Data Visualization with R
Workshop Part 2

Applying interactivity and animation

The purpose of interactivity is to display more than can be achieved with persistent plot elements, and to invite the reader to engage with the plot.

Mouse-over labels de-clutters a plot

Pan/zoom allows re-focusing attention

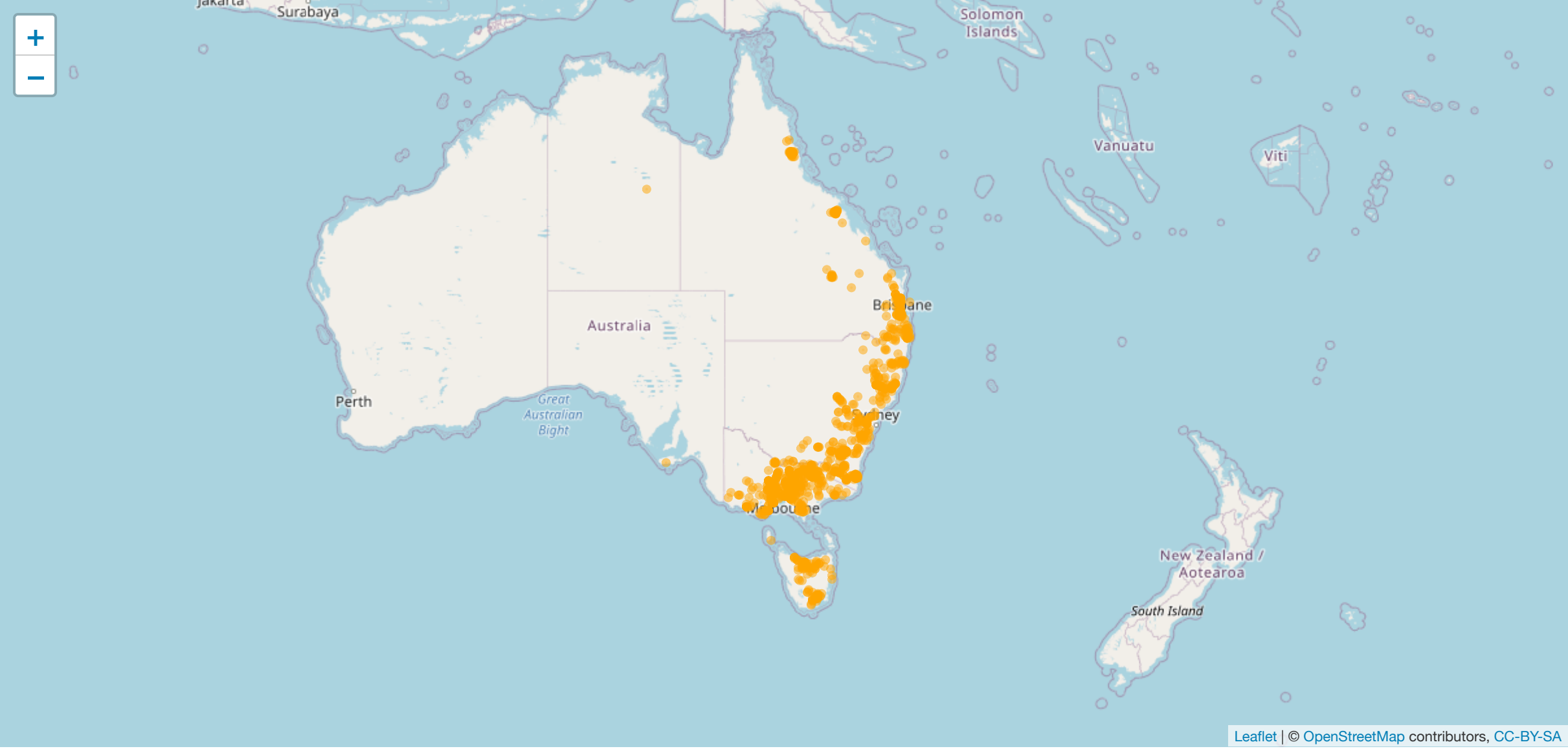
Selection allows focusing attention

Animation allows more information to be displayed, but developer keeps control. Beware that it is easy to forget what was just displayed, so keeping some elements persistent, maybe faint, can be useful for the reader.

