

Wrangling data 1



COLLABORATIVE PLAYLIST

psych252

<https://tinyurl.com/psych252spotify26>

PLAY

...

Cynthia Wu

Pronouns: she/her

Department: Psychology

Interests:

- Emotions & decisions

Fun facts:

- I did a lot of baking and ended up filling my fridge instead of my stomach



Quick recap

Anatomy of a nice ggplot

```
1 # ggplot call with global aesthetics  
2 ggplot(data = data,  
3         mapping = aes(x = cause,  
4                             y = effect)) +  
5 # add geometric objects (geoms)  
6 geom_point() +  
7 stat_summary(fun.y = "mean", geom = "point") +  
8 ... +  
9 # add text objects  
10 geom_text() +  
11 annotate() +  
12 # adjust axes and coordinates  
13 scale_x_continuous() +  
14 scale_y_continuous() +  
15 coord_cartesian() +  
16 # define plot title, and axis titles  
17 labs(title = "Title",  
18       x = "Cause",  
19       y = "Effect") +  
20 # change global aspects of the plot  
21 theme(text = element_text(size = 20),  
22        plot.margin = margin(t = 1, b = 1, l = 0.5, r = 0.5, unit = "cm")) +  
23 # save the plot  
24 ggsave(filename = "super_nice_plot.pdf",  
25         width = 8,  
26         height = 6)
```

The diagram illustrates the structure of a ggplot command with handwritten annotations:

- A blue arrow points from the word "what?" to the first line of code: `ggplot(data = data,`.
- A blue arrow points from the word "how?" to the second line of code: `mapping = aes(x = cause,`.
- A blue arrow points from the phrase "add some text?" to the ninth line of code: `geom_text() +`.
- A blue arrow points from the phrase "'local' adjustments" to the twelfth line of code: `scale_x_continuous() +`.
- A blue arrow points from the phrase "'global' adjustments" to the twenty-first line of code: `theme(text = element_text(size = 20),`.
- A blue arrow points from the phrase "save the beauty!" to the twenty-fourth line of code: `ggsave(filename = "super_nice_plot.pdf",`.

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



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?



Lisa DeBruine @debruine.bsky.social · 4h

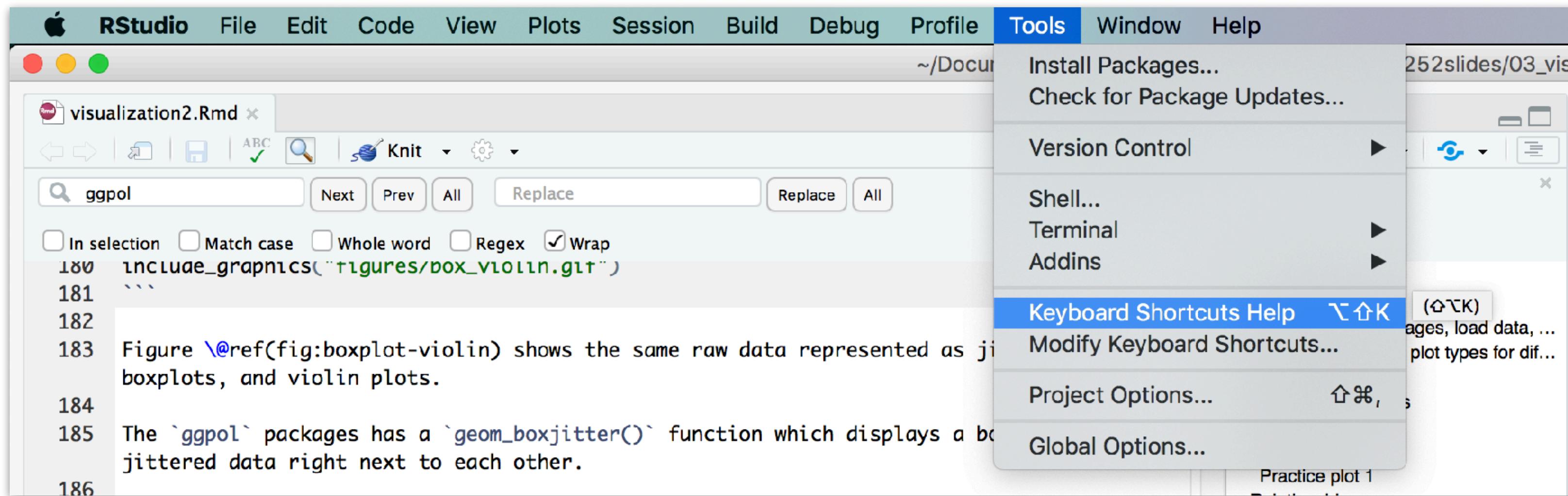
That's my argument when teaching the intro coding class for our social science masters students. I like the analogy of those translation fail memes — you need to at least know enough to spot the gibberish.



