



THE UNIVERSITY
of EDINBURGH | **U**usher
institute

HDS Tutorial 4

Week 8



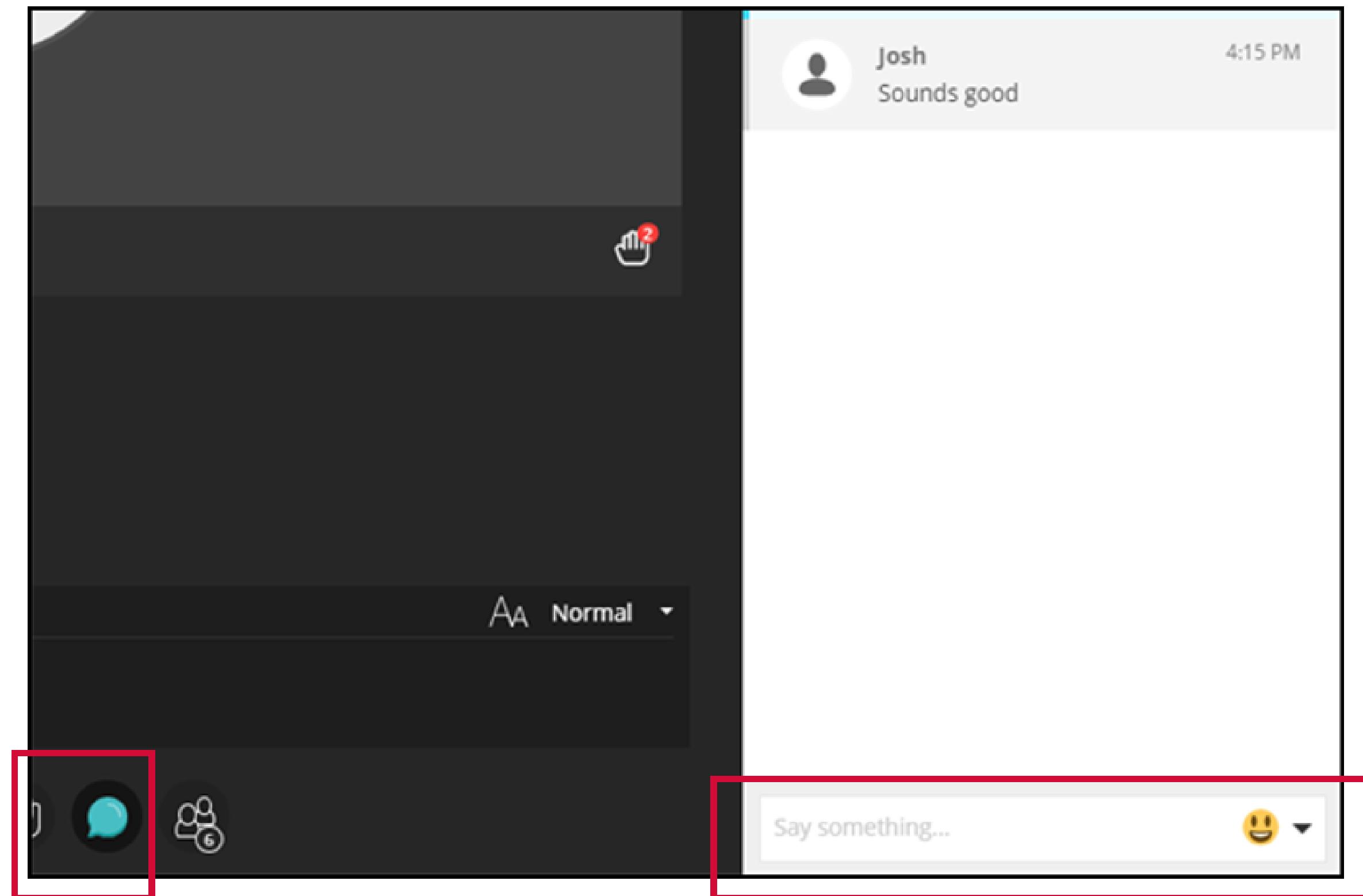
Audio check

Can you hear the presenter talking?

Please type **yes** or **no** in the “Text chat area”

If you can't hear:

- Check your Audio/Visual settings in the Collaborate Panel
- Try signing out and signing back into the session
- Type into the chat box and a moderator will try to assist you





THE UNIVERSITY
of EDINBURGH

| Uisher
Institute

Recording

Open to
the world

This session will now be recorded. Any further information that you provide during a session is optional and in doing so you give us consent to process this information.

These sessions will be stored by the University of Edinburgh for one year and published for 30 days after the event. Schools or Services may use the recordings for up to a year on relevant websites.

By taking part in a session, you give us your consent to process any information you provide during it.

Start Recording

ddi.hsc.talent@ed.ac.uk

Supported by



THE UNIVERSITY
of EDINBURGH

Data-Driven
Innovation



THE UNIVERSITY
of EDINBURGH | **U**usher
institute

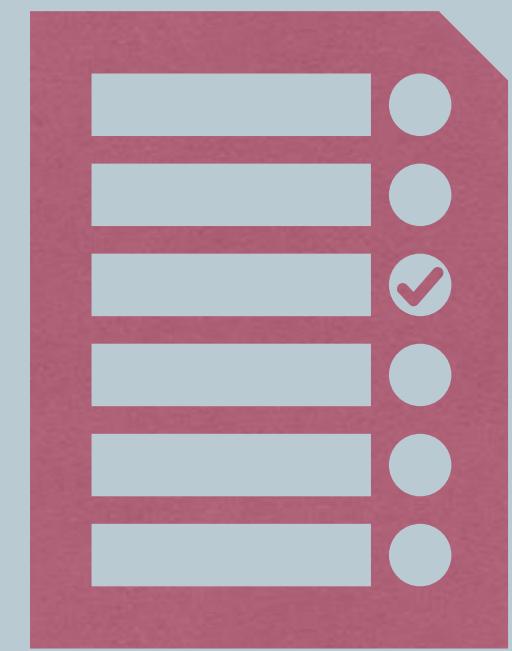
HDS Tutorial 4

Week 8

Agenda



- **Interactive overview of key functions discussed in the course**
- **R Markdown knit to PDF demo**
- **Q&A**



Have you managed to successfully
knit an R Markdown document to
PDF?

