinates
  scale_x_continuous() +
  scale_y_continuous() +
  coord_cartesian() +
  # define plot title, and axis titles
  labs(title = "Title",
       x = "Cause",
       y = "Effect") +
  # change global aspects of the plot
  theme(text = element_text(size = 20),
        plot.margin = margin(t = 1, b = 1, l = 0.5, r = 0.5, unit = "cm")) +
  # save the plot
  ggsave(filename = "super_nice_plot.pdf",
         width = 8,
         height = 6)
```

what?

how?

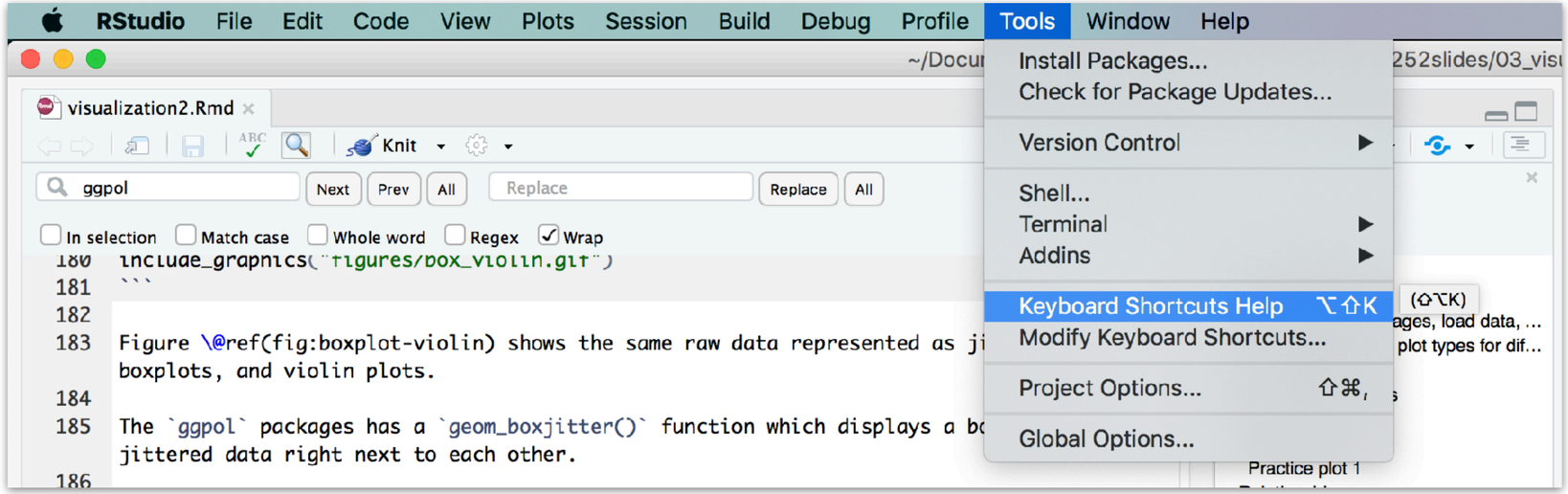
add some text?