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?

Learn the keyboard shortcuts!

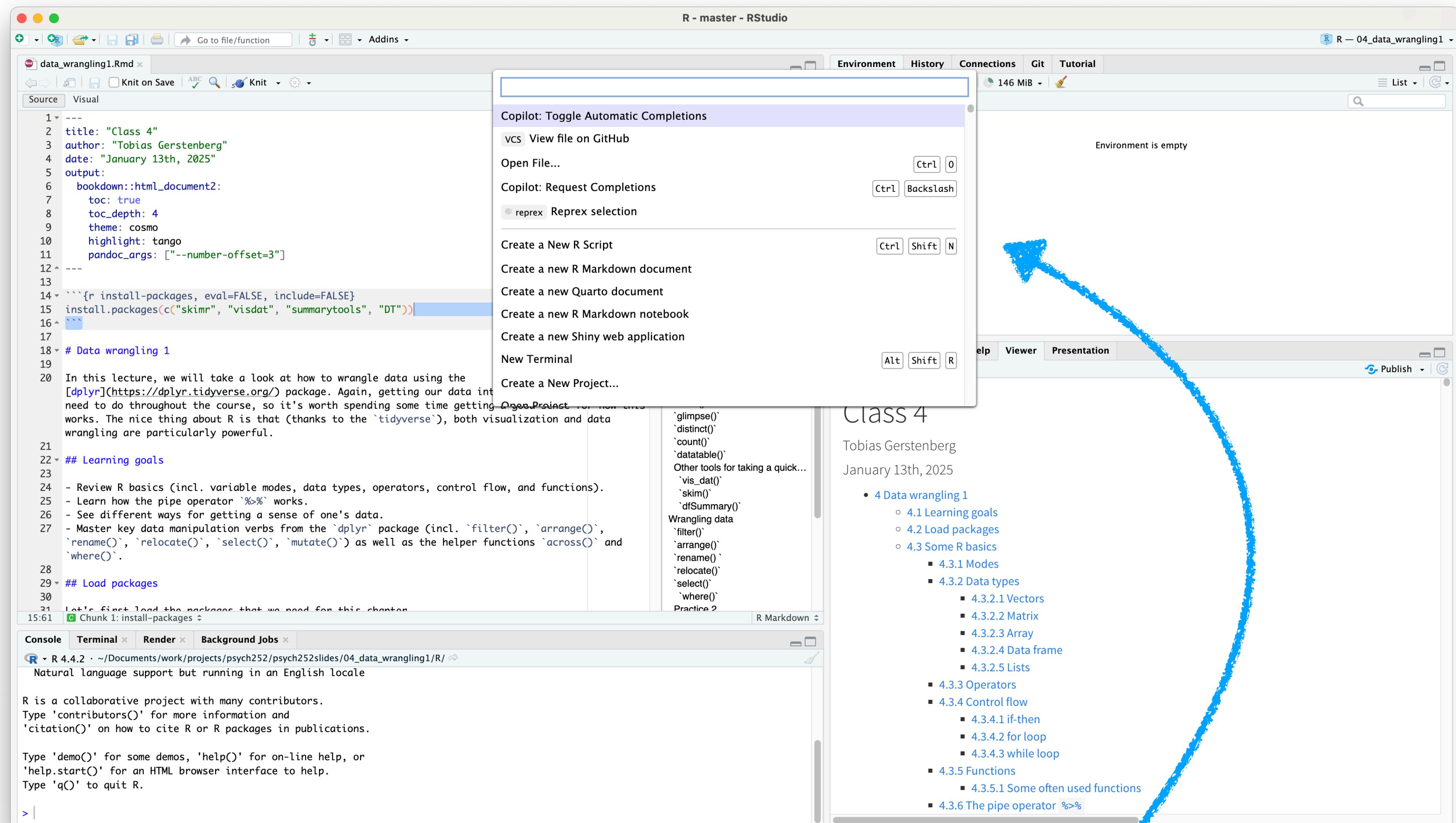


... and make
your own

A screenshot of the 'Modify Keyboard Shortcuts...' dialog box from RStudio. The title bar says 'Keyboard Shortcuts'. The left sidebar shows the 'Tools' menu with 'Modify Keyboard Shortcuts...' selected. The main area is a table titled 'Keyboard Shortcuts' with columns for 'Name', 'Shortcut', and 'Scope'. The table lists numerous keyboard shortcuts, many of which are highlighted in light blue. The 'Scope' column indicates that most shortcuts apply to 'Workbench'. At the bottom of the dialog are buttons for 'Reset...', 'Apply', and 'Cancel'.

Name	Shortcut	Scope
Browse Addins		Workbench
Check Spelling	F7	Workbench
Check for RStudio Updates		Workbench
Clear All Breakpoints...		Workbench
Clear All Plots...		Workbench
Clear Console	Ctrl+L	Workbench
Clear Knitr Cache		Workbench
Clear Prerendered Output		Workbench
Clear Terminal Buffer		Workbench
Clear Workspace		Workbench
Close All Documents		Workbench
Close Current Document	Cmd+W	Workbench
Close Current Project		Workbench
Close Other Documents	Shift+Alt+Cmd+W	Workbench
Close Terminal		Workbench
Compile Notebook	Shift+Cmd+K	Workbench
Console on Left		Workbench
Console on Right		Workbench
Copy Current Plot to Clipboard...		Workbench

Learn the keyboard shortcuts!

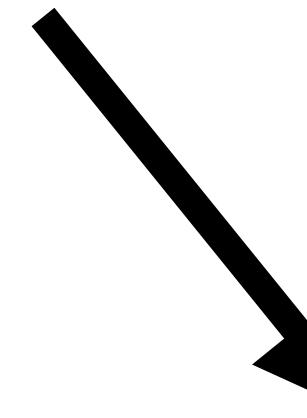