Summary of key functions





Importing Data

★ `read_csv()`

```
#from a webpage
activity_raw <- read_csv("https://www.opendata.nhs.scot/dataset/0e17f3fc-9429-48aa-b1ba-2b7e55688253
/resource/748e2065-b447-4b75-99bd-f17f26f3eaef/download/hd_activitybyhbr.csv")

#from a saved file on your computer in a folder called data
mortality_raw <- read_csv(here::here("./data/heartdiseaseMortalitybyHB.csv"))

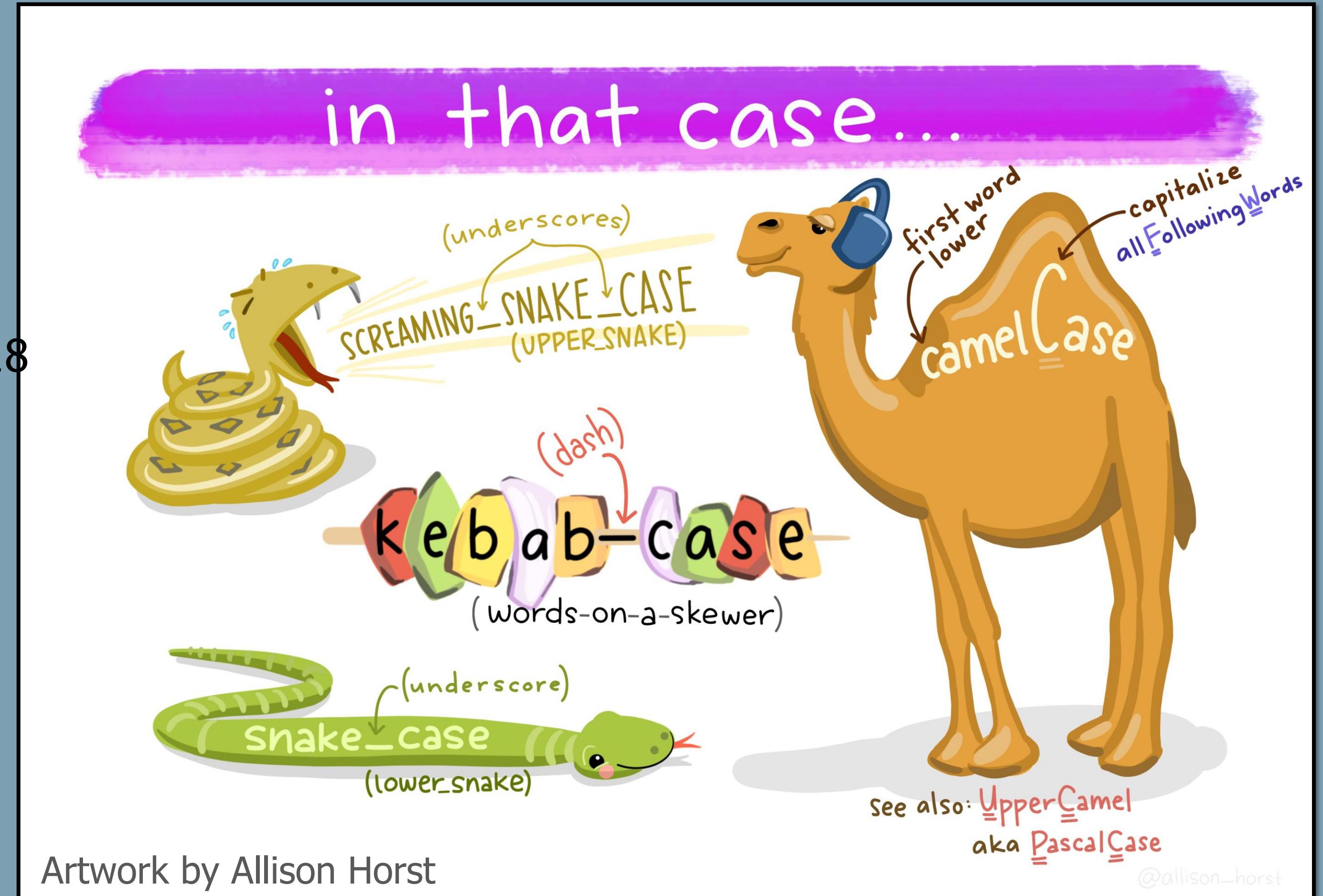
#or without the here package
mortality_raw <- read_csv("./data/heartdiseaseMortalitybyHB.csv")
```



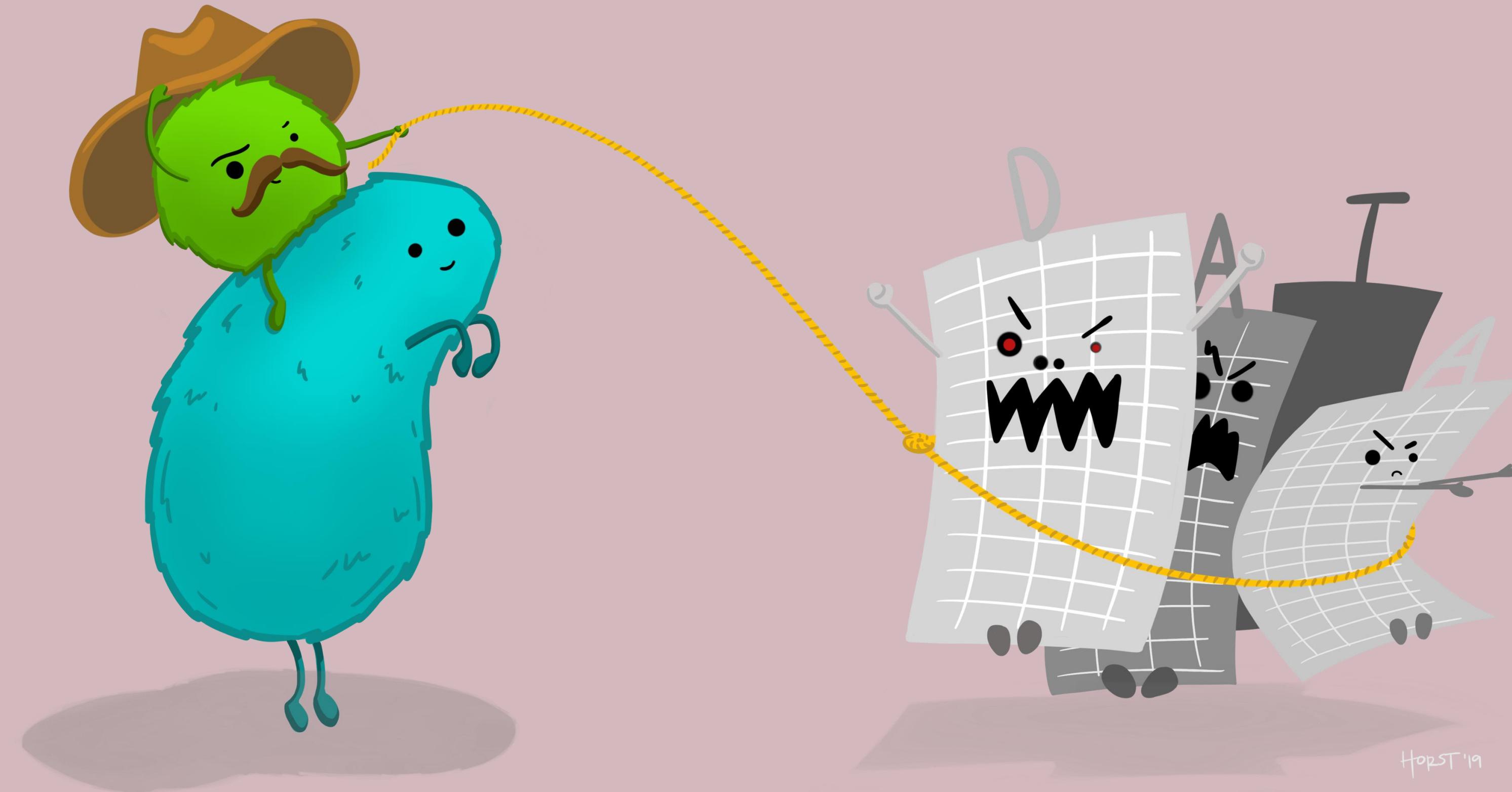
Variable naming conventions

✿ clean_names()

