Data Visualization with R
Workshop Part 2

Tuberculosis incidence

The TB data is from the WHO.

```
## # A tibble: 40,800 × 5
##    country      year age_group sex   count
##    <chr>       <dbl> <fct>     <chr> <dbl>
##  1 Afghanistan  1997 15-24     m        10
##  2 Afghanistan  1997 25-34     m         6
##  3 Afghanistan  1997 35-44     m         3
##  4 Afghanistan  1997 45-54     m         5
##  5 Afghanistan  1997 55-64     m         2
##  6 Afghanistan  1997 65-       m         0
##  7 Afghanistan  1997 15-24     f        38
##  8 Afghanistan  1997 25-34     f        36
##  9 Afghanistan  1997 35-44     f        14
## 10 Afghanistan  1997 45-54     f         8
## # … with 40,790 more rows
```

What is a choropleth map?

Why use a choropleth map?

How do we get a map?

A polygon map of the world can be extracted from the `maps` package.

```
world_map <- map_data("world")
world_map %>%
  filter(region == "Australia") %>%
      DT::datatable(width=1150, height=100)
```

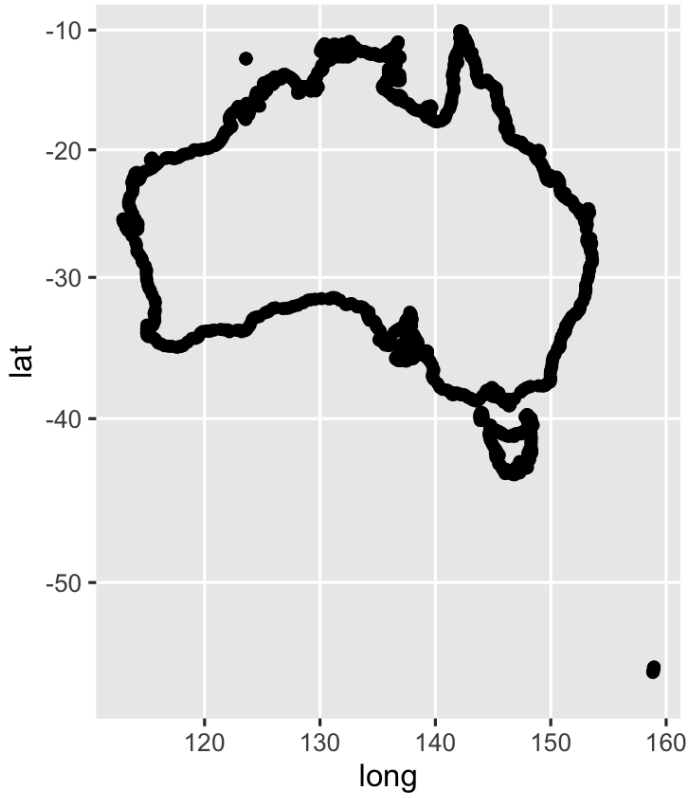
Show 10 entries Search:

	long	lat	group	order	region	subregion
1	123.594528198242	-12.4256830215454	133	7115	Australia	Ashmore and Cartier Islands
2	123.595207214355	-12.4359369277954	133	7116	Australia	Ashmore and Cartier Islands
3	123.573150634766	-12.4341802597046	133	7117	Australia	Ashmore and Cartier Islands
4	123.572463989258	-12.4239253997803	133	7118	Australia	Ashmore and Cartier Islands
5	123.594528198242	-12.4256830215454	133	7119	Australia	Ashmore and Cartier Islands
6	158.878799438477	-54.7097625732422	139	7267	Australia	Macquarie Island
7	158.84521484375	-54.7492179870605	139	7268	Australia	Macquarie Island