"local" adjustments

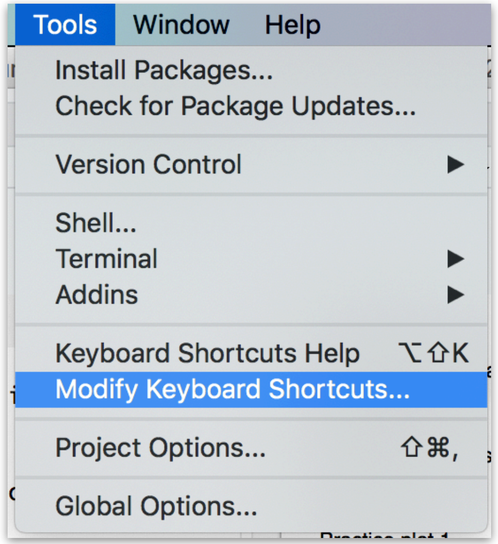
"global" adjustments

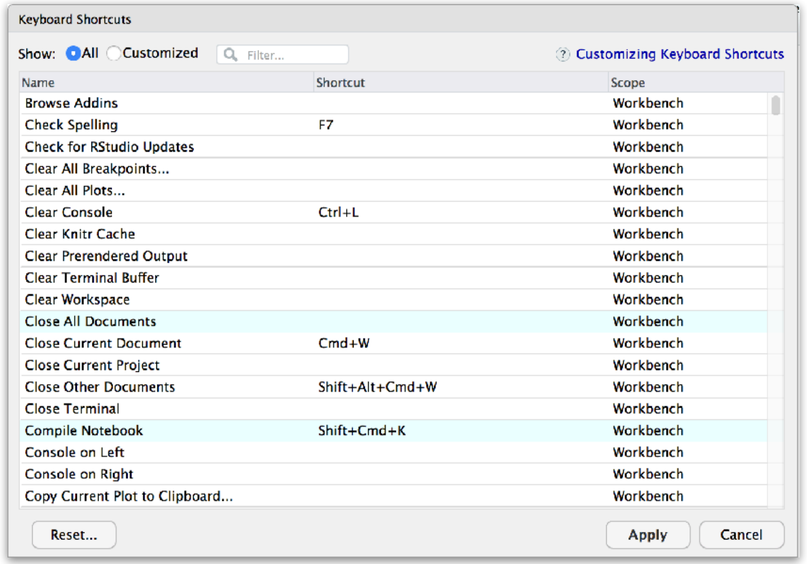
save the beauty!

Learn the keyboard shortcuts!

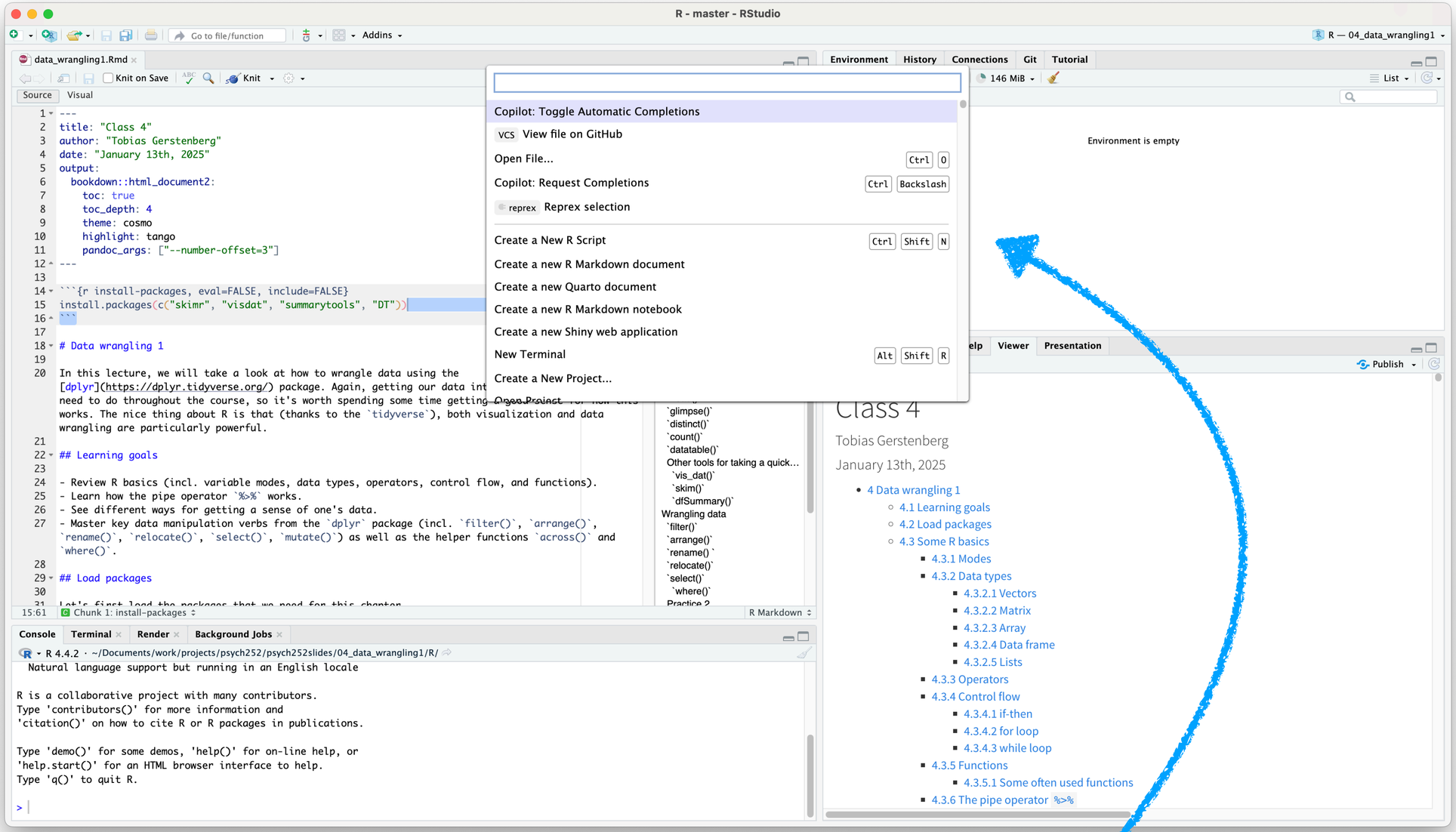


... and make your own



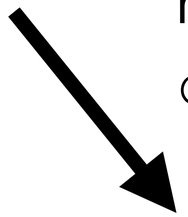


Learn the keyboard shortcuts!



