

# Data wrangling 2



COLLABORATIVE PLAYLIST

**psych252**

<https://tinyurl.com/psych252spotify>

PLAY ...

# Quick recap

# 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

# 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.

**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)	rev(x)
Return x sorted.	Return x reversed.
table(x)	unique(x)
See counts of values.	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'.
------------	----------------------------

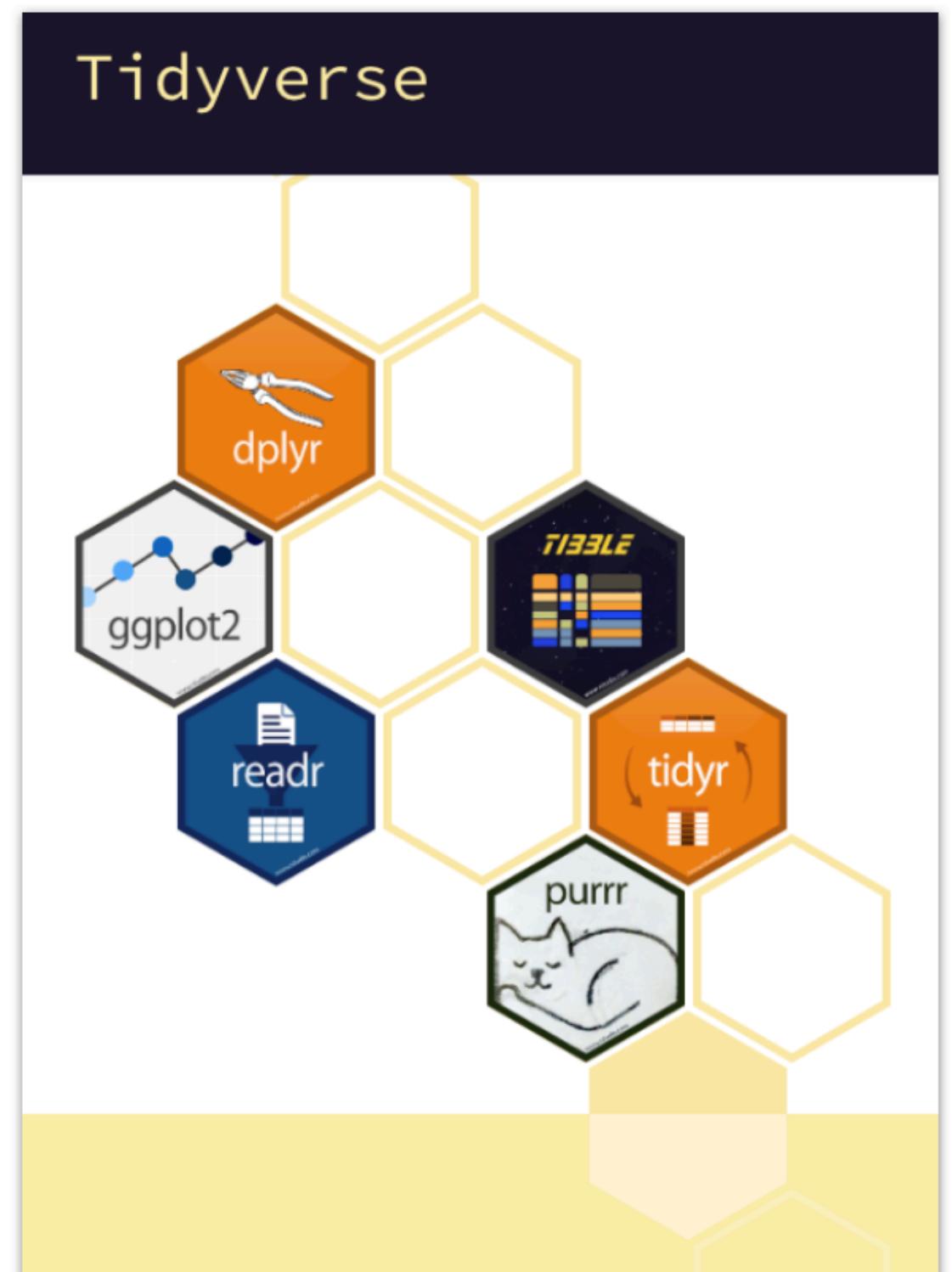
**Using Packages**

- 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.



### 4.3.3 Operators

Table 4.3 shows the comparison operators that result in logical outputs.