Defaults to snake case, but there are 18 options you can choose from



Wrangling



Artwork by Allison Horst

Logical Operators

<

Less than

>

Greater than

==

Equal to

<=

Less than equal to

>=

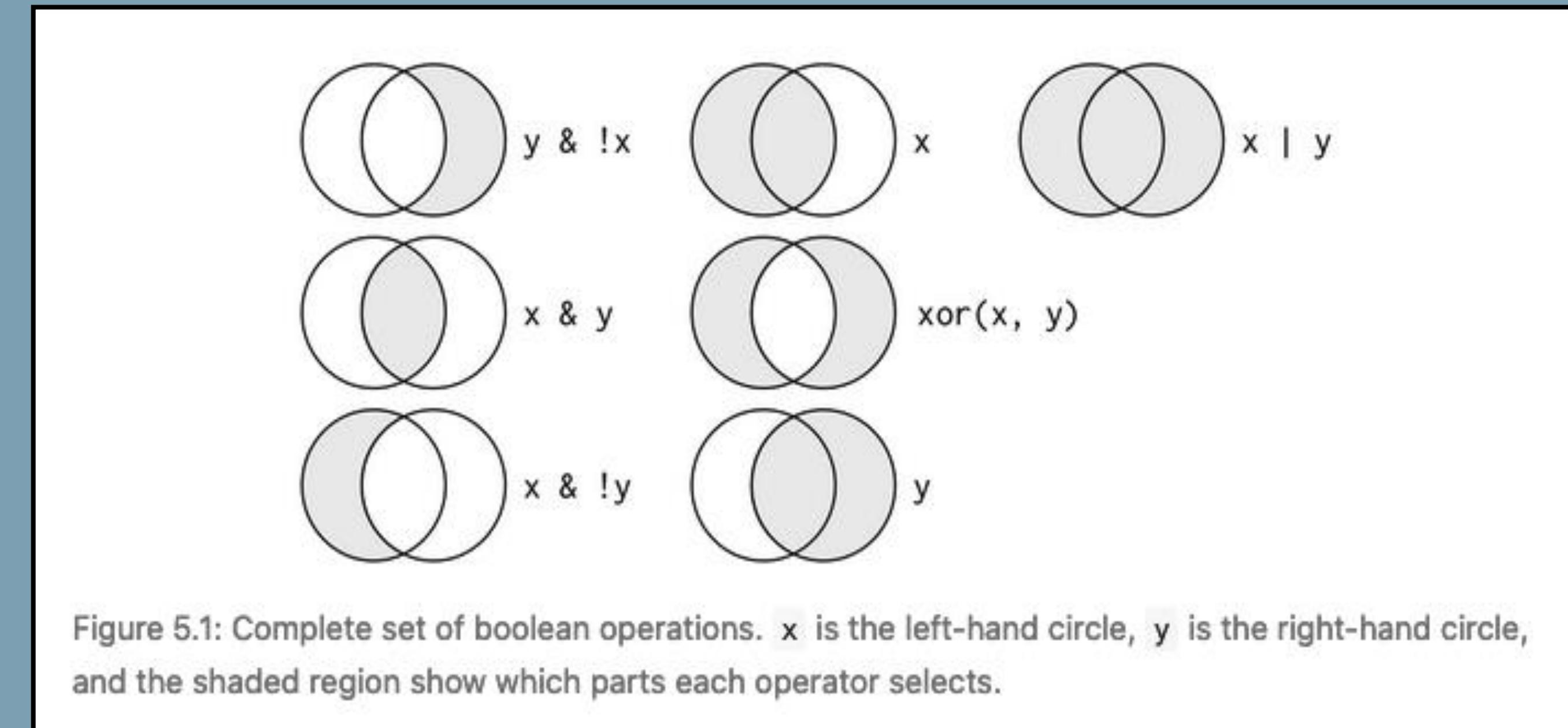
Greater than equal to

!=

Not equal to

%in%

Group membership



Source: R for Data Science book, Figure 5.1



Subsetting Data (observations)

 filter()

Extract rows of existing data
that meeting logical
conditions

data.frame

logical
test

filter(surveys, year >= 1985)

data.frame

logical
test

surveys %>% filter(year >= 1985)

Credit: [Coding TogetheR](#)

For more check out this RStudio Data Manipulation video from Garrett Grolemund https://www.youtube.com/watch?v=Zc_ufg4uW4U



Subsetting Data (variables)

★ select()

Select columns by name or helper functions

Helper functions for select - ?select

```
select(iris, contains("."))
```

Select columns whose name contains a character string.

```
select(iris, ends_with("Length"))
```

Select columns whose name ends with a character string.

```
select(iris, everything())
```

Select every column.

```
select(iris, matches(".t."))
```

Select columns whose name matches a regular expression.

```
select(iris, num_range("x", 1:5))
```

Select columns named x1, x2, x3, x4, x5.

```
select(iris, one_of(c("Species", "Genus")))
```

Select columns whose names are in a group of names.

```
select(iris, starts_with("Sepal"))
```

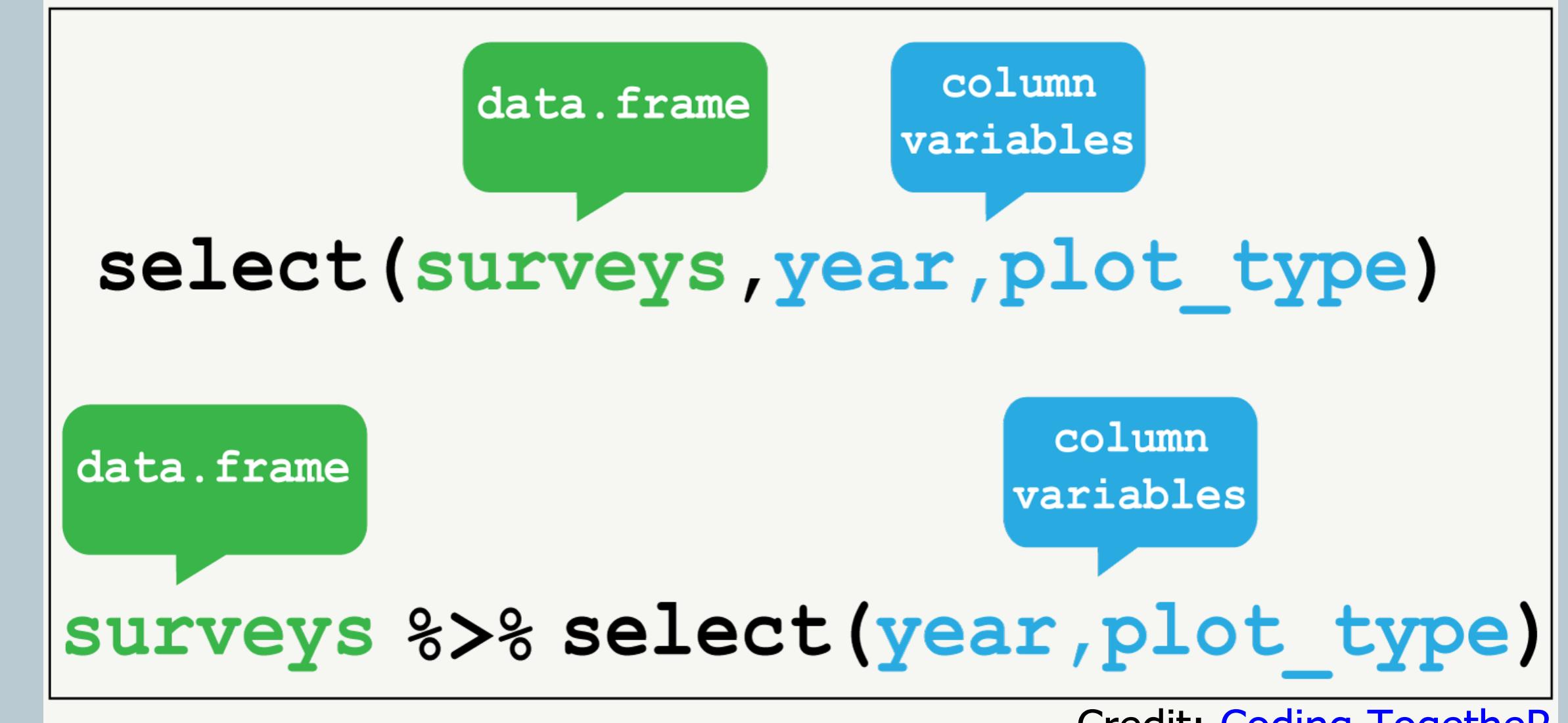
Select columns whose name starts with a character string.

```
select(iris, Sepal.Length:Petal.Width)
```

Select all columns between Sepal.Length and Petal.Width (inclusive).

```
select(iris, -Species)
```

Select all columns except Species.