cmd + shift + p

Reformatting code

```
1 ggplot(data = df.diamonds[1:150], mapping = aes(x = color, y = price)) +  
2   # individual data points (jittered horizontally)  
3   geom_point(alpha = 0.2,  
4             position = position_jitter(width = 0.1, height = 0),  
5             size = 2) +  
6   # error bars  
7   stat_summary(fun.data = "mean_cl_boot",  
8               geom = "linerange",  
9               color = "black",  
10              size = 1) +  
11   # means  
12  stat_summary(fun.y = "mean",  
13               geom = "point",  
14               shape = 21,  
15               fill = "red",  
16               color = "black",  
17               size = 4)
```

highlight code and press
cmd + i



```
1 ggplot(data = df.diamonds[1:150], mapping = aes(x = color, y = price)) +  
2   # individual data points (jittered horizontally)  
3   geom_point(alpha = 0.2,  
4             position = position_jitter(width = 0.1, height = 0),  
5             size = 2) +  
6   # error bars  
7   stat_summary(fun.data = "mean_cl_boot",  
8               geom = "linerange",  
9               color = "black",  
10              size = 1) +  
11   # means  
12  stat_summary(fun.y = "mean",  
13               geom = "point",  
14               shape = 21,  
15               fill = "red",  
16               color = "black",  
17               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")
```