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")
```

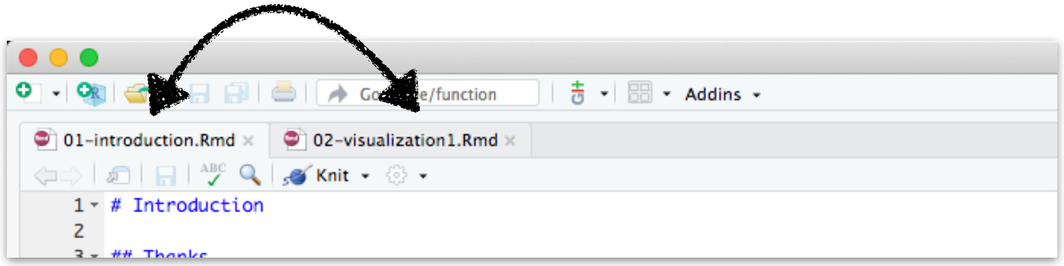
comment

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



Keyboard Shortcuts

Show: All Customized tab Customizing Keyboard Shortcuts

Name	Shortcut	Scope
Move Tab Left	Shift+Alt+Cmd+Left	Workbench
Move Tab Right	Shift+Alt+Cmd+Right	Workbench
Move Tab to First		Workbench
Move Tab to Last		Workbench
Open First Tab	Ctrl+Shift+F11	Workbench
Open Last Tab	Ctrl+Shift+F12	Workbench
Open Next Tab	Alt+Cmd+Right	Workbench