Table 4.3: Table of comparison operators that result in boolean (TRUE/FALSE) outputs.

symbol	name
<code>==</code>	equal to
<code>!=</code>	not equal to
<code>&gt; , &lt;</code>	greater/less than
<code>&gt;= , &lt;=</code>	greater/less than or equal
<code>&amp; ,   , !</code>	logical operators: and, or, not
<code>%in%</code>	checks whether an element is in an object

### 4.3.4 Control flow

#### 4.3.4.1 if-then

```
number = 3

if (number == 1) {
  print("The number is 1.")
} else if (number == 2) {
  print("The number is 2.")
} else {
  print("The number is neither 1 nor 2.")
}
```



```
eat(
  slice(
    bake(
      put(
        pour(
          mix(ingredients),
          into = baking_form),
        into = oven),
      time = 30),
      pieces = 6),
    1)
```

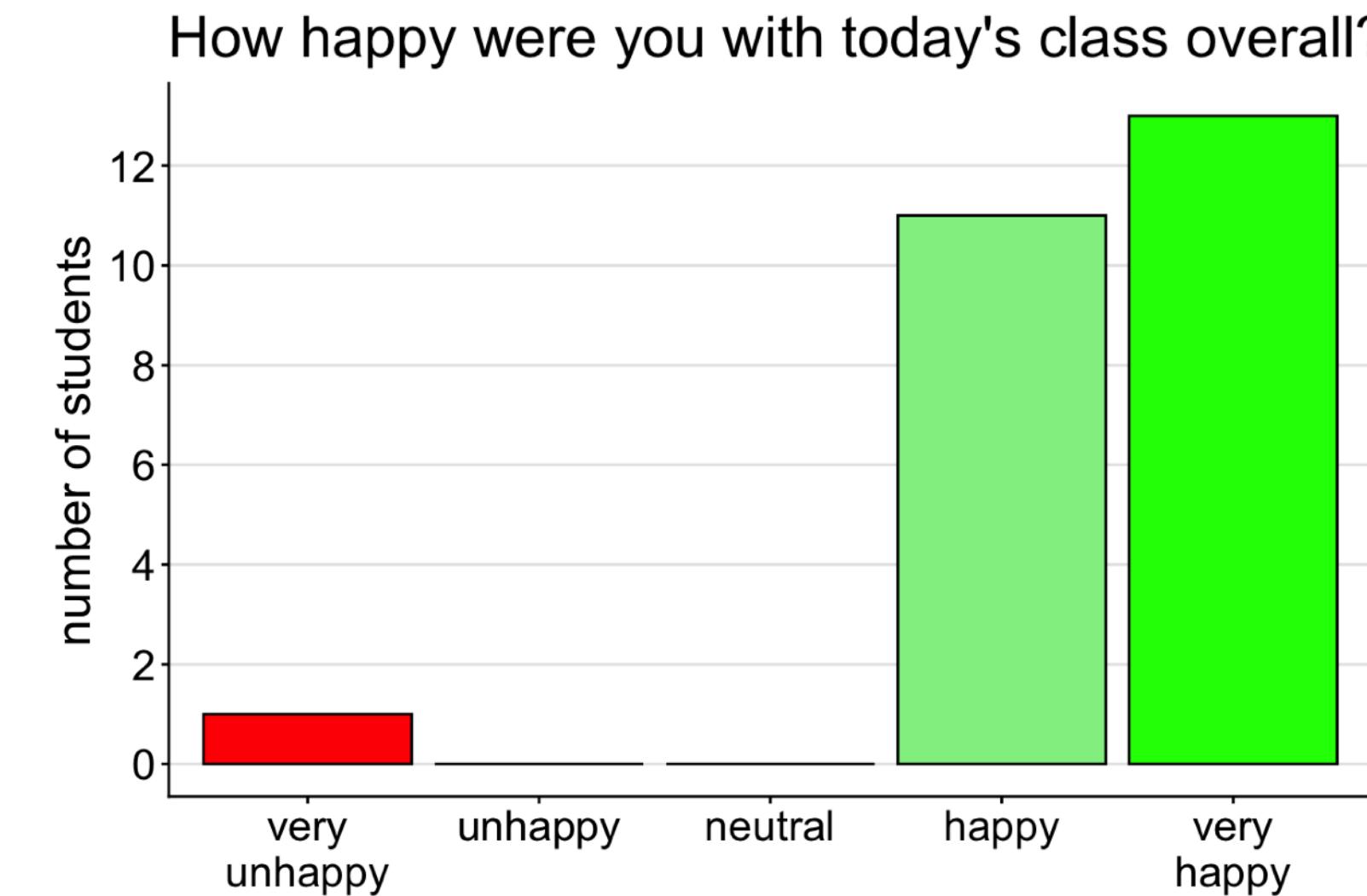
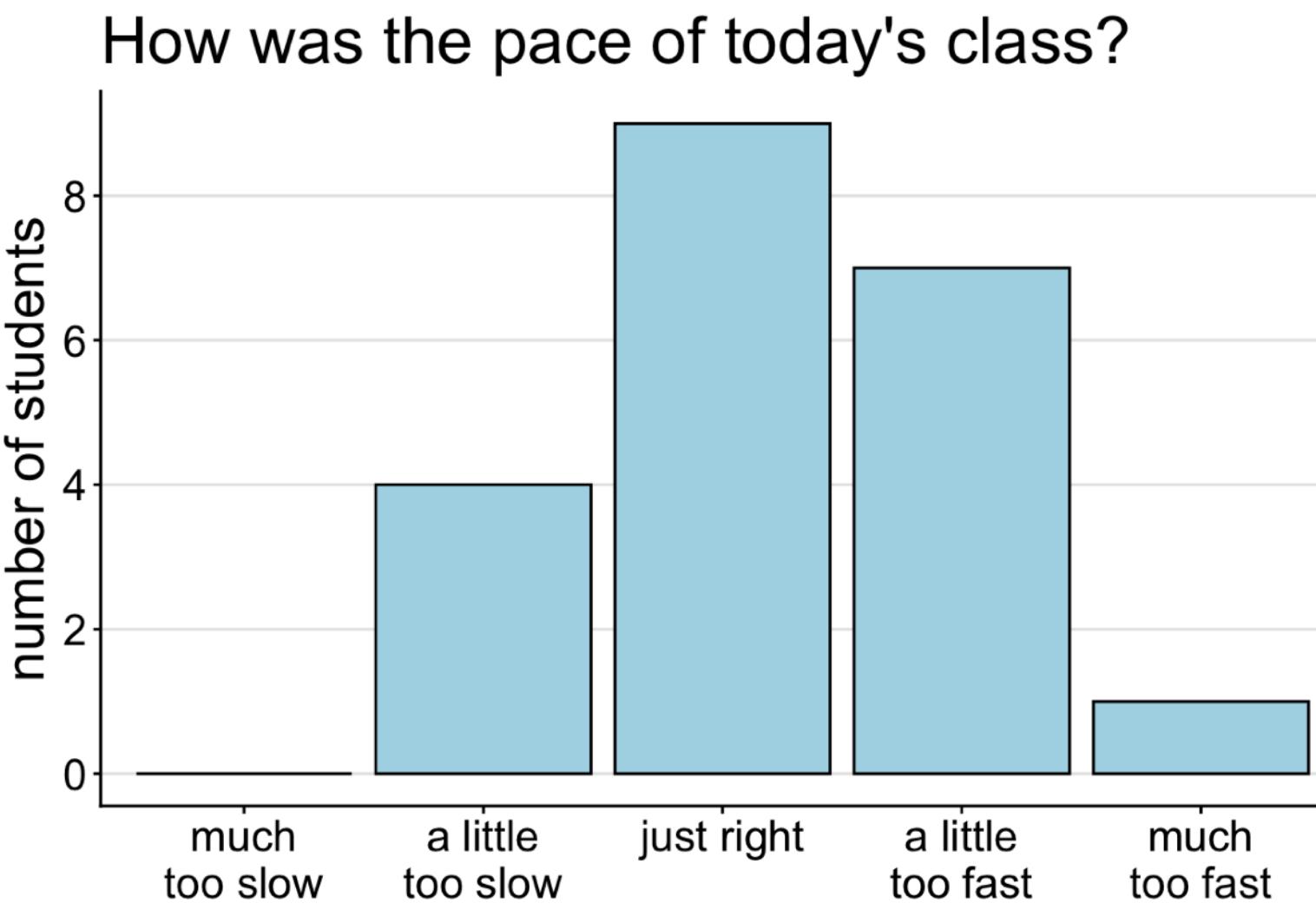
```
ingredients %>%
  mix %>%
  pour(into = baking_form) %>%
  put(into = oven) %>%
  bake(time = 30) %>%
  slice(pieces = 6) %>%
  eat(1)
```

### 4.6 Wrangling data

- 4.6.1 `filter()`
- 4.6.2 `arrange()`
- 4.6.3 `rename()`
- 4.6.4 `relocate()`
- 4.6.5 `select()`
- 4.6.5.1 `where()`

# Your feedback

# Your feedback



Really helpful! Could you please **remind us what the shortcuts are as you use them?** I'm grateful to learn them but need reminders.