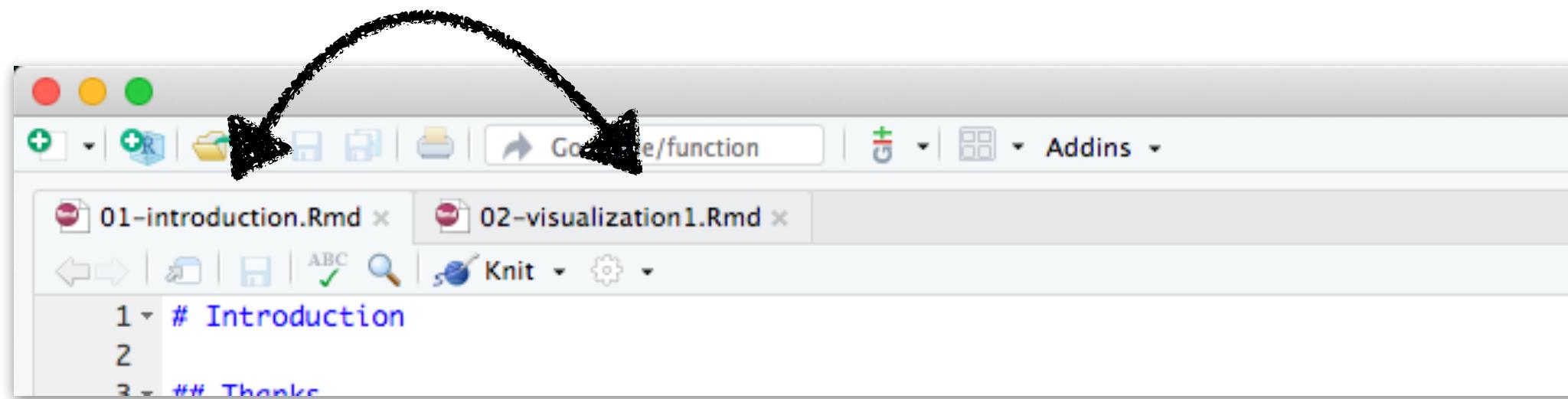


comment
cmd + shift + c

```
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

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

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 ````  
203
```



Keyboard Shortcuts		
Show:	All	Customized
<input type="text"/> chunk	<input type="button" value=""/>	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

`?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

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

Vector Functions

<code>sort(x)</code>	<code>rev(x)</code>
Return x sorted.	Return x reversed.
<code>table(x)</code>	<code>unique(x)</code>
See counts of values.	See unique values.

Selecting Vector Elements

By Position

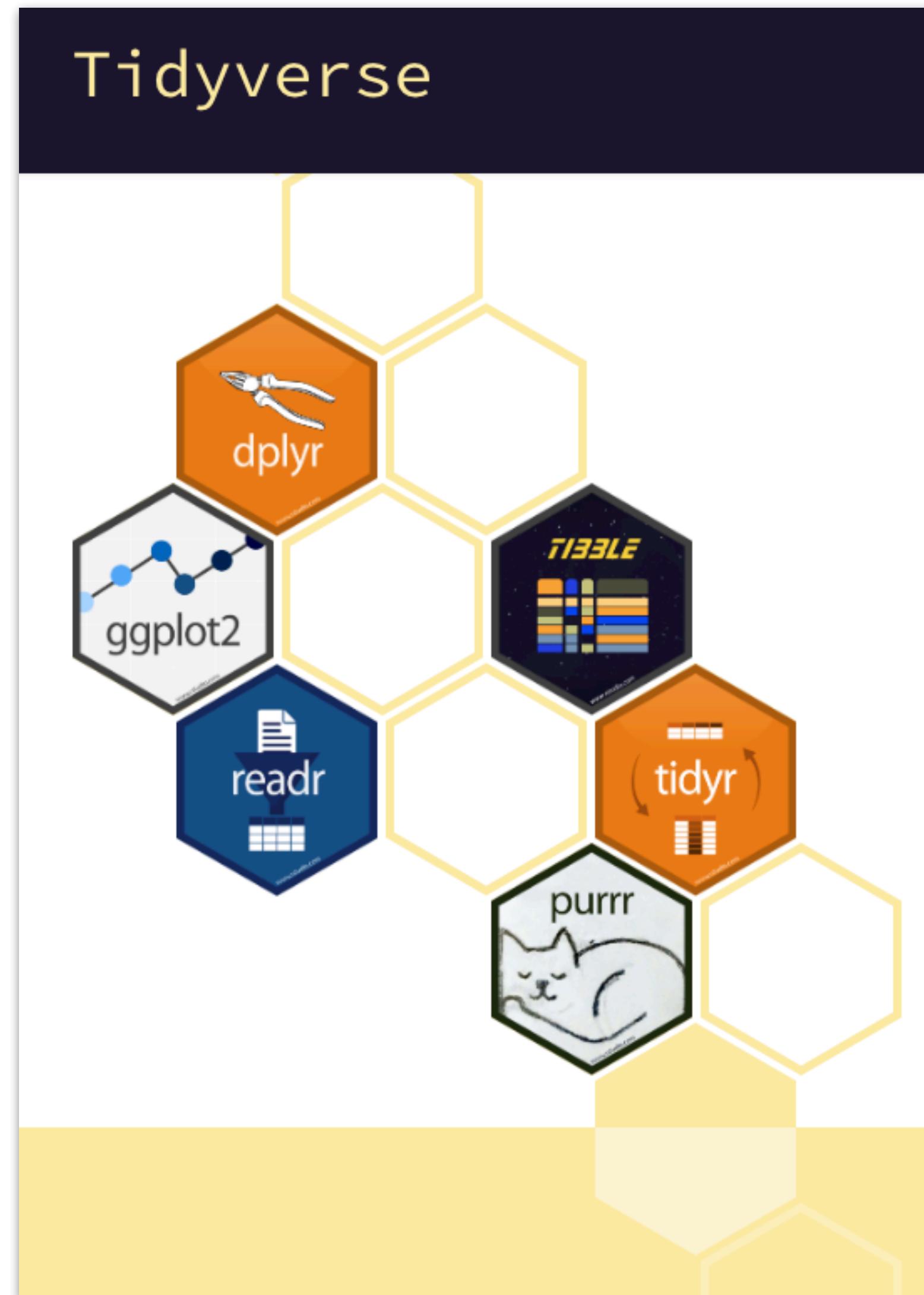
<code>x[4]</code>	The fourth element.
<code>x[-4]</code>	All but the fourth.
<code>x[2:4]</code>	Elements two to four.
<code>x[-(2:4)]</code>	All elements except two to four.
<code>x[c(1, 5)]</code>	Elements one and five.