Maps are basically groups of connected dots

These are the points, defining the country boundary for Australia

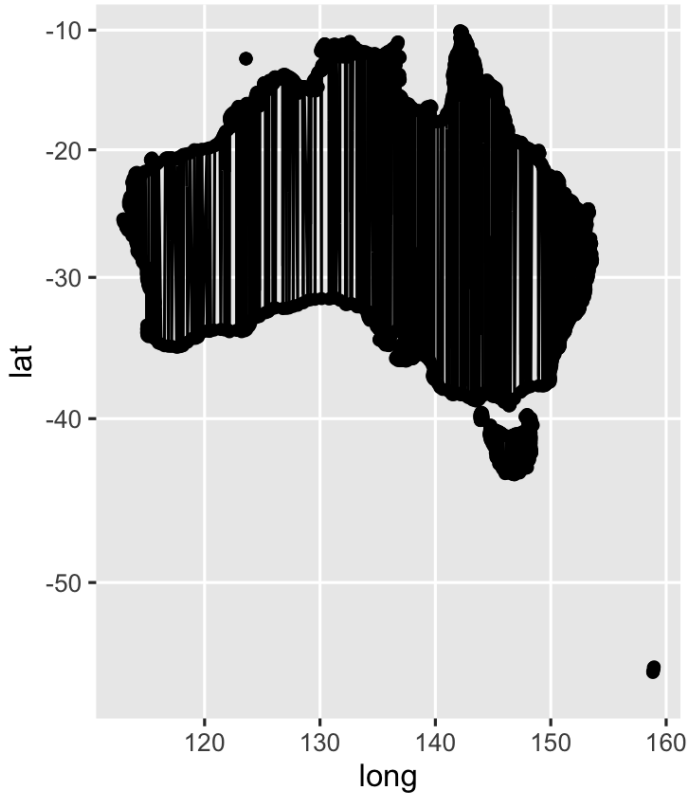
```
oz <- world_map %>%
  filter(region == "Australia")
ggplot(oz, aes(x = long, y = lat)) +
  geom_point() +
  coord_map()
```



Maps are basically groups of connected dots

Connect the dots

```
ggplot(oz, aes(x = long, y = lat,
               group = group)) +
  geom_point() +
  geom_line() +
  coord_map()
```

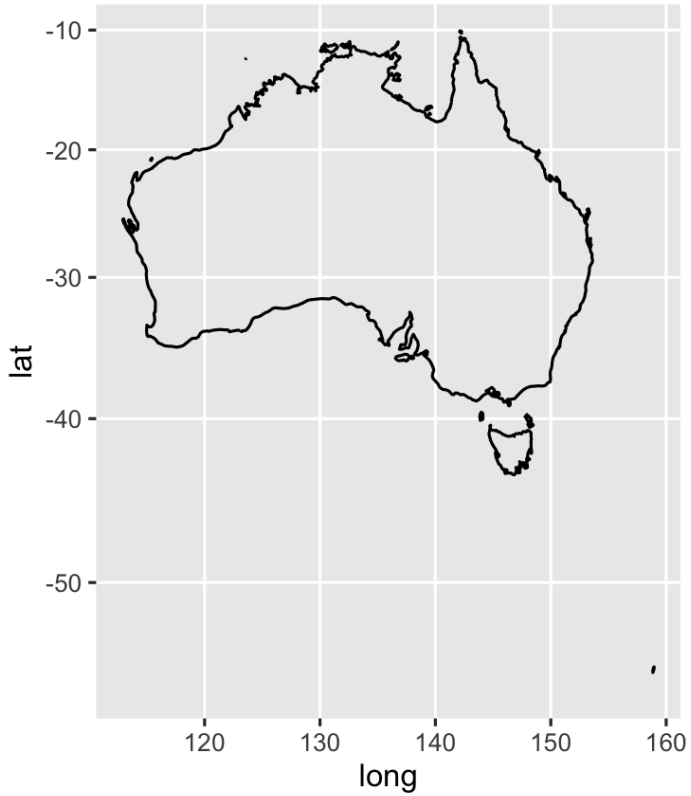


What happened?

Maps are basically groups of connected dots

Connect the dots

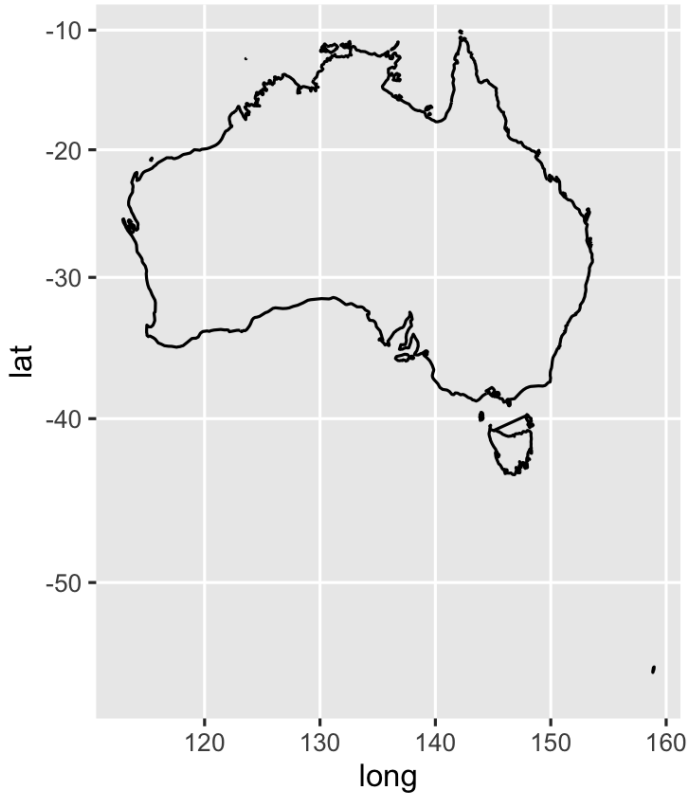
```
ggplot(oz, aes(x = long, y = lat,
               group = group)) +
  #geom_point() +
  geom_path() +
  coord_map()
```