I appreciate having the instructors code the examples and giving us space to try things on our own. I think **group coding might also be helpful** for examples that are a little more complicated. I am roughly familiar with R, but I think if I had less familiarity the groundwork information would have felt very fast because with my base it felt a little fast

What would be helpful is after we go through the solution for the practice, **leaving it up for a while to compare where I went wrong.**

# Data wrangling time ...

dplyr : go wrangling



# Tidy data

“**TIDY DATA** is a standard way of mapping the meaning of a dataset to its structure.”

-HADLEY WICKHAM

## In tidy data:

- each variable forms a column
- each observation forms a row
- each cell is a single measurement

each column a variable

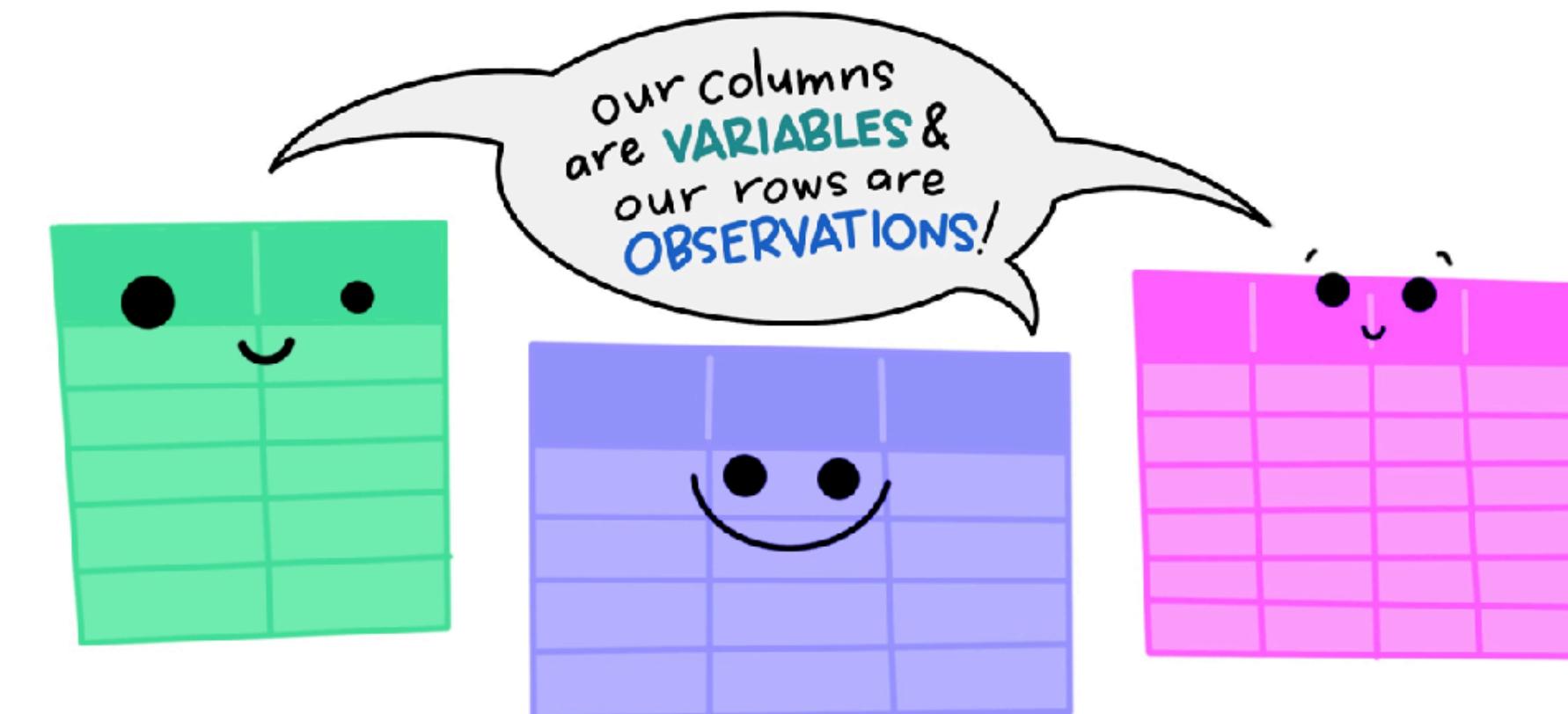
each row an observation

id	name	color
1	floof	gray
2	max	black
3	cat	orange
4	donut	gray
5	merlin	black
6	panda	calico

Wickham, H. (2014). Tidy Data. Journal of Statistical Software 59 (10). DOI: 10.18637/jss.v059.i10