Open Previous Tab	Alt+Cmd+Left	Workbench
Switch to Tab...	Ctrl+Shift+.	Workbench
Table		Addin

Reset... Apply Cancel

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  ```
````

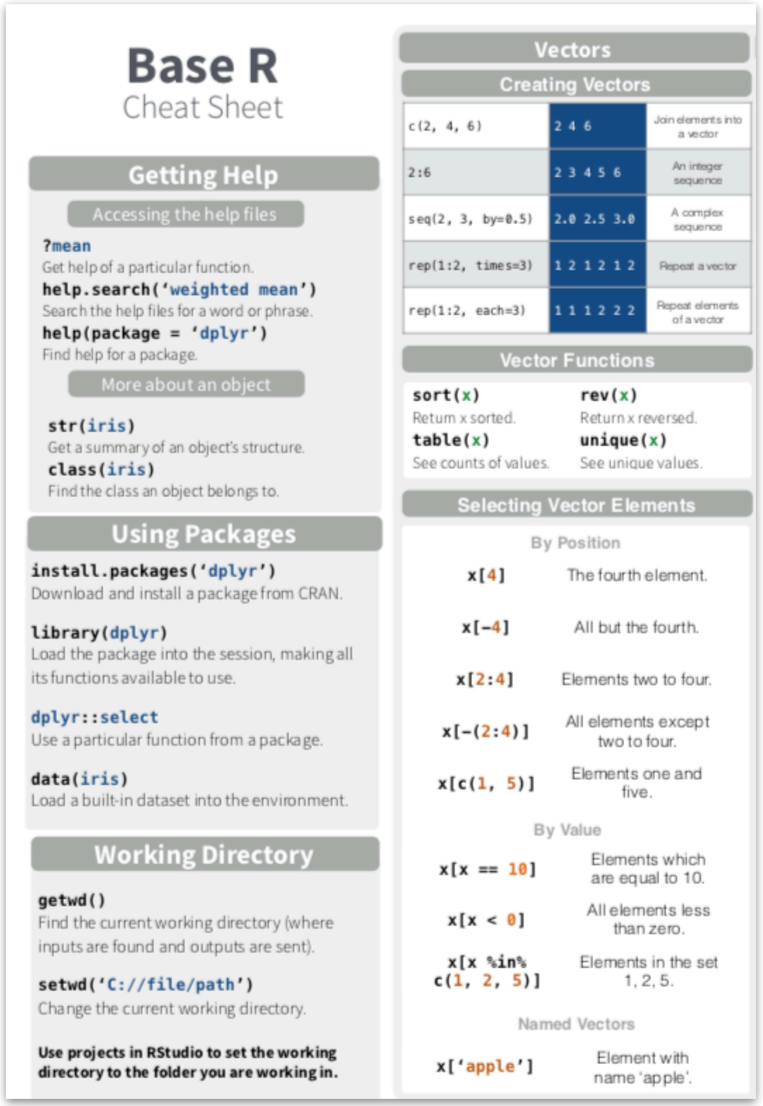
Keyboard Shortcuts

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

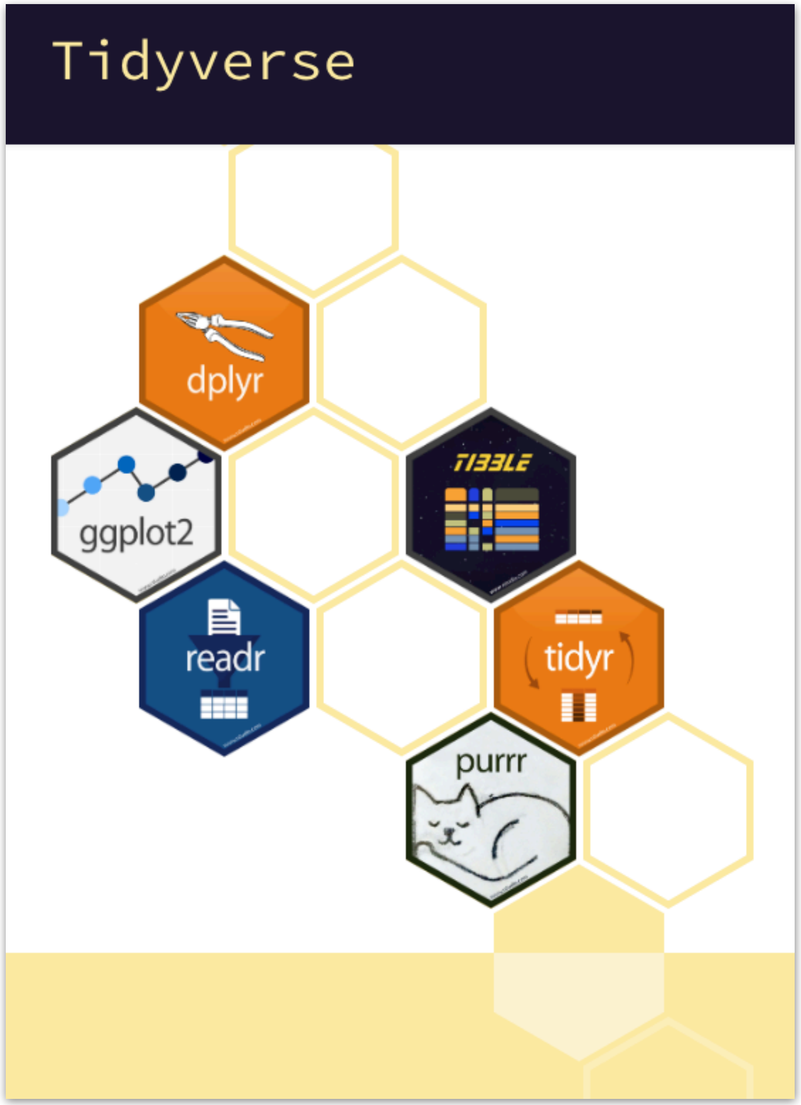
x[4]	The fourth element.
x[-4]	All but the fourth.
x[2:4]	Elements two to four.
x[-(2:4)]	All elements except two to four.
x[c(1, 5)]	Elements one and five.