By Value

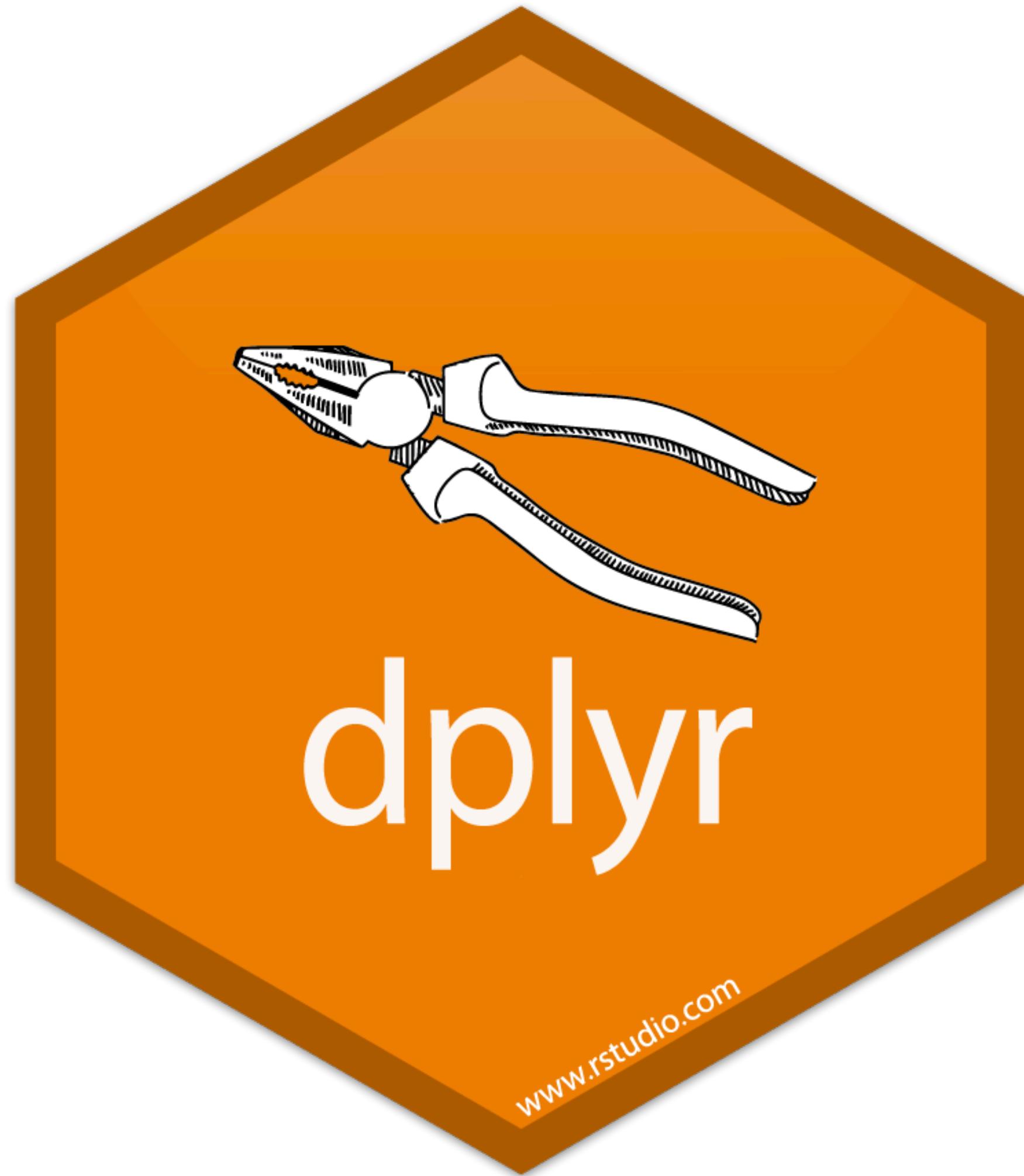
<code>x[x == 10]</code>	Elements which are equal to 10.
<code>x[x < 0]</code>	All elements less than zero.
<code>x[x %in% c(1, 2, 5)]</code>	Elements in the set 1, 2, 5.