Interactive maps

Leaflet

```
load(here::here("data/platypus.rda"))
platypus <- platypus %>%
  filter(year(eventDate) > 2018)
platypus %>%
  leaflet() %>%
  addTiles() %>%
  addCircleMarkers(
    radius = 1, opacity = 0.5, color = "orange", label = ~eventDate,
    lat = ~Latitude, lng = ~Longitude)
```


Surabaya
Solomon Islands
Vanuatu
Viti
Australia
Perth
Great Australian Bight
New Zealand / Aotearoa
South Island
Leaflet | © OpenStreetMap contributors, CC-BY-SA

Reflection on leaflet

Advantages

fast, scalable, reliable

many map formats

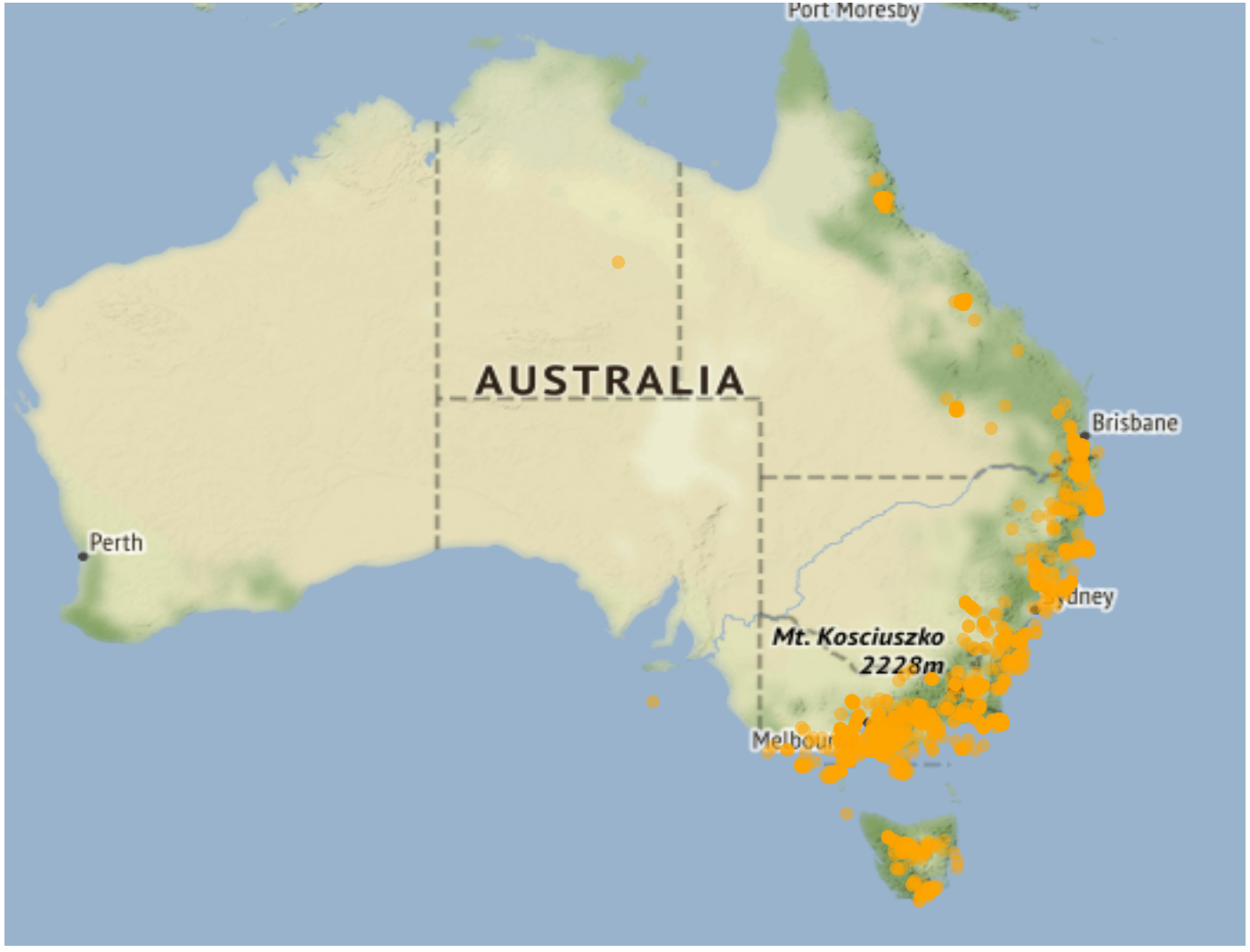
Disadvantages

specialist syntax