Credit: [Coding TogetheR](#)



New Variables

✿ mutate()

Compute and append one or more new columns – changes an existing column or adds a new one

Works with grouped data or the whole dataset

Original columns remain after being passed to mutate

data.frame

new column variable

expression

```
mutate(surveys, weight_kg = weight/1000)
```

data.frame

new column variable

expression

```
surveys %>% mutate(weight_kg = weight/1000)
```

Credit: [Coding TogetheR](#)

For more check out this RStudio Data Manipulation video from Garrett Grolemund https://www.youtube.com/watch?v=Zc_ufg4uW4U



Summarise

★ summarise()

Summarise data into single row of values

Drops variables not listed in group_by() or created inside it, drops columns after calculating the new one

```
data.frame          new column variable      expression  
summarise(surveys, mean_weight = mean(weight, na.rm = TRUE))  
  
data.frame          new column variable      expression  
surveys %>% summarise(mean_weight = mean(weight, na.rm = TRUE))
```

Credit: [Coding TogetheR](#)

For more check out this RStudio Data Manipulation video from Garrett Grolemund https://www.youtube.com/watch?v=Zc_ufg4uW4U



Group Data

★ group_by()

Group data into rows according to column variables

★ ungroup()

Remove grouping information from the data frame – particularly useful when wrangling data for tables

data.frame

column
variables

```
group_by(surveys, species_id, rodent_type) %>%  
  summarise(mean_weight = mean(weight, na.rm = TRUE))
```

data.frame

column
variables

```
surveys %>% group_by(species_id, rodent_type) %>%  
  summarise(mean_weight = mean(weight, na.rm = TRUE))
```

Credit: [Coding TogetheR](#)

For more check out this RStudio Data Manipulation video from Garrett Grolemund https://www.youtube.com/watch?v=Zc_ufg4uW4U

Data formats & 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

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

each row an observation

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

Ways data can become untidy:

- Column headers contain values, rather than names
- Multiple variables are stored in a single column
- Variables are stored in both rows and columns
- Multiple observational types are stored in a single table
- A single observational unit is stored in multiple tables

Wickham, H. (2014). Tidy data. *Journal of statistical software*, 59(1), 1-23.

For more on tidy data see the above paper & Chapter 12 of the R for Data Science Book

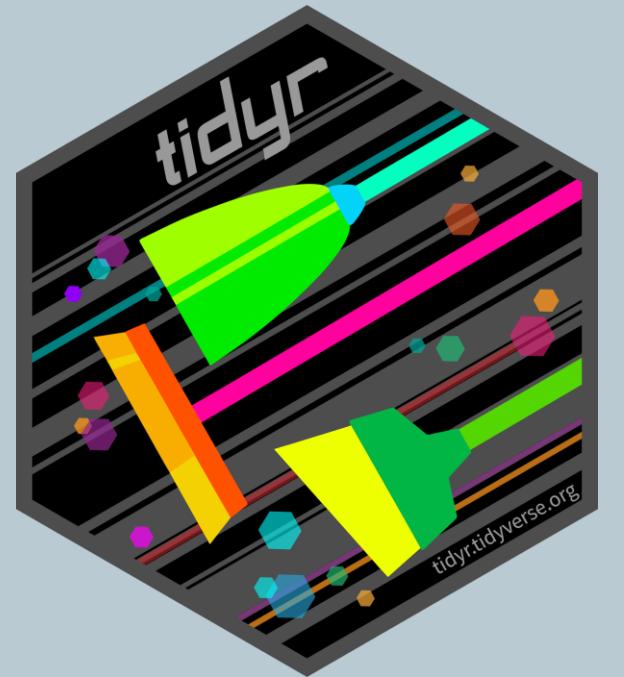
Data formats & Tidy Data

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

“Long” format		
country	year	metric
x	1960	10
x	1970	13
x	2010	15
y	1960	20
y	1970	23
y	2010	25
z	1960	30
z	1970	33
z	2010	35

“Wide” format			
country	yr1960	yr1970	yr2010
x	10	13	15
y	20	23	25
z	30	33	35

Wide format = generally untidy, but found in many datasets

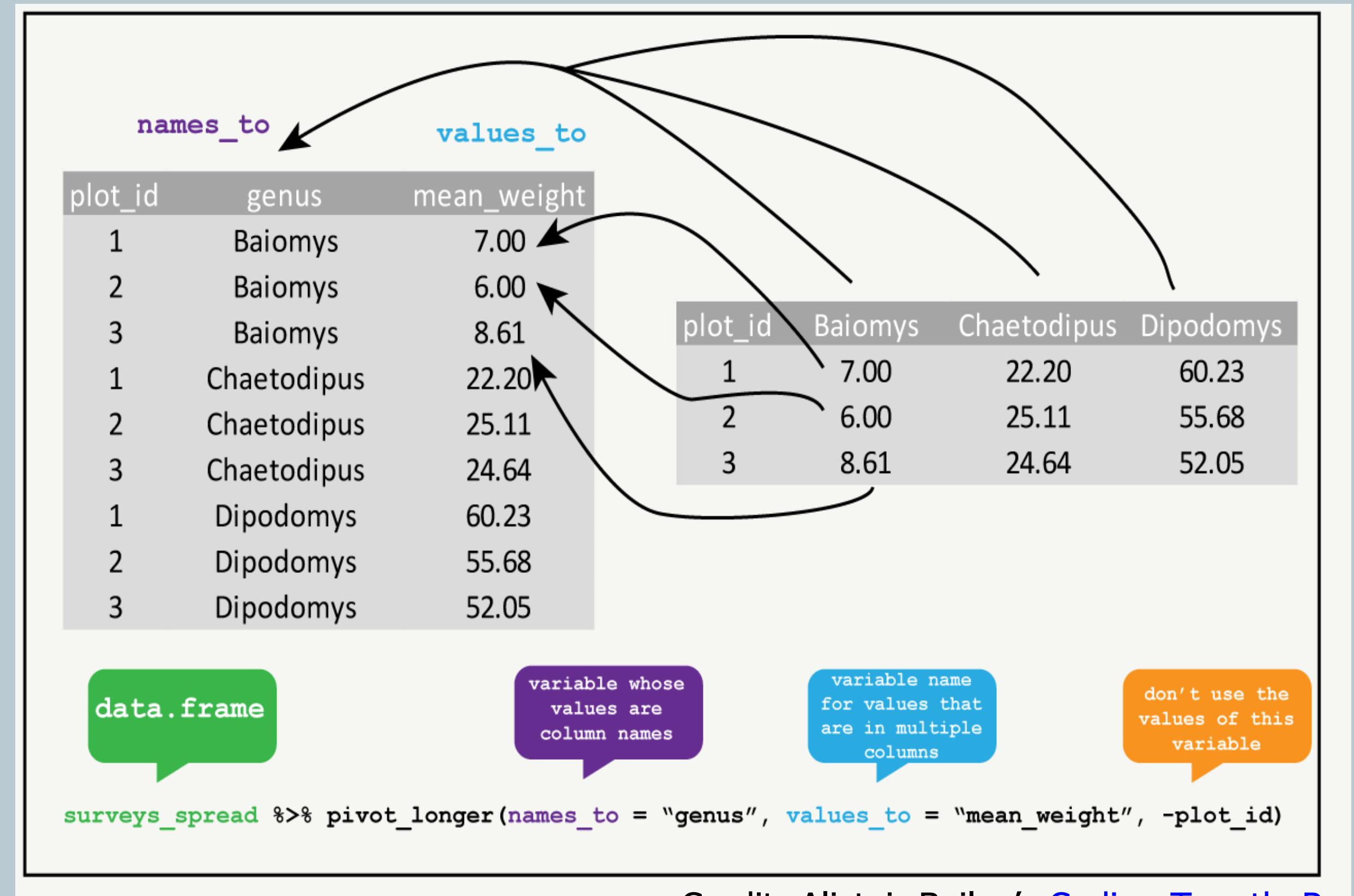


Transforming Data (wide to long)

✿ pivot_longer()

Requires:

1. data = data you want to pivot
2. names_to = name of the column you want to create to capture condition, requires a character string
3. values_to = name of column you want to contain data values, requires a character string
4. column X:column Y = range of columns that you have and want to pivot longer, or that you do not want to pivot



Credit: Alistair Bailey's [Coding TogetheR](#)



Transforming Data

★ `pivot_longer()` = wide to long

country	1999	2000	2001	2002
Angola	800	750	925	1020
India	20100	25650	26800	27255
Mongolia	450	512	510	586