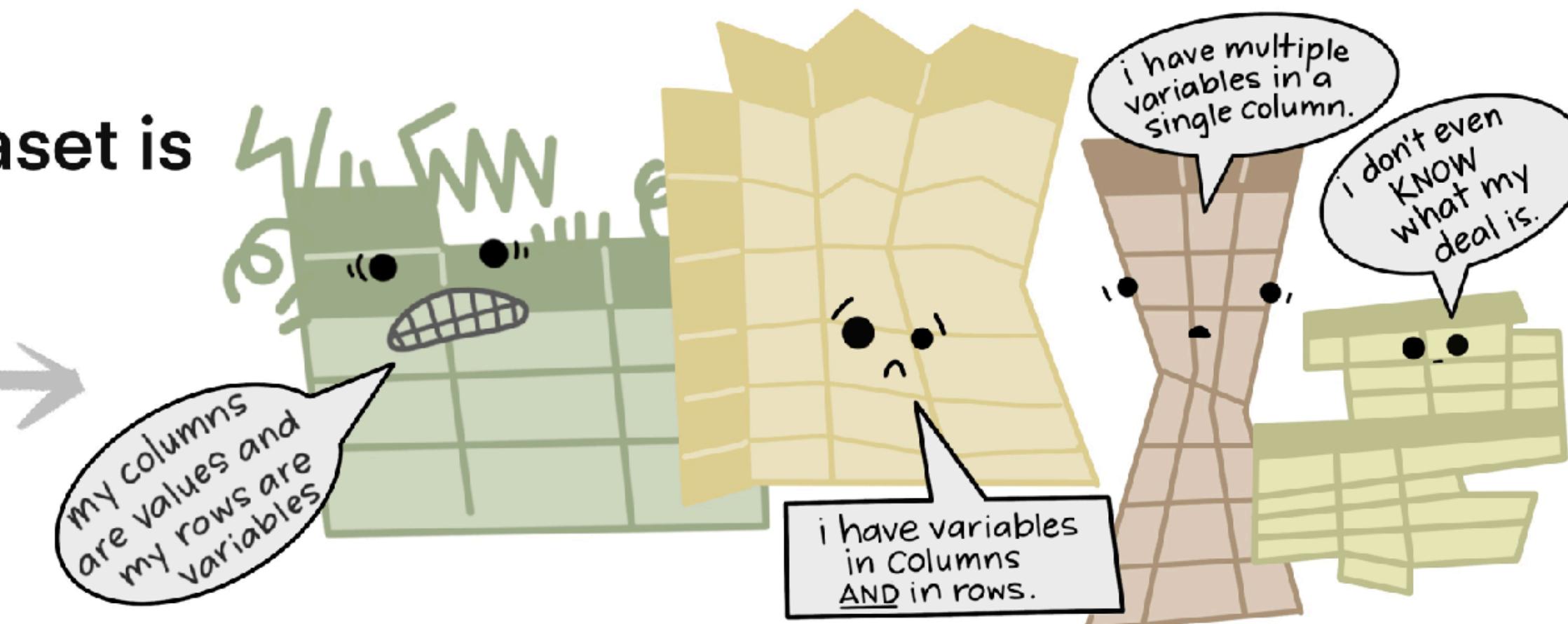
# Tidy data

The standard structure of tidy data means that  
“tidy datasets are all alike...”