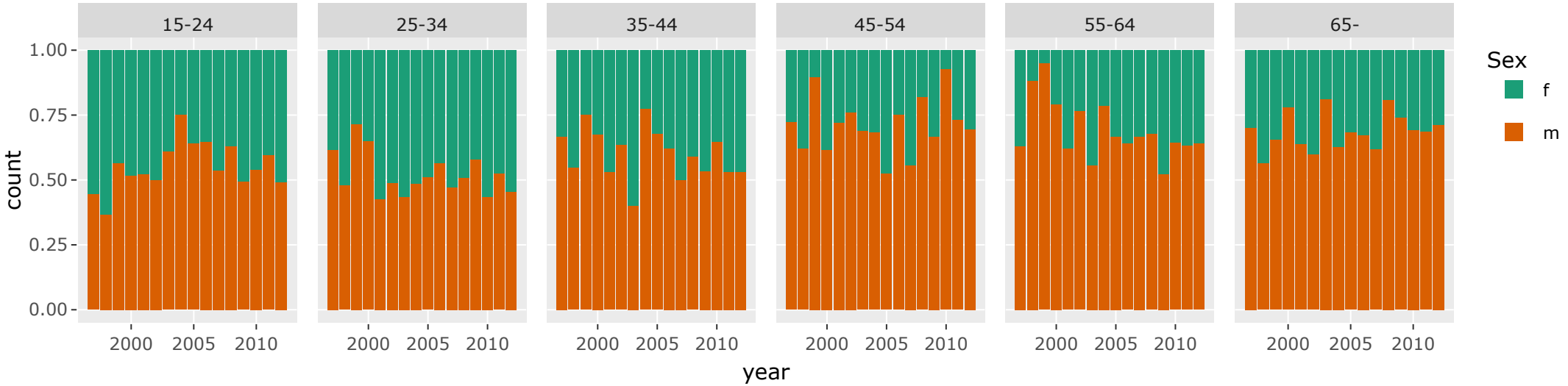
limited capabilities

Building on ggplot with `plotly`

```
load(here::here("data/oz_map.rda"))
p <- ggmap(oz_map) +
  geom_point(data = platypus,
             aes(x = Longitude, y = Latitude, label=eventDate),
             alpha = 0.5, colour = "orange") +
  theme_map()
ggplotly(p, tooltip = "label")
```


Port Moresby
AUSTRALIA
Brisbane
Perth
Mt. Kosciuszko
2228m

```r
p1 <- ggplot(tb_oz, aes(x = year, y = count, fill = sex)) +
  geom_bar(stat = "identity", position = "fill") +
  facet_wrap(~age_group, ncol = 6) +
  scale_fill_brewer(name = "Sex", palette = "Dark2")
ggplotly(p1)
```