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

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

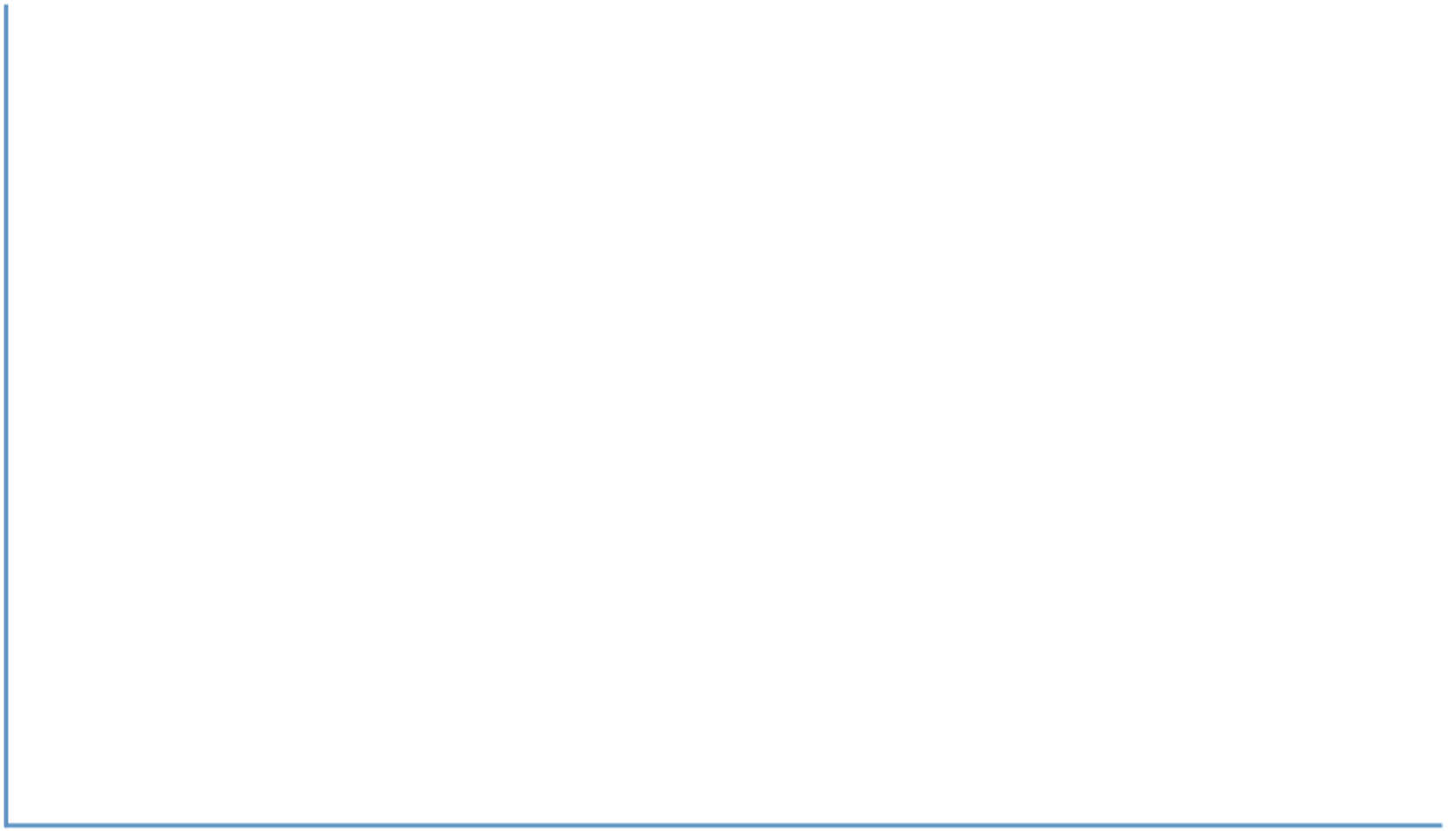
Timers

blue

pink

Feedback

How was the pace of today's class?



Start the presentation to see live content. Still no live content? Install the app or get help at **PollEv.com/app**

How happy were you with today's class overall?

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