Maps are basically groups of connected dots

This map doesn't have states and territory connections, and also sub-regions is not uniquely defining islands.

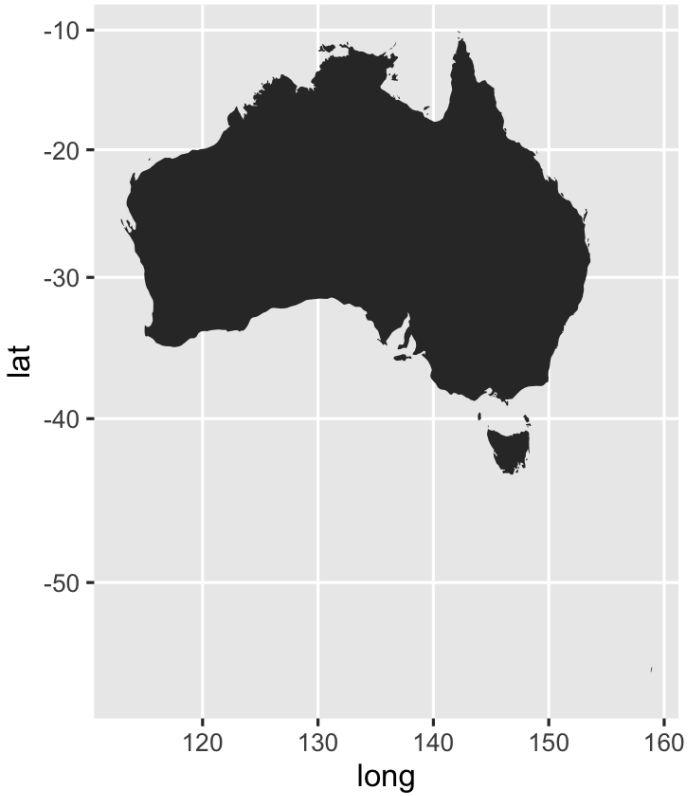
```
ggplot(oz, aes(x = long, y = lat,
               group = subregion)) +
  geom_path() +
  coord_map()
```



Maps are basically groups of connected dots

We can also plot the map using `geom_polygon`, and fill with colour.

```
ggplot(oz, aes(x = long, y = lat,
               group = group)) +
  geom_polygon() +
  coord_map()
```



Maps are basically groups of connected dots

Using a map theme makes the result look more map-like

```
ggplot(oz, aes(x = long, y = lat,
               group = group)) +
  geom_polygon() +
  coord_map() +
  theme_map()
```

Tips for mapping

For data analysis, maps are a set of points, connected correctly to generate polygons.