Modifying plotly

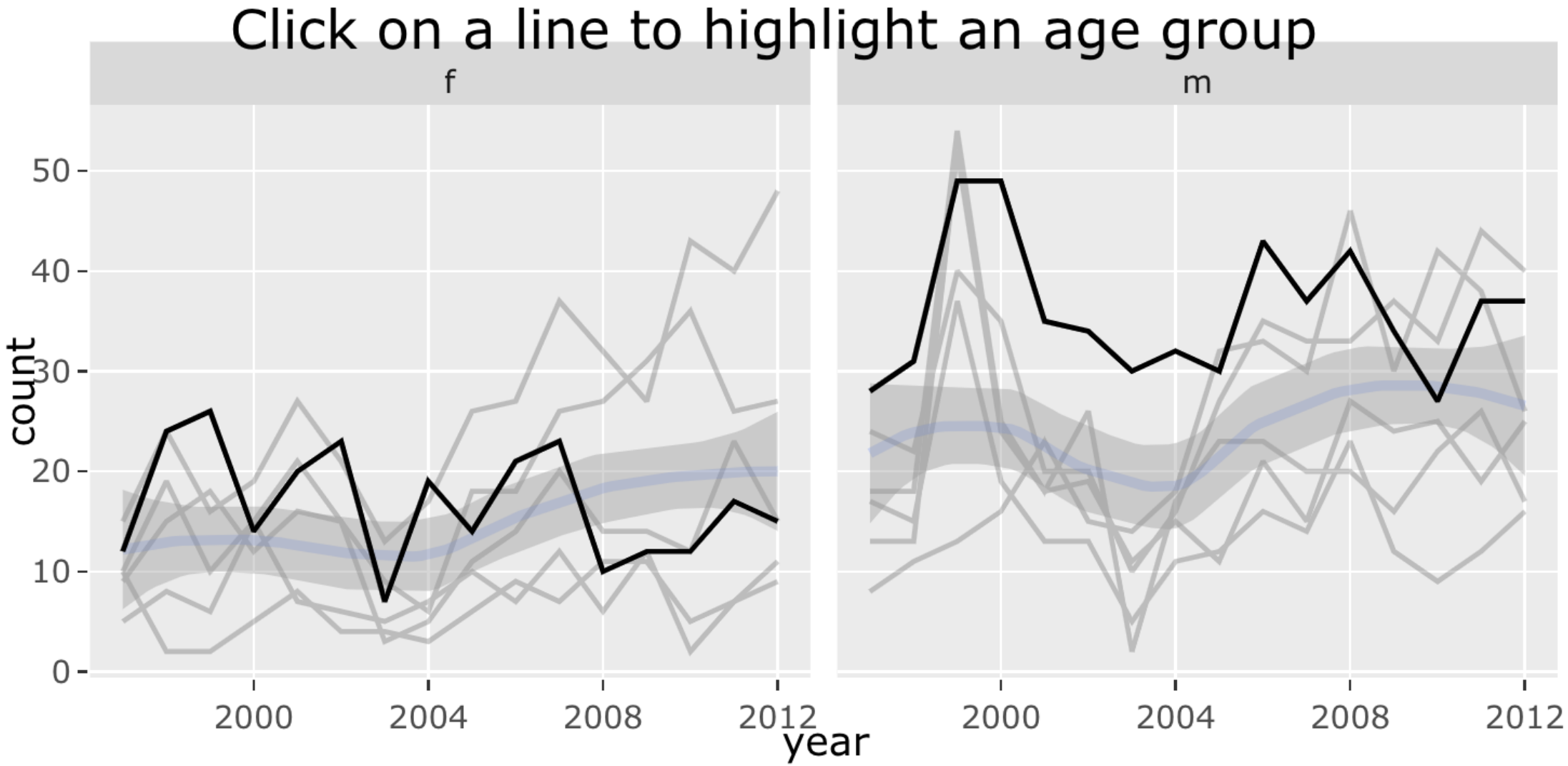
`plotly` uses elements of `crosstalk` to provide additional interactivity, such as linked highlighting. It only runs in a shiny environment, eg RStudio plot window, so copy the block of code into your R window.

```
tb_action <- highlight_key(tb_oz, ~age_group)

p2 <- ggplot(tb_action, aes(x = year, y = count)) +
  geom_line(aes(group = age_group)) +
  geom_smooth() +
  facet_wrap(~sex)

gg <- ggplotly(p2, height = 300, width = 600) %>%
   layout(title = "Click on a line to highlight an age group")

highlight(gg)
```


Click on a line to highlight an age group
f
m
count
50
40
30
20
10
0
2000
2004
2008
2012
year