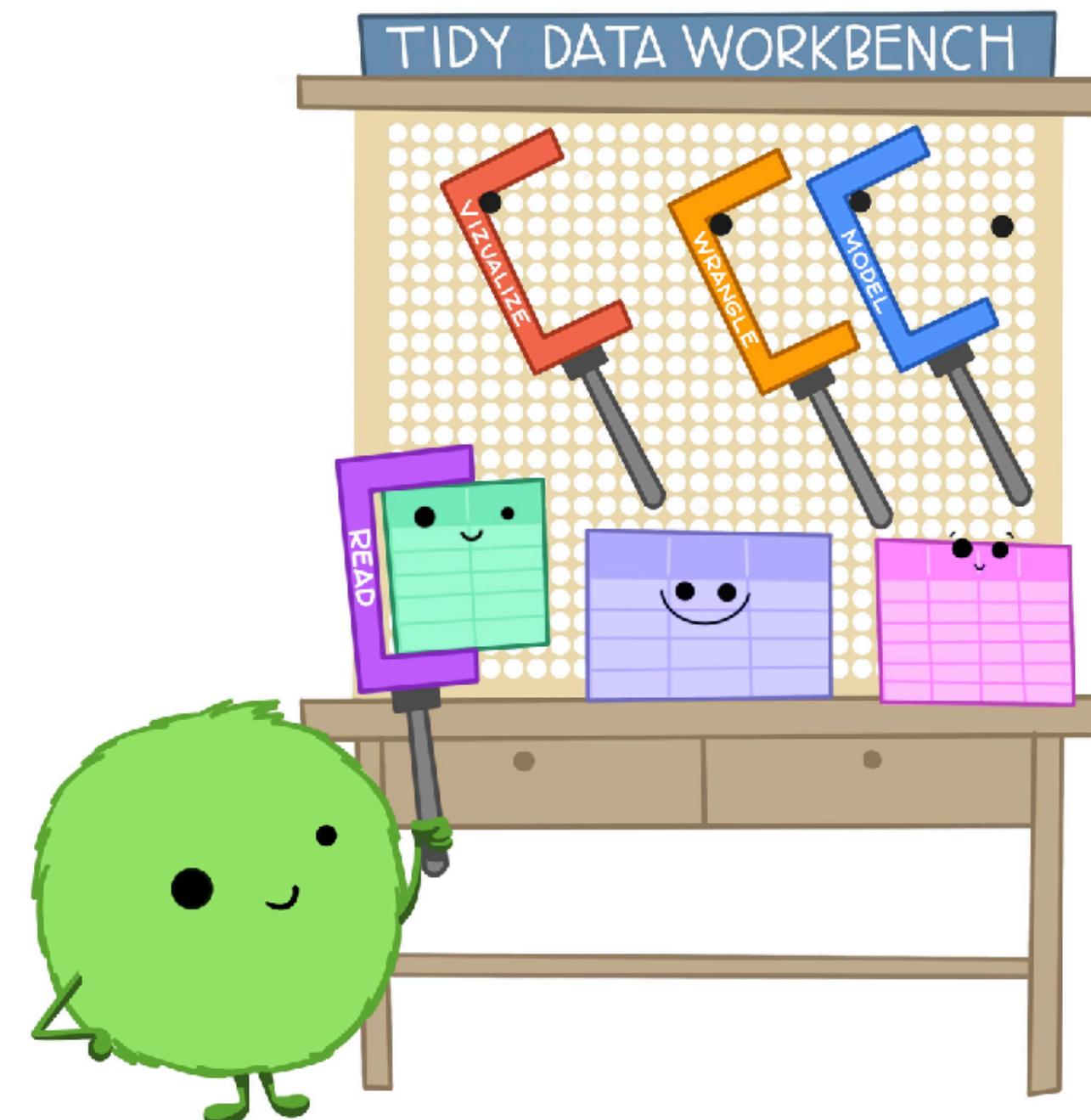
“...but every messy dataset is  
messy in its own way.”