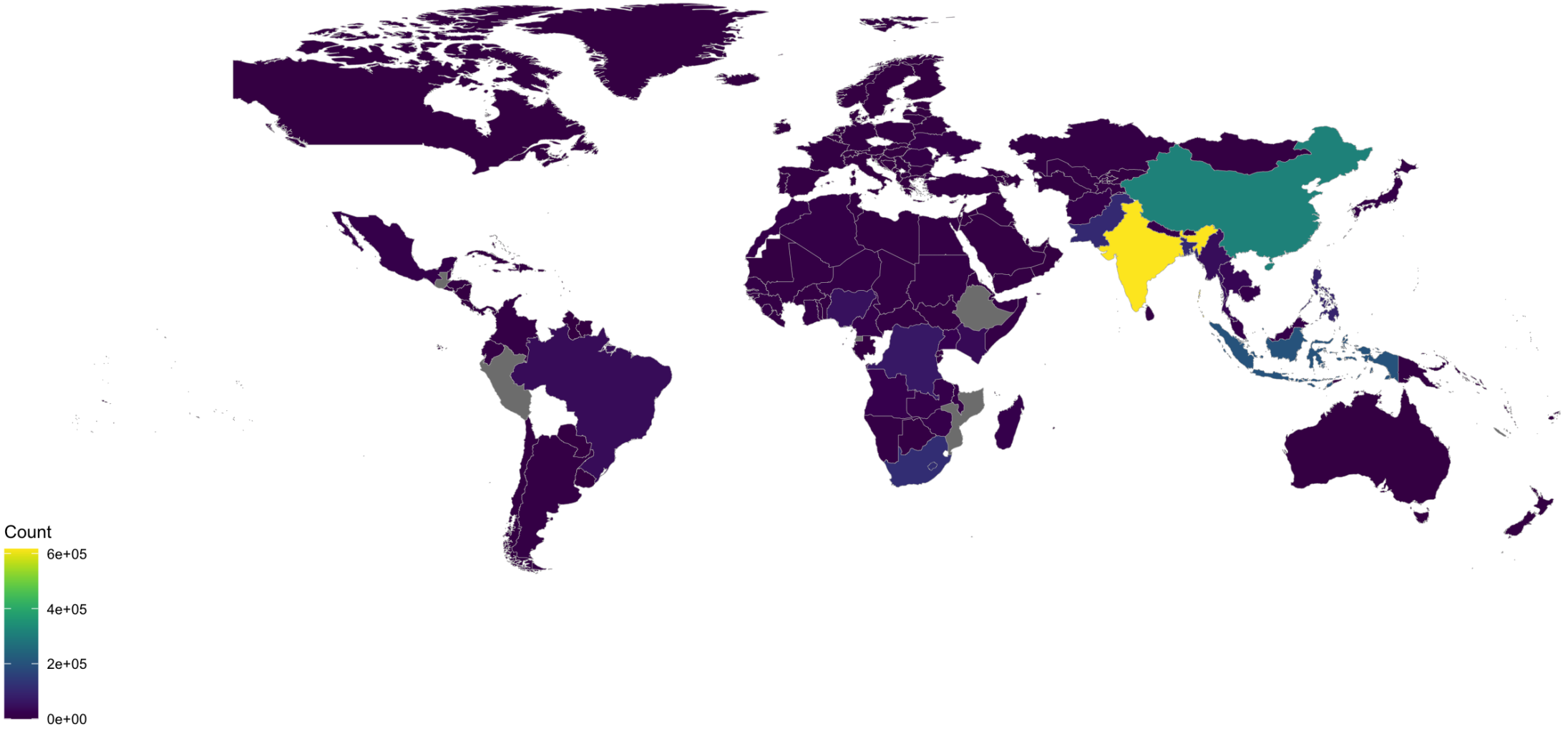
Note: It is important when converting spatial objects from a mapping software to a data analysis project is "thinning" the map to make it smaller and efficient to work with. Both the rmapshaper package and `st_simplify` in the sf have tools to thin the number of points defining a polygon, while respecting the shape, and adjacent boundaries.

Let's make a choropleth map of tuberculosis

Pre-process the data

Aggregate counts across sex and age group for 2012

```
tb_2012 <- tb %>%
  filter(year == 2012) %>%
  rename(region = country) %>%
  group_by(region) %>%
  summarise(count = sum(count))
ggplot(tb_2012, aes(map_id = region)) +
    geom_map(aes(fill = count), map = world_map,
             color="grey70", size = 0.1, na.rm = TRUE) +
    expand_limits(x = world_map$long, y = world_map$lat) +
    scale_fill_viridis("Count") +
    theme_map()
```



What happened to the USA? UK?

Check the name matching

```
wm_names <- world_map %>%
  select(region) %>%
  distinct()
tb_names <- tb %>%
  filter(year == 2012) %>%
  select(country) %>%
  distinct()
tb_miss_from_wm <- anti_join(tb_names, wm_names,
                             by=c("country" = "region"))
wm_miss_from_tb <- anti_join(wm_names, tb_names,
                             by=c("region" = "country"))
```

```
DT::datatable(tb_miss_from_wm, width = 1150, height = 100)
```

Show 10 entries Search:

	country
1	Antigua and Barbuda
2	Bolivia (Plurinational State of)
3	British Virgin Islands
4	Brunei Darussalam
5	Cabo Verde
6	China, Hong Kong SAR
7	China, Macao SAR
8	Congo
9	Côte d'Ivoire
10	Curaçao

Showing 1 to 10 of 33 entries Previous 1 2 3 4 Next

```
DT::datatable(wm_miss_from_tb, width = 1150, height = 100)
```

Show 10 entries Search:

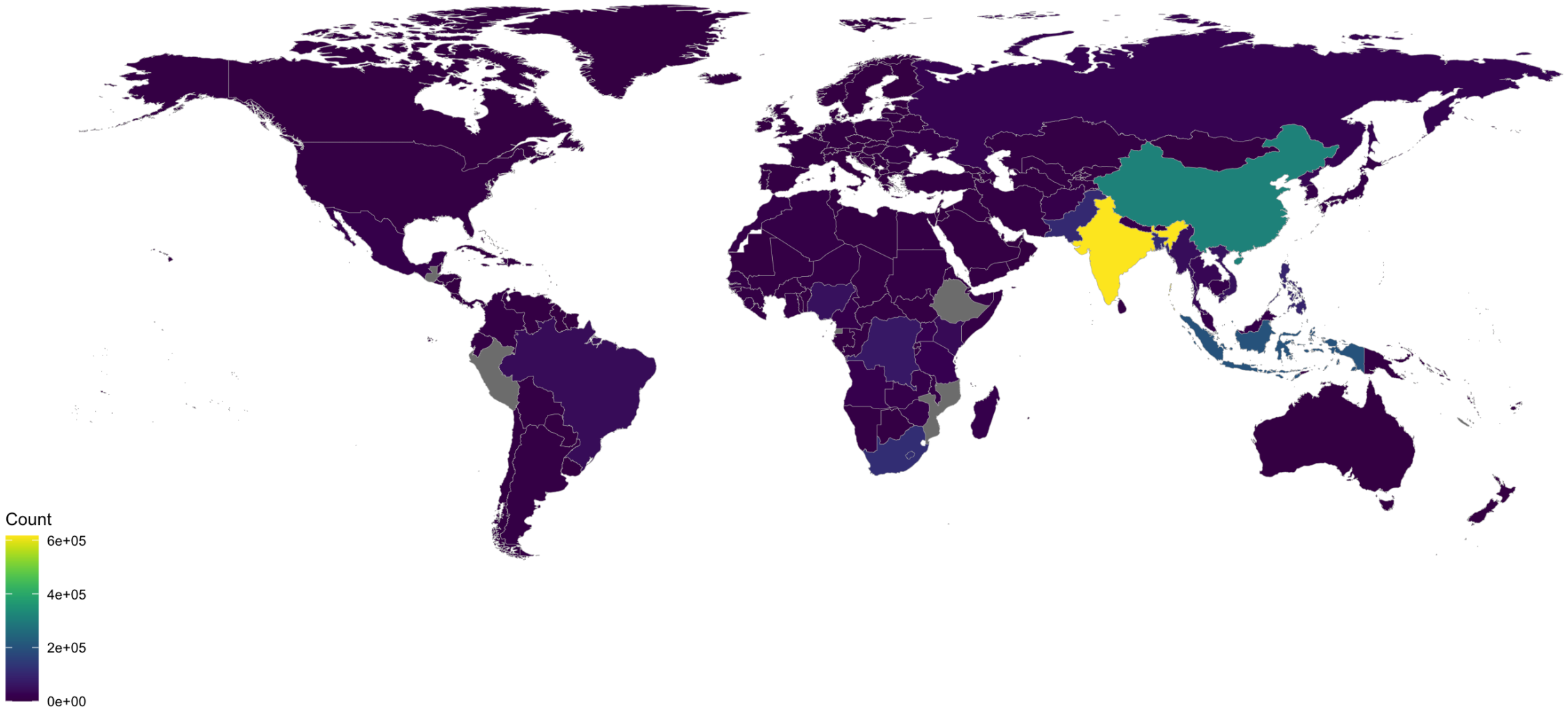
	region
1	Antarctica
2	French Southern and Antarctic Lands
3	Antigua
4	Barbuda
5	Saint Barthelemy
6	Bolivia
7	Brunei
8	Ivory Coast
9	Republic of Congo
10	Cape Verde

Showing 1 to 10 of 70 entries Previous 1 2 3 4 5 6 7 Next

```r
tb_fixed <- tb %>%
  mutate(region=recode(country,
                       "United States of America" = "USA",
                       "United Kingdom of Great Britain and Northern Ireland" =
                       "Russian Federation" = "Russia",
                       "Viet Nam" = "Vietnam",
                       "Venezuela (Bolivarian Republic of)" = "Venezuela",
                       "Bolivia (Plurinational State of)" = "Bolivia",
                       "Czechia" = "Czech Republic",
                       "Iran (Islamic Republic of)" = "Iran",
                       "Iran (Islamic Republic of)" = "Laos",
                       "Democratic People's Republic of Korea" = "North Korea",
                       "Republic of Korea" = "South Korea",
                       "United Republic of Tanzania" = "Tanzania",
                       "Congo" = "Republic of Congo"))
```