Animations

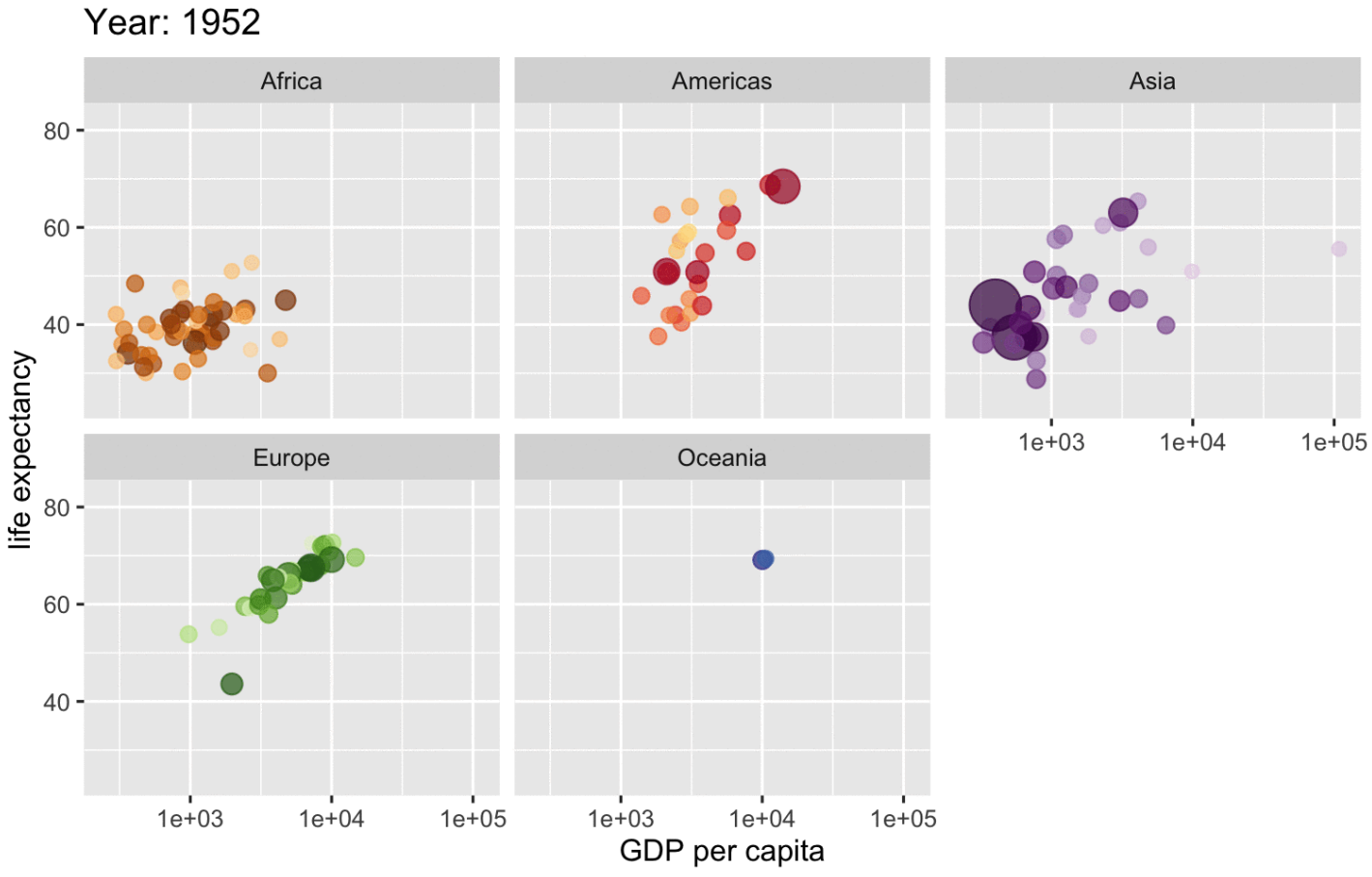
`gganimate` (Lin-Pederson) allows to make and save animations (also `plotly` can too)

Animations are different from interactive graphics in that the viewer does not have any control

useful for different important stages of a visualization (e.g. time) and to keep track of how different visualizations are related

makes slides come alive in talks.

An example animation: gapminder



Countries are colored manually by `country_colors` (hue shows continent, saturation is individual country)

How does `gganimate` work?

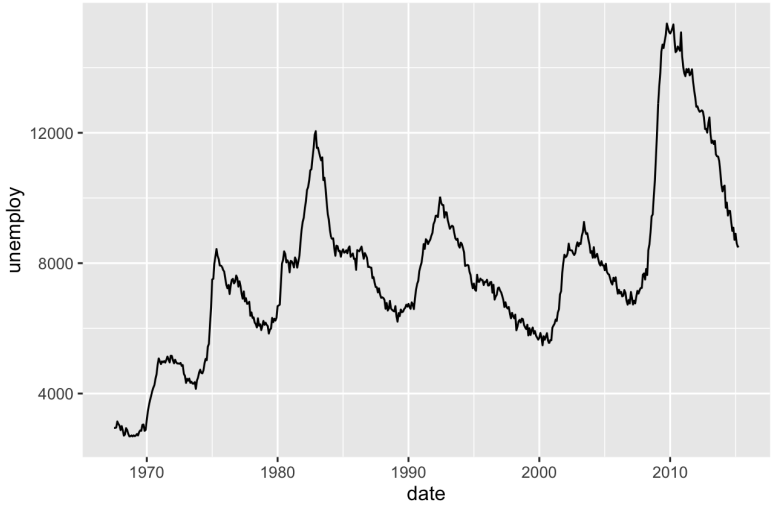
Layer up!