-HADLEY WICKHAM

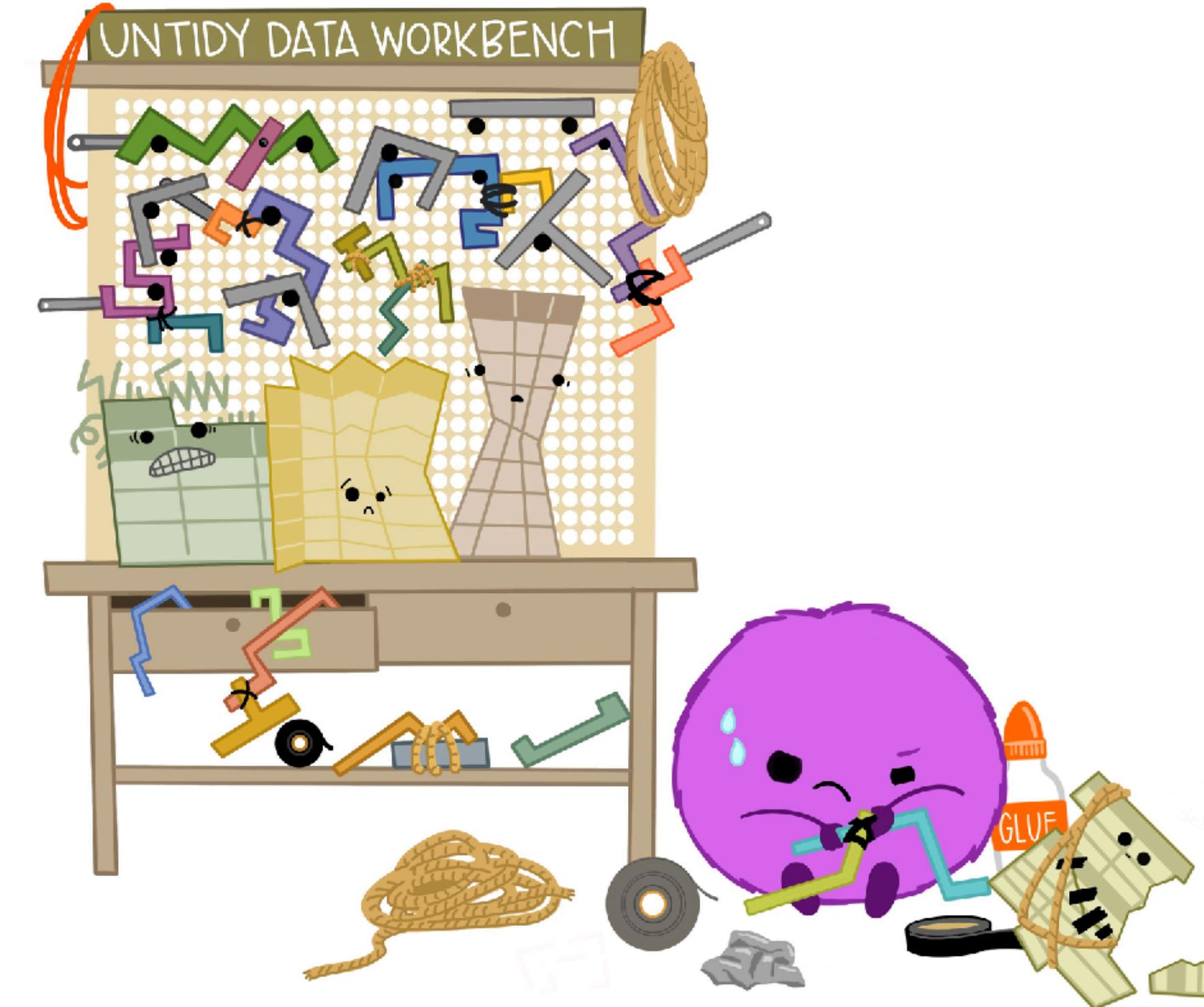


# Tidy data

When working with tidy data,  
we can use the **same tools** in  
**similar ways** for different datasets...



...but working with untidy data often means  
reinventing the wheel with **one-time**  
**approaches** that are **hard to iterate or reuse**.



# Using functions with tidyverse verbs

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

my  
recommendation

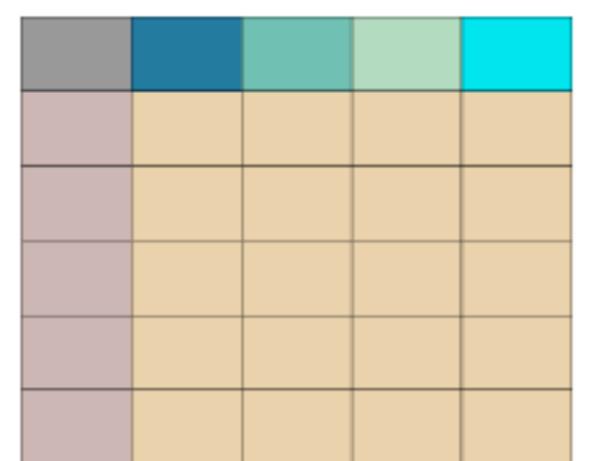
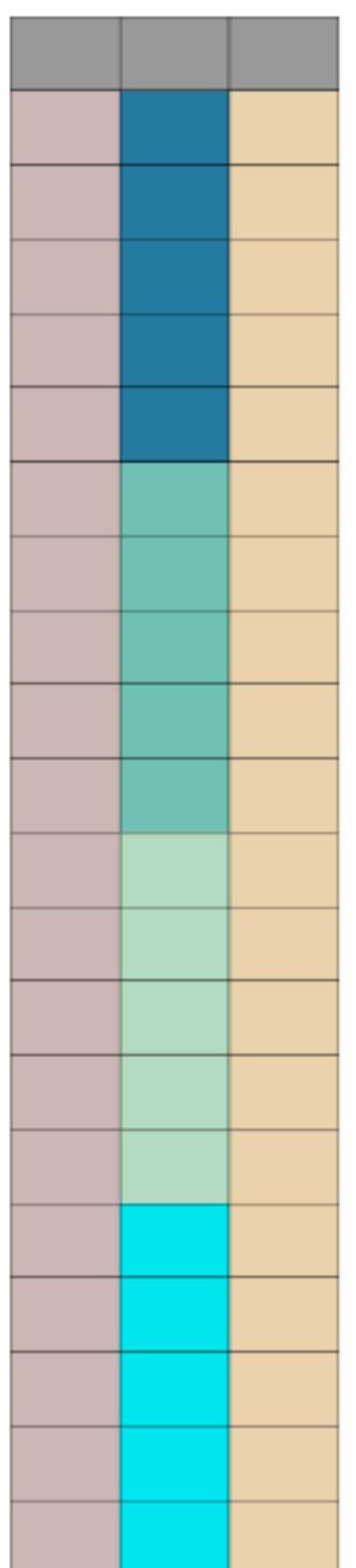
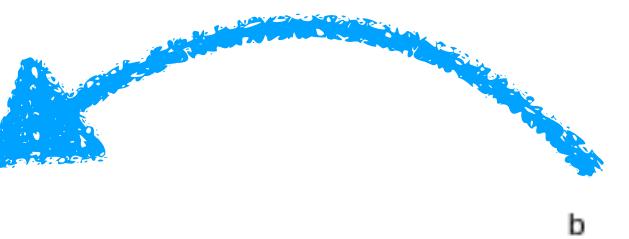
- fn = is.numeric
- fn = "is.numeric"
- fn = function(x) {is.numeric(x)}
- fn = ~ is.numeric(.)