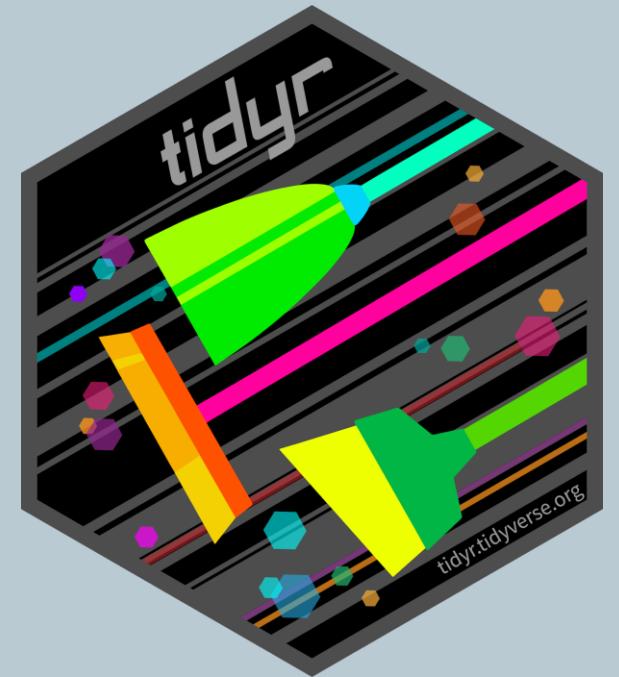
Pivot data longer

```
data %>%
  pivot_longer(
    cols = 1999:2002,
    names_to = "year",
    values_to = "cases"
  )
```



country	year	cases
Angola	1999	800
Angola	2000	750
Angola	2001	925
Angola	2002	1020
India	1999	20100
India	2000	25650
India	2001	26800
India	2002	27255
Mongolia	1999	450
Mongolia	2000	512
Mongolia	2001	510
Mongolia	2002	586

Credit: [Epidemiologist R Handbook](#)



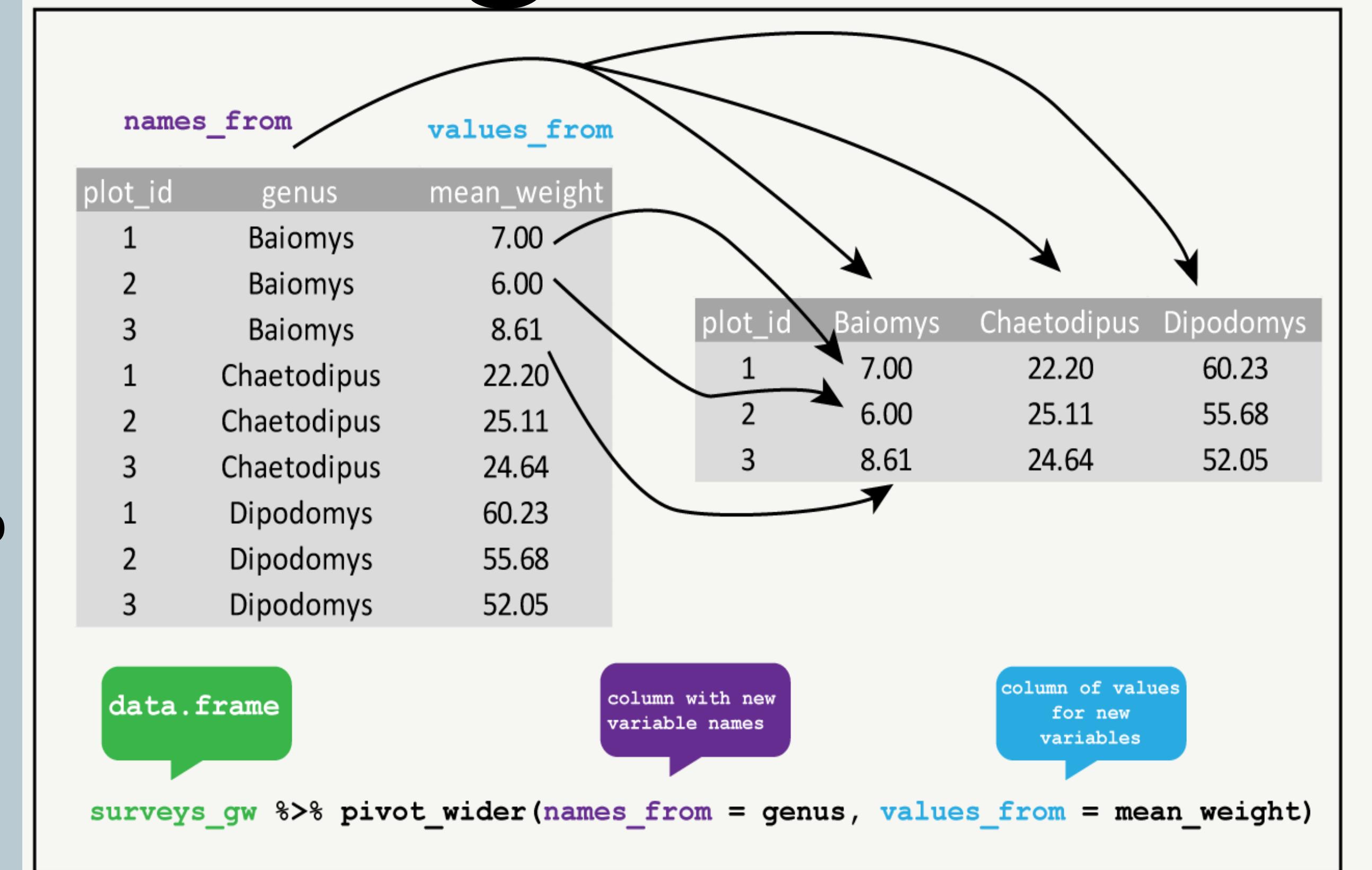
Transforming Data

(long to wide)

✿ pivot_wider()

Requires:

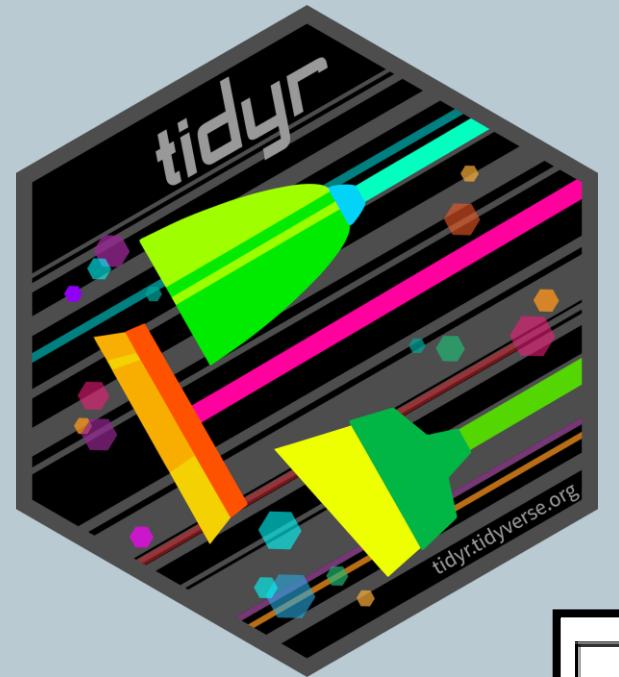
1. data = data you want to pivot
2. names_from = name of column you want to end up in several columns
3. values_from = name of column that currently contains data values



Credit: Alistair Bailey's [Coding TogetheR](#)

For more check out this RStudio Data Wrangling video from Garrett Grolemund

<https://www.youtube.com/watch?v=1ELALQIO-yM> - however includes the now superseded functions `gather()` & `spread()`



Transforming Data

✿ pivot_wider() = long to wide

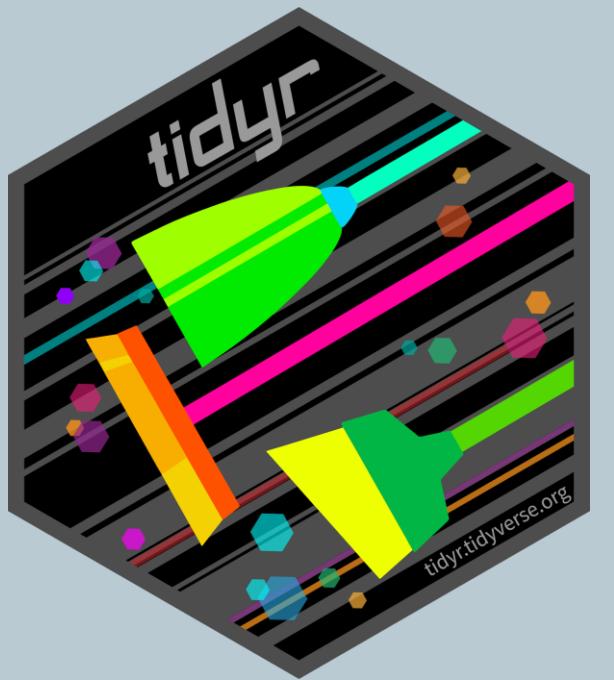
country	year	cases
Angola	1999	800
Angola	2000	750
Angola	2001	925
Angola	2002	1020
India	1999	20100
India	2000	25650
India	2001	26800
India	2002	27255
Mongolia	1999	450
Mongolia	2000	512
Mongolia	2001	510
Mongolia	2002	586

country	1999	2000	2001	2002
Angola	800	750	925	1020
India	20100	25650	26800	27255
Mongolia	450	512	510	586