1. Start with a ggplot2 specification
2. Add animation specifications

A simple example (thanks to Mitch O'Hara Wild)

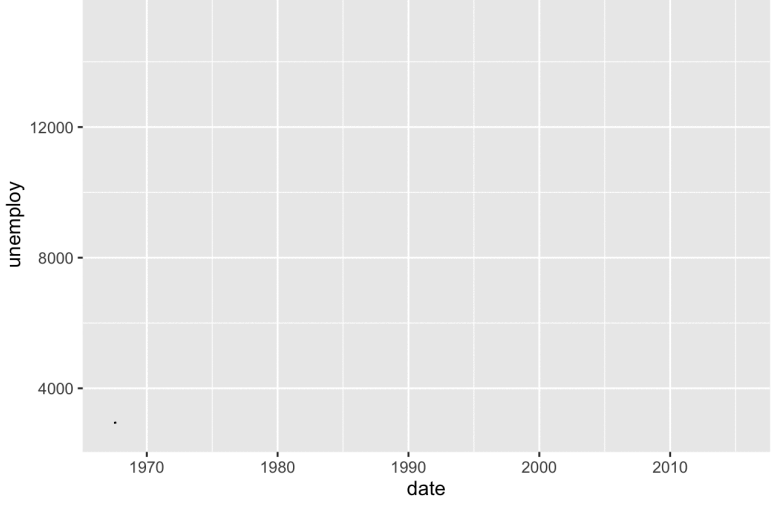
Start with a ggplot specification!

```
ggplot(economics) +
  aes(date, unemploy) +
  geom_line()
```



One line turns creates animation!

```
ggplot(economics) +
  aes(date, unemploy) +
  geom_line() +
  transition_reveal(date)
```



Controlling an animation

We control plot movement with (a grammar of animation):

Transitions: `transition_*()` define how the data should be spread out and how it relates to itself across time.

Views: `view_*()` defines how the positional scales should change along the animation.

Shadows: `shadow_*()` defines how data from other points in time should be presented in the given point in time.

Entrances/Exits: `enter_*()` and `exit_*()` define how new data should appear and how old data should disappear during the course of the animation.

Easing: `ease_aes()` defines how different aesthetics should be eased during transitions.

```
ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop, colour
  geom_point(alpha = 0.7) +
  scale_colour_manual(values = country_colors, guide=FALSE) +
  scale_size("Population size", range = c(2, 12), breaks=c(1*
  scale_x_log10() +
  facet_wrap(~continent) +
  theme(legend.position = "none") +
  # Here comes the gganimate specific bits
  labs(title = 'Year: {frame_time}',
       x = 'GDP per capita',
       y = 'life expectancy') +
  gganimate::transition_time(year) +
  gganimate::ease_aes('linear')
```

A not-so-simple example, the datasaurus dozen

Again, we first pass in the dataset to ggplot

```
library(datasauRus)
ggplot(datasaurus_dozen)
```

What's in the data?

Show 10 entries Search:

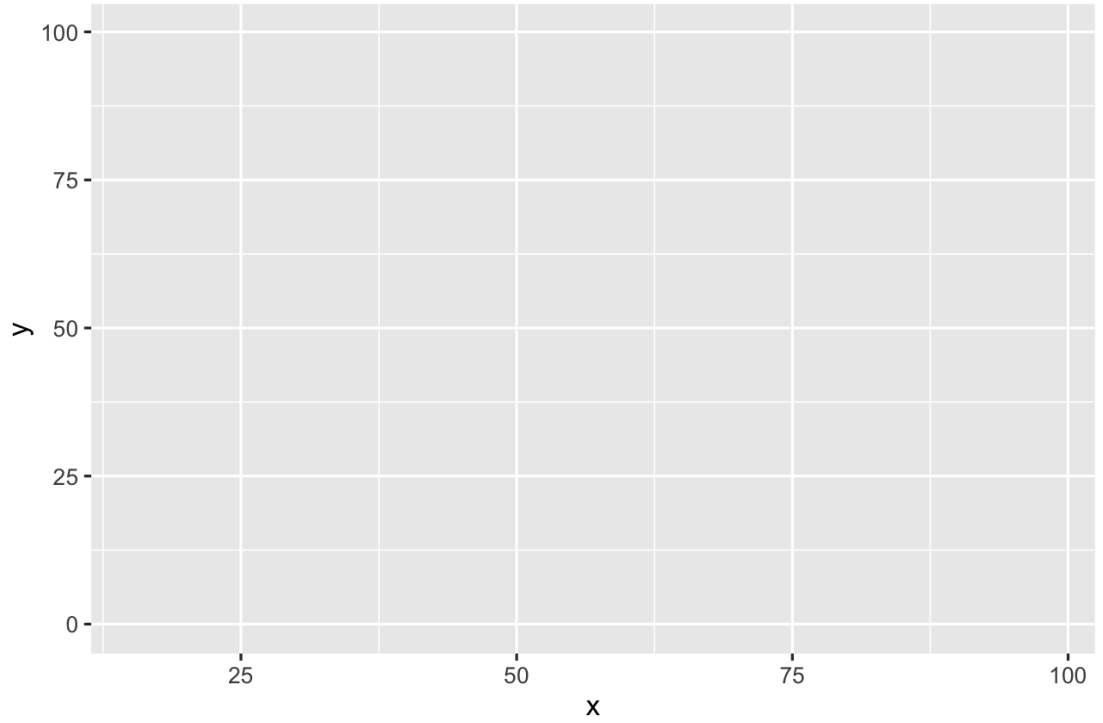
	dataset	x	y
1	dino	55.3846	97.1795
2	dino	51.5385	96.0256
3	dino	46.1538	94.4872
4	dino	42.8205	91.4103
5	dino	40.7692	88.3333
6	dino	38.7179	84.8718
7	dino	35.641	79.8718
8	dino	33.0769	77.5641
9	dino	28.9744	74.4872
10	dino	26.1538	71.4103