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

- fn = ~ !is.numeric(.)

select all  
variables that  
are not numeric

# pivot\_longer()



- Group 1
- Group 2
- Group 3
- Group 4
- Data
- Header
- ID

# pivot\_wider()



# pivot\_longer() & pivot\_wider()

wide

id	x	y	z
1	a	c	e
2	b	d	f

# left\_join()

left\_join(x, y)

1	x1	1	y1
2	x2	2	y2
3	x3	4	y4

# left\_join()

left\_join(x, y)

1	x1	1	y1
2	x2	2	y2
3	x3	4	y4
		2	y5

# **Timer**



Please help.

**blue**

I'm done.

**pink**

# **Feedback**

# How was the pace of today's class?

much  
too  
slow

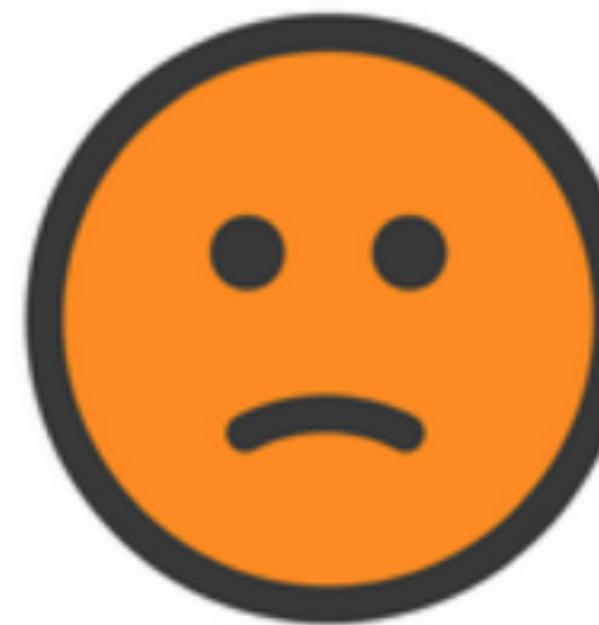
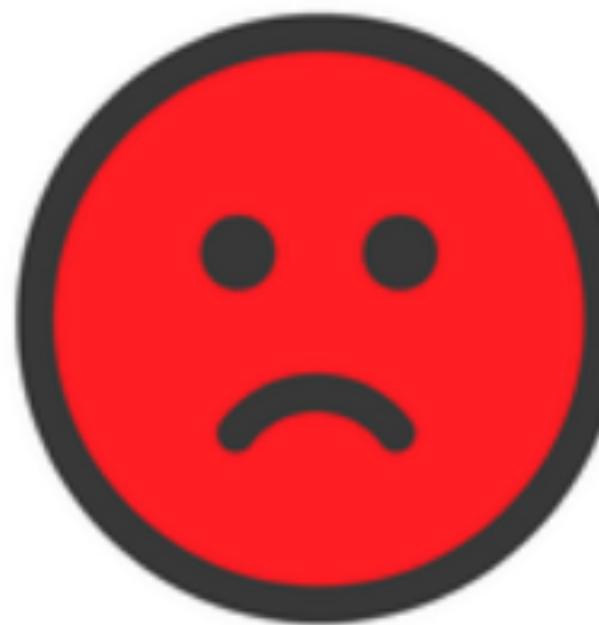
a little  
too  
slow

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?**

**Thank you!**