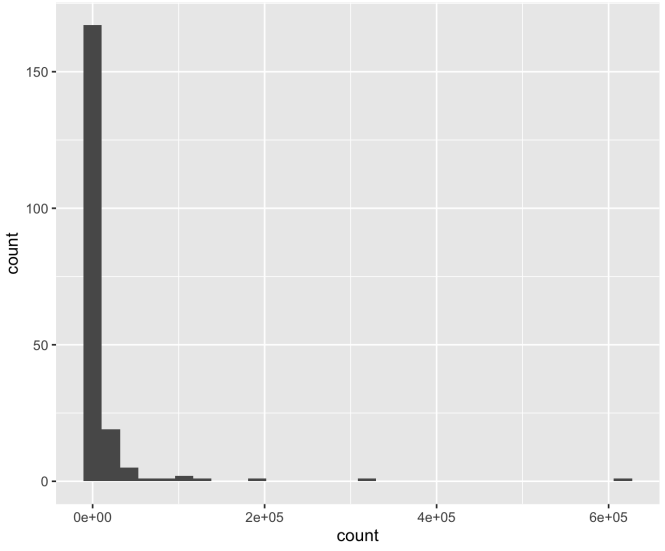
😄 Try again!

```
tb_2012 <- tb_fixed %>%
  filter(year == 2012) %>%
  group_by(region) %>%
  summarise(count = sum(count))
ggplot(tb_2012, aes(map_id = region)) +
    geom_map(aes(fill = count), map = world_map,
             color = "grey70", size = 0.1, na.rm = TRUE) +
    expand_limits(x = world_map$long, y = world_map$lat) +
    scale_fill_viridis("Count") +
    theme_map()
```


Count
6e+05
4e+05
2e+05
0e+00

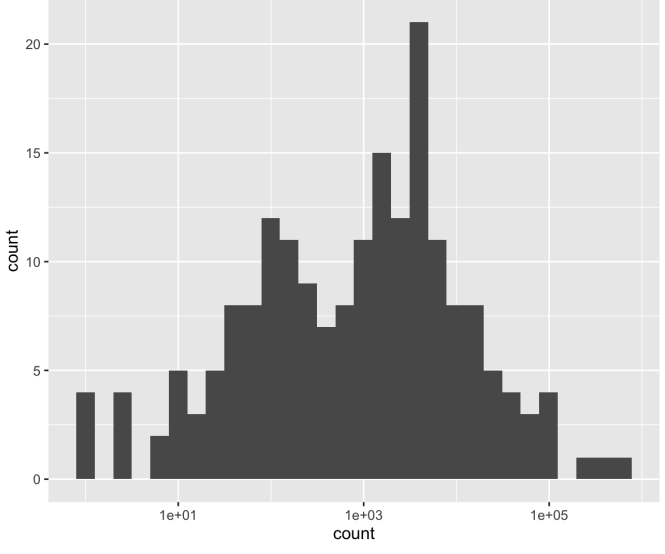
Counts are typically skewed

```
ggplot(tb_2012, aes(x = count)) +
  geom_histogram()
```



Symmetrising count, helps visual perception of a choropleth map.