Showing 1 to 10 of 1,846 entries Previous 1 2 3 4 5 … 185 Next

A not-so-simple example, the datasaurus dozen

For each dataset we have x and y values, in addition we can map dataset to color

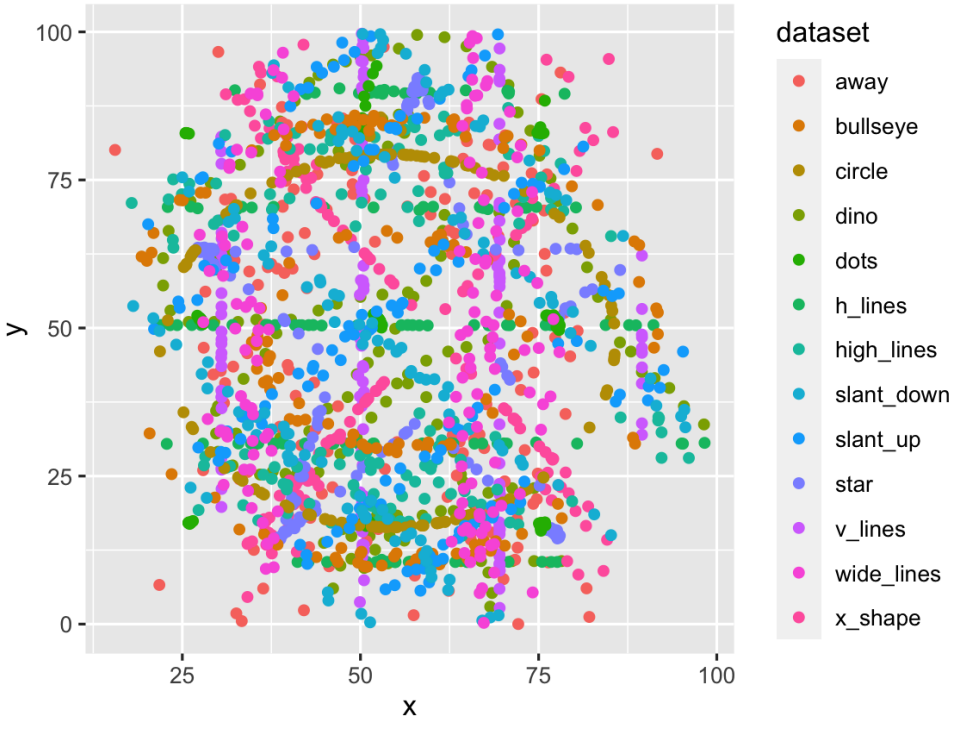
```
ggplot(datasaurus_dozen) +
  aes(x, y, color = dataset)
```