Pivot data wider

```
data %>%  
  pivot_wider(  
    names_from = "year",  
    values_from = "cases"  
)
```

Credit: [Epidemiologist R Handbook](#)



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

Credit: [Garrick Aden-Buie](#)
& [Mara Averick](#)



Joins

- left_join()
- inner_join()
- full_join()

left_join(x, y)

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

inner_join(x, y)

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

full_join(x, y)

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

For a fun explanation using The Beatles & Rolling Stones see Nic Crane's tweet:

<https://bit.ly/3qfhBb9>

Credit: Garrick Aden-Buie

<https://www.garrickadenbuie.com/project/tidyexplain/>



Dates

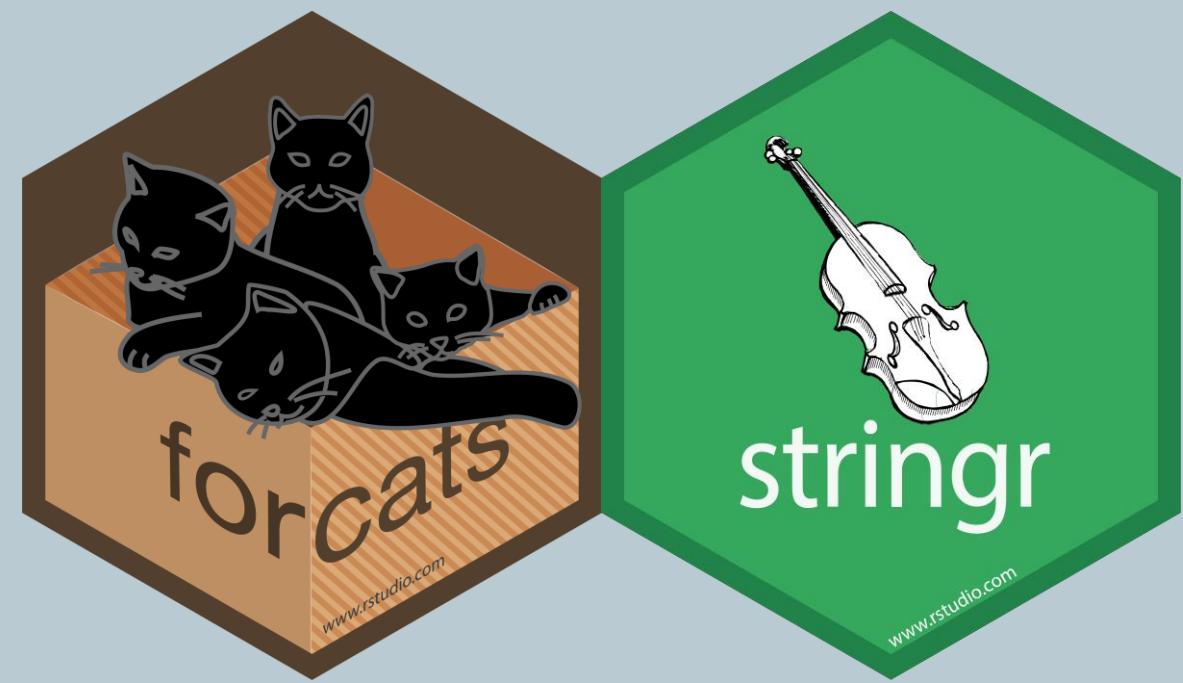
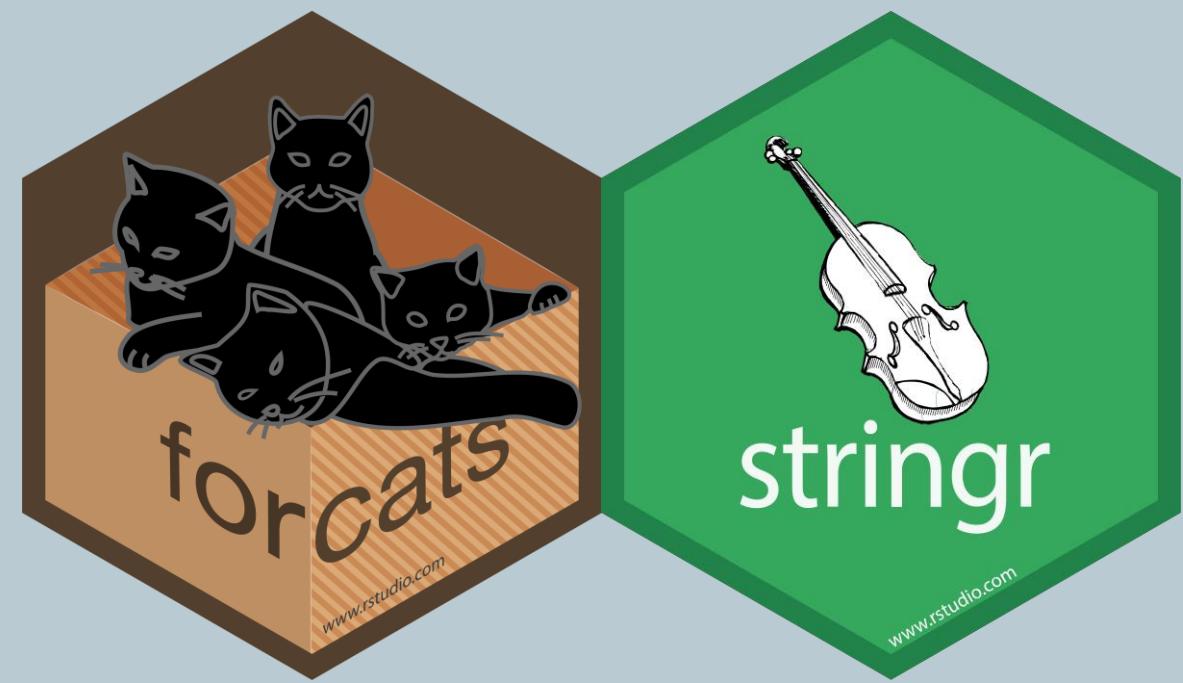
✿ `separate()` turns a character column into multiple columns

Order of elements in date-time	Parse function
year, month, day	⭐ <code>ymd()</code>
year, day, month	<code>ydm()</code>
month, day, year	<code>mdy()</code>
day, month, year	<code>dmy()</code>
hour, minute	<code>hm()</code>
hour, minute, second	<code>hms()</code>
year, month, day, hour, minute, second	<code>ymd_hms()</code>

*adapted from *Dates and Times Made Easy with lubridate* (Grolemund & Wickham, 2011)



```
#where col = name of column to separate  
# into = vector of names for column to be separated into  
# sep = value to separate column at  
separate(data, col, into, sep)  
  
#example we have seen before  
separate(financial_year, into = c("year", NA), sep = "/")
```



Factors & Strings

- ★ `levels()` to see the set of levels in the factor
- ★ `fct_relevel()` to manually reorder factor levels
- ★ `fct_recode()` to manually change the levels labels
- ★ `fct_collapse()` to collapse levels into manually defined groups
- ★ `str_replace()` to change the labels of a string
- ★ `str_wrap()` to wrap the text of a string if it is too long into 2+ lines