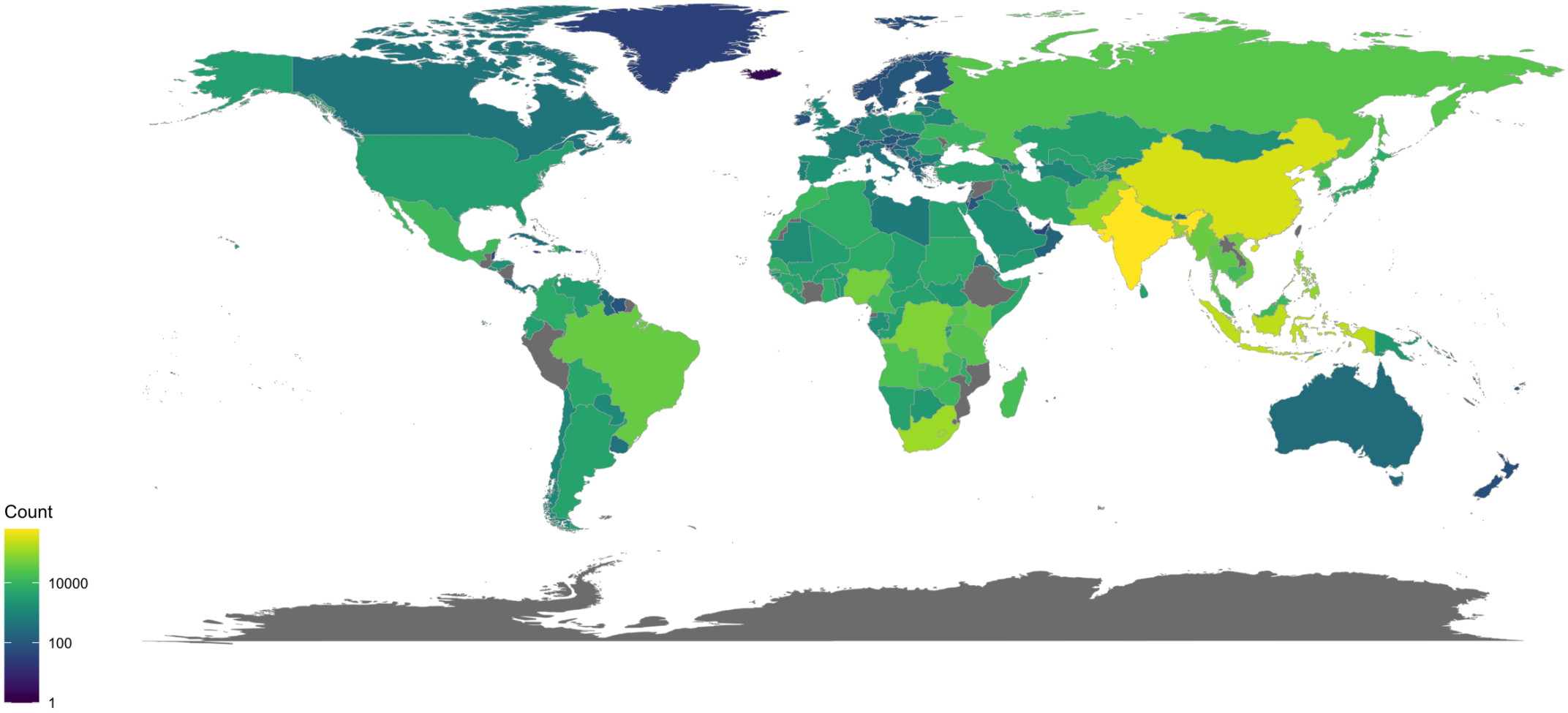
```
ggplot(tb_2012, aes(x = count)) +
  geom_histogram() +
  scale_x_log10()
```



Choropleth on log scale

```
tb_2012_map <- world_map %>% left_join(tb_2012)
ggplot(tb_2012_map, aes(x = long, y = lat, group=group)) +
    geom_polygon(aes(fill = count),
              color="grey70", size = 0.1, na.rm = TRUE) +
    expand_limits(x = world_map$long*1.1, y = world_map$lat*1.1) +
    scale_fill_viridis("Count", trans = "log10") +
    theme_map()
```

Note: `geom_polygon()` can be used instead of `geom_map()`. Also `geom_sf()` works similarly with `sf` spatial polygons.


Count
10000
100
1

Choropleth maps can be misleading

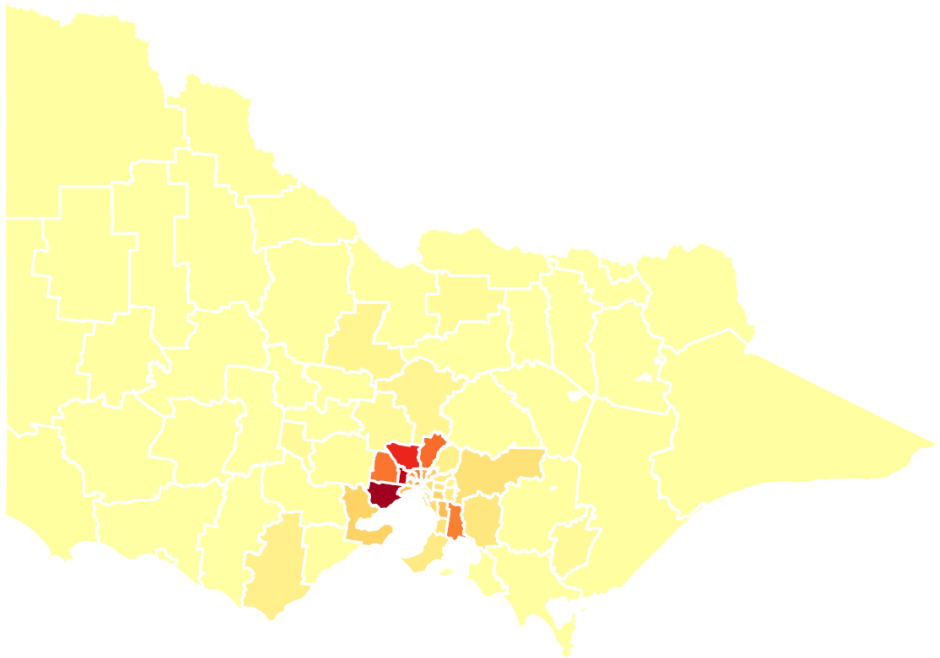
The population is not likely to be uniformly distributed across space. Big spatial areas may have few individuals, and high density areas are likely to be very small spatially. Choropleth maps can mislead the reader about the distribution of the statistic relative to a population.