A not-so-simple example, the datasaurus dozen

Trying a simple scatter plot first, but there is too much information

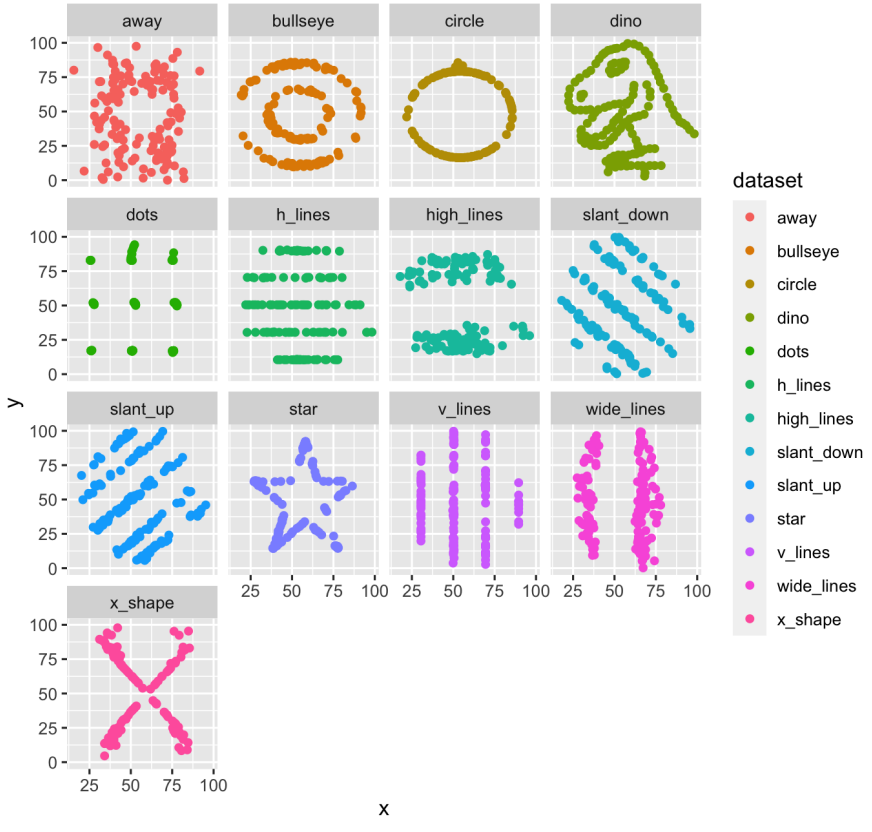
```
ggplot(datasaurus_dozen) +
  aes(x, y, color = dataset) +
  geom_point() +
  theme(aspect.ratio = 1)
```



A not-so-simple example, the datasaurus dozen

We can use facets to split up by dataset, revealing the different distributions

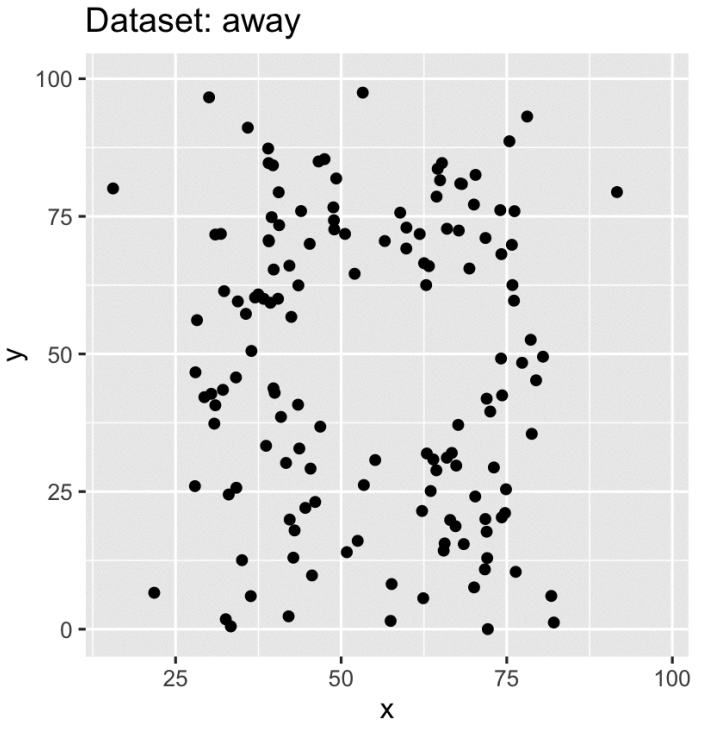
```
ggplot(datasaurus_dozen) +
  aes(x, y, color = dataset) +
  geom_point() +
  facet_wrap(~dataset) +
  theme(aspect.ratio = 1)
```



A not-so-simple example, the datasaurus dozen

We can just as easily turn it into an animation, transitioning between dataset states!

```
ggplot(datasaurus_dozen) +
  aes(x, y) +
  geom_point() +
  transition_states(dataset, 3, 1) +
  labs(title = "Dataset: {closest_state
  theme(aspect.ratio = 1)
```



Resources

Carson Sievert Interactive web-based data visualization with R, plotly, and shiny

website for gganimate

Mitch O'Hara-Wild's tutorial on gganimate

</> Open `part2-exercise-03.Rmd`

15:00

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)
##
```