Example: Visualisation of the filter() function

```
filter(bill_length_mm > 45)
```

	species	bill_length_mm		species	bill_length_mm	
1	Adelie	41.80		1	Chinstrap	45.40
2	Adelie	36.50		2	Chinstrap	46.50
3	Adelie	40.30		3	Chinstrap	46
4	Chinstrap	45.40		4	Gentoo	45.10
5	Chinstrap	46.50		5	Gentoo	50.70
6	Chinstrap	46				
7	Gentoo	43.60				
8	Gentoo	45.10				
9	Gentoo	50.70				

Tidy Data Tutor

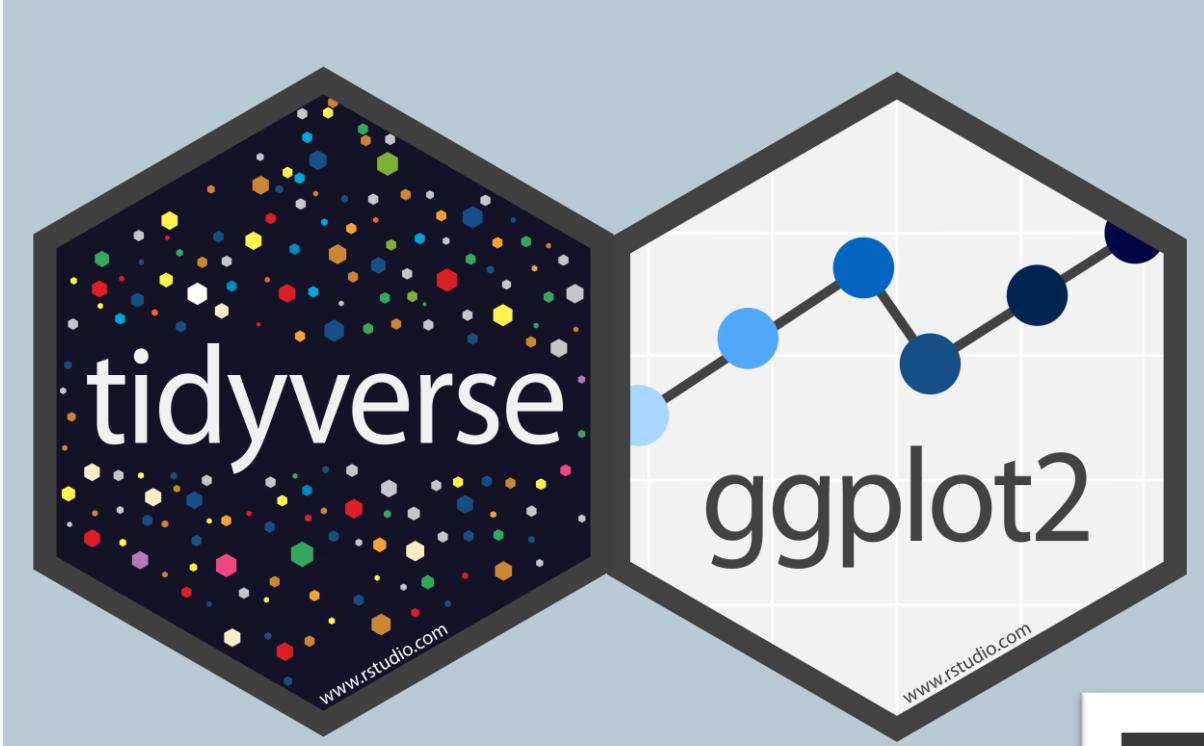
A potentially helpful revision and consolidation tool