Named Vectors

<code>x['apple']</code>	Element with name 'apple'.
-------------------------	----------------------------







Making Data Wrangling Suck Less

RStudio time

STAR
WARS



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



Please help.

blue

I'm done.

pink

Feedback

How was the pace of today's class?

much
too
slow

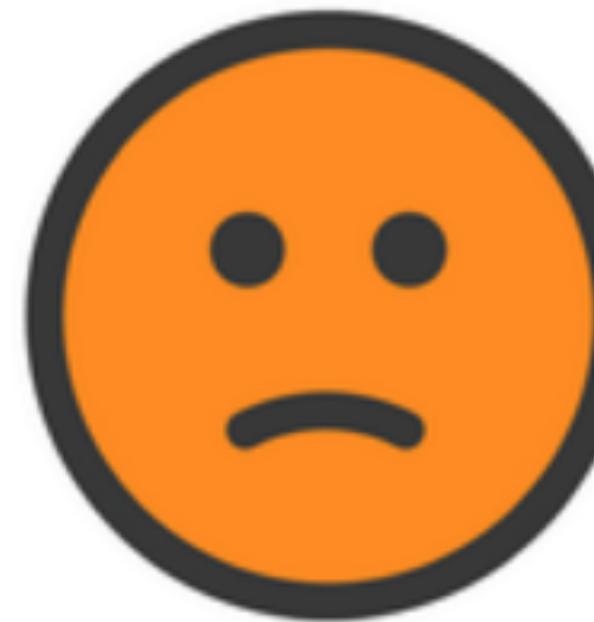
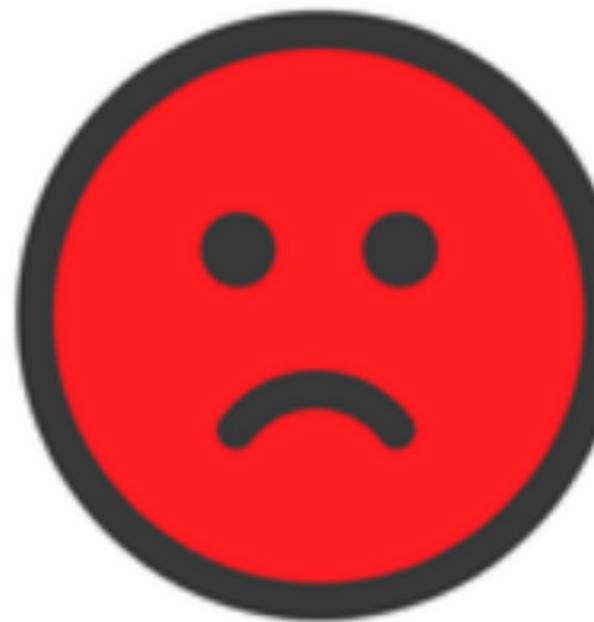
a little
too
slow

just
right

a little
too
fast

much
too
fast

How happy were you with today's class overall?



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

Thanks!