Think about using a cartogram, which transforms the spatial polygons to represent the population whilst keeping faithful to geographic proximity.

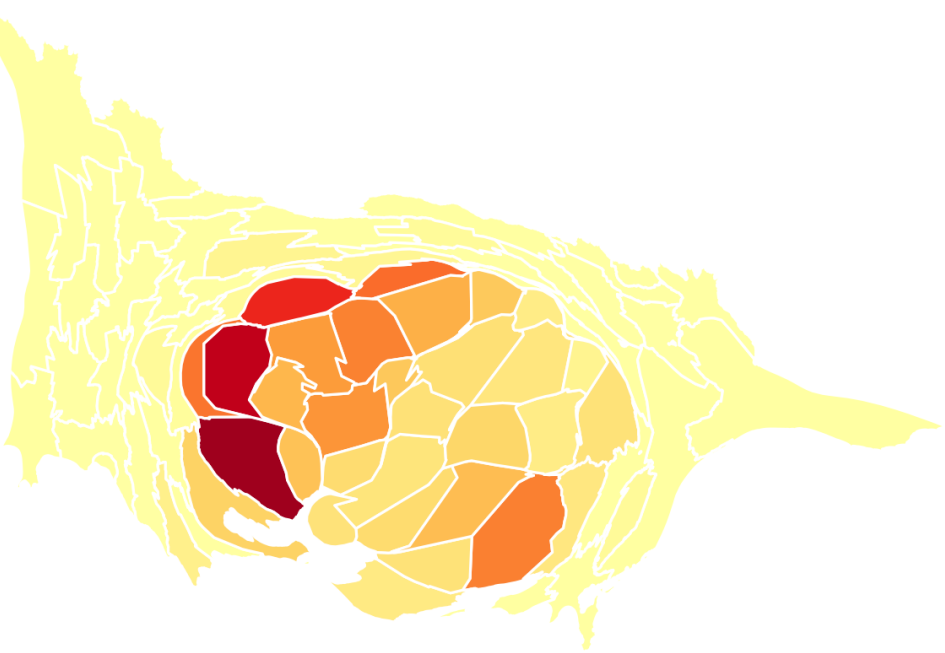
"Land doesn't vote, people do"

COVID incidence in Victoria 2020

Choropleth



Cartogram



The cartogram package can be used to transform the polygons. Other alternative include hexagon binning with sugarbag, and spatial facets with geofacet.

Point data overlaid on a map

Where are the platypi?

```
load(here::here("data/platypus.rda"))
p <- ggplot(platypus) +
  geom_point(aes(x = Longitude,
                 y = Latitude),
             alpha = 0.1)
p
```

```
p + coord_map()
```

Extract Open Street Map using `ggmap`

Download and save the map, so that you don't need to do multiple downloads.

```
oz_bbox <- c(112.9, # min long
             -45, # min lat
             159, # max long
             -10) # max lat
oz_map <- get_map(location = oz_bbox, source = "osm")
save(oz_map, file=here::here("data/oz_map.rda"))
```

Platypus occurrences across Australia

```
load(here::here("data/oz_map.rda"))
ggmap(oz_map) +
  geom_point(data = platypus,
             aes(x = Longitude,
                 y = Latitude),
             alpha = 0.1,
             colour = "orange") +
  theme_map()
```

</> Open `part2-exercise-02.Rmd`

15:00

Resources

These are sites with lots of useful information about making maps in R:

https://www.littlemissdata.com/blog/maps

https://www.r-spatial.org/r/2018/10/25/ggplot2-sf.html

https://www.paulamoraga.com/book-geospatial/sec-spatialdataandCRS.html

https://rspatialdata.github.io

Thematic maps with tmap

Spatial polygons with sf

Session Information

```
devtools::session_info()
```

```
## ─ Session info ─────────────────────────────────────────────────────────────
##  setting  value
##  version  R version 4.1.2 (2021-11-01)
##  os       macOS Big Sur 11.5.1
##  system   aarch64, darwin20
##  ui       X11
##  language (EN)
##  collate  en_AU.UTF-8
##  ctype    en_AU.UTF-8
##  tz       Australia/Melbourne
##  date     2022-02-20
##  pandoc   2.16.2 @ /usr/local/bin/ (via rmarkdown)
##
```