<https://tidydatatutor.com/>

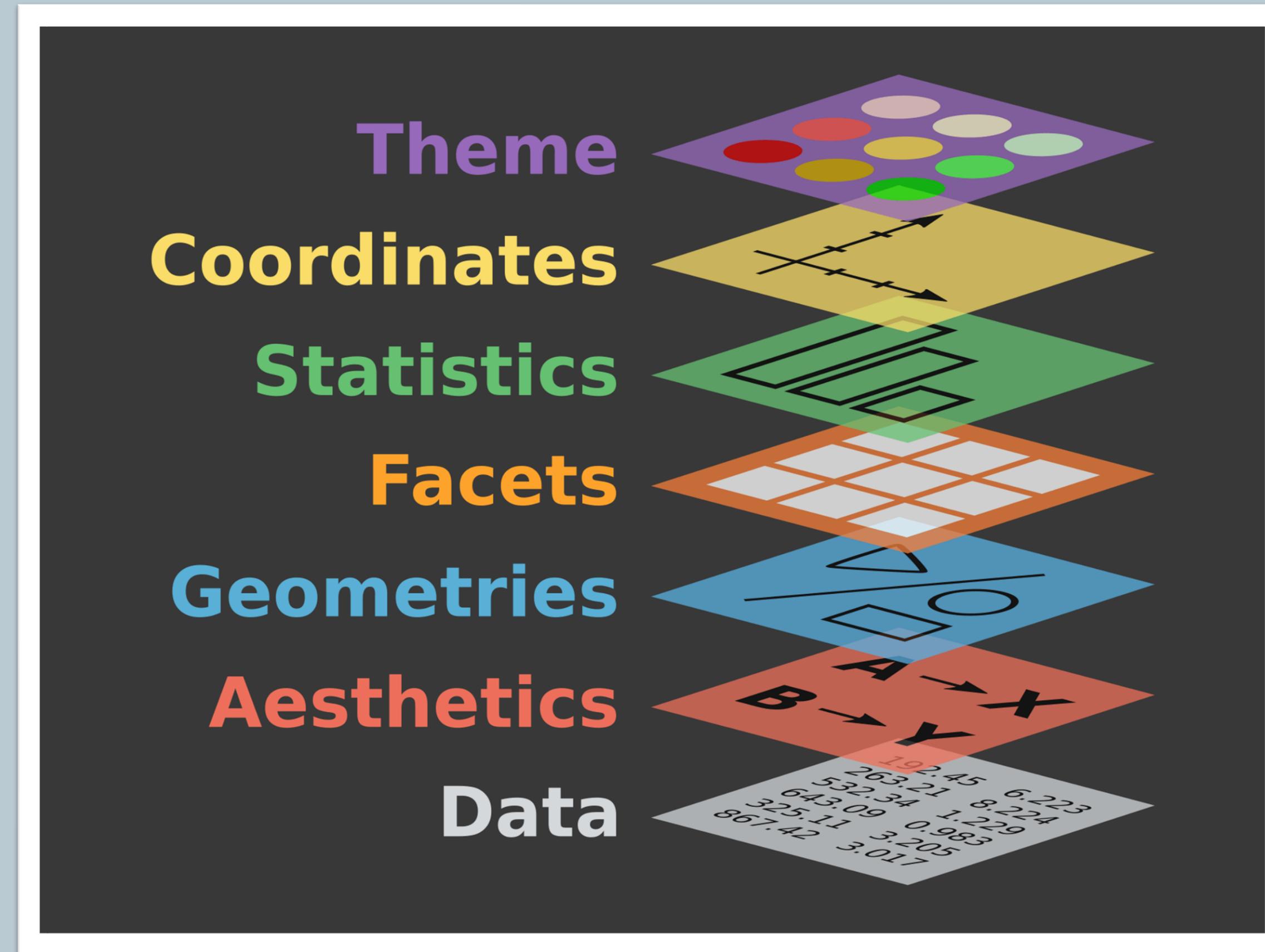
Plotting



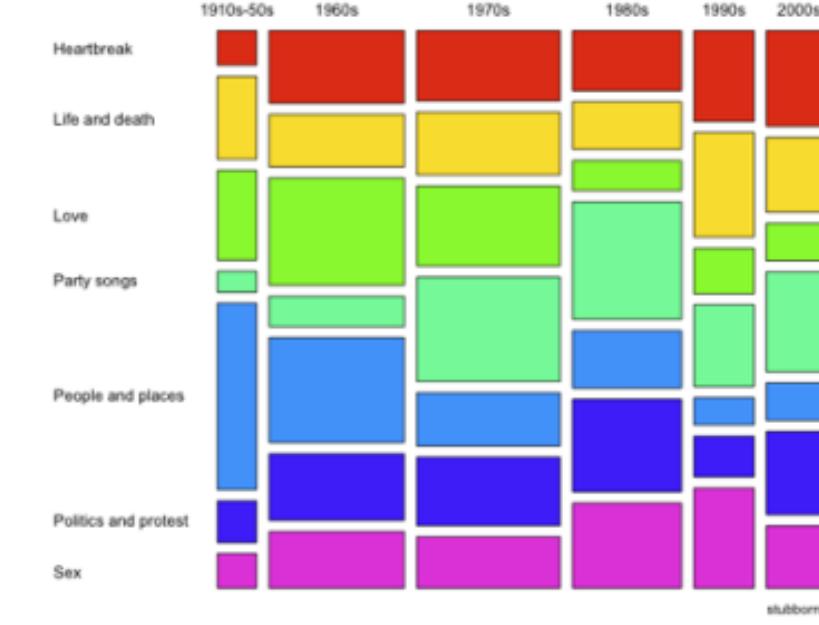
Artwork by Allison Horst



Grammar of Graphics



- ★ + theme_bw()
+ theme_classic() etc.
- ★ + coord_flip()
 - Example: + stat_summary()
- ★ + facet_wrap()
+ facet_grid(x~y)
- ★ + geom_line() + geom_bar() +
geom_jitter() etc.
- ★ ggplot(data, aes(x, y, color, shape,
fill))
- ★ ggplot(data)

	Numerical	Categorical
Numerical	<ul style="list-style-type: none"> Scatter plot (point) 2D binning (bin2d, hex) Contour plot (density2d) Quantiles (quantile, qq) Lines (line, smooth) Ribbons (ribbon, area) 	<ul style="list-style-type: none"> Boxplot (boxplot, violin) Counts (count, tile) Error bars (errorbar) Columns (col)
Categorical	<p>Which ggplot2 data viz is right for your data? (geoms in parentheses)</p>	 <ul style="list-style-type: none"> Mosaic (ggmosaic::geom_mosaic) Counts (count, tile)

Importance of variable class

From Dr Sam Tyner's guest lecture recording in Week 4

ggplot2 Cheatsheet

Data Visualization with ggplot2 :: CHEAT SHEET



Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a data set, a **coordinate system**, and **geoms**—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

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

required
Not required, sensible defaults supplied

`ggplot(data = mpg, aes(x = cty, y = hwy))` Begins a plot that you finish by adding layers to. Add one geom function per layer.

`last_plot()` Returns the last plot.

`ggsave("plot.png", width = 5, height = 5)` Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Aes Common aesthetic values.

color and **fill** - string ("red", "#RRGGBB")

linetype - integer or string (0 = "blank", 1 = "solid", 2 = "dashed", 3 = "dotted", 4 = "dotdash", 5 = "longdash", 6 = "twodash")

lineend - string ("round", "butt", or "square")

linejoin - string ("round", "mitre", or "bevel")

size - integer (line width in mm) 

shape - integer/shape name or a single character ("a") 



Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

a + geom_blank() and **a + expand_limits()**
Ensure limits include values across all plots.

b + geom_curve(aes(yend = lat + 1,
xend = long + 1), curvature = 1) - x, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend = "butt",
linejoin = "round", linemetre = 1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(alpha = 50)) - x, y, alpha, color, fill, group, subgroup, linetype, size

b + geom_rect(aes(xmin = long, ymin = lat,
xmax = long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, group, linetype, size

a + geom_ribbon(aes(ymin = unemploy - 900,
ymax = unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

a + geom_label(aes(label = cty), nudge_x = 1,
nudge_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_point()
x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile()
x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl")
x, y, alpha, color, linetype, size

e + geom_smooth(method = lm)
x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1,
nudge_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept = 0, slope = 1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend = lat + 1, xend = long + 1))

b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly()
x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5)
x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy))
x, y, alpha, color, fill, linetype, size, weight

discrete

```
d <- ggplot(mpg, aes(fl))
```

d + geom_bar()
x, alpha, color, fill, linetype, size, weight

g + geom_count()
x, y, alpha, color, fill, shape, size, stroke

e + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size

THREE VARIABLES

```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))
```

l + geom_contour(aes(z = z))
x, y, z, alpha, color, group, linetype, size, weight

l + geom_raster(aes(fill = z), hjust = 0.5,
vjust = 0.5, interpolate = FALSE)
x, y, alpha, fill

l + geom_contour_filled(aes(fill = z))
x, y, alpha, color, fill, group, linetype, size, subgroup

l + geom_tile(aes(fill = z))
x, y, alpha, color, fill, linetype, size, width

continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))
```

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density_2d()
x, y, alpha, color, group, linetype, size

h + geom_hex()
x, y, alpha, color, fill, size

continuous function

```
i <- ggplot(economics, aes(date, unemploy))
```

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))
```

j + geom_crossbar(fatten = 2) - x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar() - x, ymax, ymin, alpha, color, group, linetype, size, width
Also **geom_errorbarh()**.

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange() - x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

```
data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))
```

k + geom_map(aes(map_id = state), map = map)
+ expand_limits(x = map\$long, y = map\$lat)
map_id, alpha, color, fill, linetype, size

Color palettes

Emil Hvitfeldt has created a “one stop destination for anyone looking for a color palette to use in r.”

<https://github.com/EmilHvitfeldt/r-color-palettes>

Including an interactive color selector!
<https://emilhvitfeldt.github.io/r-color-palettes/discrete.html>



introverse package

https://sjspielman.github.io/introverse/articles/introverse_online.html

introverse 0.0.1 Home Get help here Get help in RStudio RStudio reference Resources and tutorials ▾

Get help online

Source: vignettes/introverse_online.Rmd

Contents

Get help with the example datasets

Get help with the example datasets

- [carnivores](#)
- [msleep](#)

Get help with operators and magrittr pipes

Get help with operators and magrittr pipes

- [Assignment operators in R](#)
- [Mathematical operators in R](#)
- [Logical operators in R](#)
- [magrittr pipe](#)
- [magrittr assignment pipe](#)

Get help with Base R

Get help with Base R

Get help with ggplot2

Get help with dplyr

Get help with tidyselect helpers

Get help withforcats

Get help with readr

Get help with tibble

Get help with tidyverse

Get help with stringr

Get help with glue

Report generation



Artwork by Allison Horst



✿ kbl() for nice tables

R Markdown

The screenshot shows an RStudio interface with an R Markdown document. The code editor contains the following content:

```
1 ---  
2 title: "Title"  
3 author: "Author"  
4 date: "Date"  
5 output: html_document  
---  
|  
8 ``{r setup, include=FALSE}  
9 knitr::opts_chunk$set(echo = TRUE)  
10  
11  
12 ## R Markdown  
13  
14 This is an R Markdown document. Markdown is a simple formatting syntax for  
authoring HTML, PDF, and MS Word documents. For more details on using R Markdown  
see <http://rmarkdown.rstudio.com>.  
15  
16 When you click the **Knit** button a document will be generated that includes  
both content as well as the output of any embedded R code chunks within the  
document. You can embed an R code chunk like this:  
17  
18 ``{r cars}  
19 summary(cars)  
20  
21  
22 ## Including Plots  
23  
24 You can also embed plots, for example:  
25  
26 ``{r pressure, echo=FALSE}  
27 plot(pressure)  
28  
29  
30 Note that the `echo = FALSE` parameter was added to the code chunk to prevent  
printing of the R code that generated the plot.  
31
```

Annotations with curly braces point to specific parts of the code:

- A blue brace labeled "YAML" points to the YAML front matter at the top of the file.
- Four red braces labeled "Text" point to the explanatory text blocks (e.g., "This is an R Markdown document...", "When you click the **Knit** button...").
- Three purple braces labeled "Code" point to the R code chunks (e.g., ``{r setup, include=FALSE}``, ``{r cars}``, ``{r pressure, echo=FALSE}``).

A bit more on knitting to PDFs



Questions?

R Programming Assignment

Due: Monday, 28th November at 12 noon GMT

See Learn pages “Assessments” for more info

Q&A around the assignment Wednesday 9th November 6pm!