Advanced data visualization with R
Workshop Day 2

Lesson 1: Overview of tools for interactive plots

Presented by Di Cook

Department of Econometrics and Business Statistics

MONASH University

8-9th Dec 2021 @ Statistical Society of Australia | Zoom

Click me: https://ebsmonash.shinyapps.io/VICfire/

Historical locations on fires, and ignition causes, in Victoria over 2000-2019.

Choose year:

2000 2010 2017

2000 2004 2008 2012 2016 2019

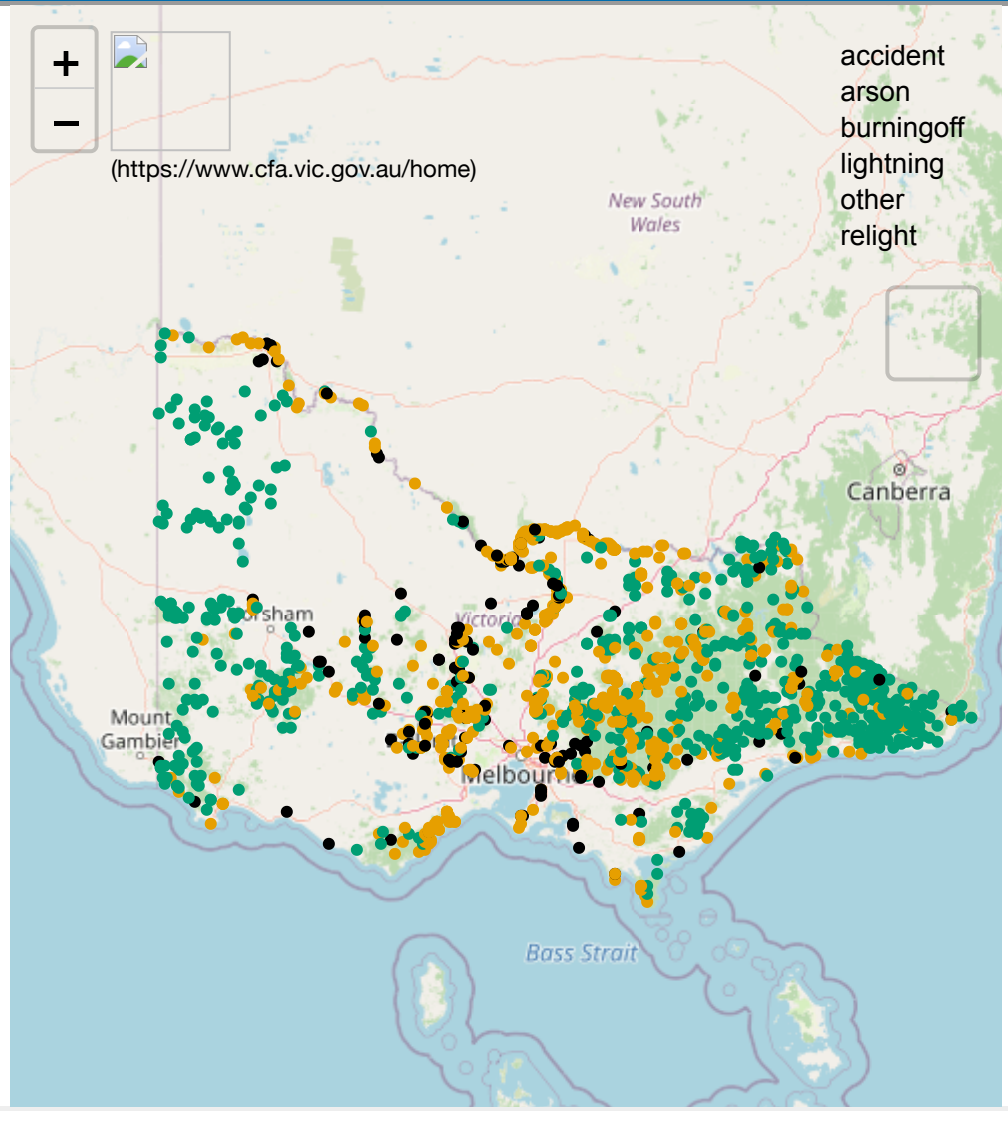
Choose month:

Jan	Feb	Mar
Apr	May	Jun
Jul	Aug	Sep
Oct	Nov	Dec

Choose reason:

- [x] accident
- [x] arson
- [ ] burningoff
- [x] lightning
- [ ] other
- [ ] relight

Disconnected from the server.
Reload



About

This Shiny App helps visualise fires in Victoria for last two decades. YAfter choose the year, month, and ignition reason. The fires match these condition will automatically shown on the map. Due to the limitation of the package, the density plot cannot be refreshed automatically. Each time you change the conditions, you have to clear and re-plot the densitiy plot. By clicking a fire on the map, relevant infomation will pop up and weather infomation will be shown below.

Landscape of interactive plotting packages

Applying interactivity

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 EASY
- Pan/zoom allows re-focusing attention EASY
- Selection allows focusing attention MODERATE
- Linking connects elements from multiple plots DIFFICULT
- Sorting re-orders elements, most useful for tables EASY
- Transforming may be changing bin sizes in a histogram, or applying a log scale, or using a "fish eye lens" DIFFICULT & RARE

Direct manipulation vs GUI

Direct manipulation

When these actions

- Mouse-over
- Pan/zoom
- Selection
- Linking
- Sorting
- Transforming

are taken by the user directly on the plot itself.

Graphical User Interface

Elements of the plot changes are made using GUI items like sliders, checkboxes, menus

Getting hands-on with plotly

What is plotly?

`plotly.js` is a javascript plotting library. `plotly` is an R package with a key function `ggplotly()` that converts ggplot objects into a JSON object to be rendered in a web browser by `plotly.js`.

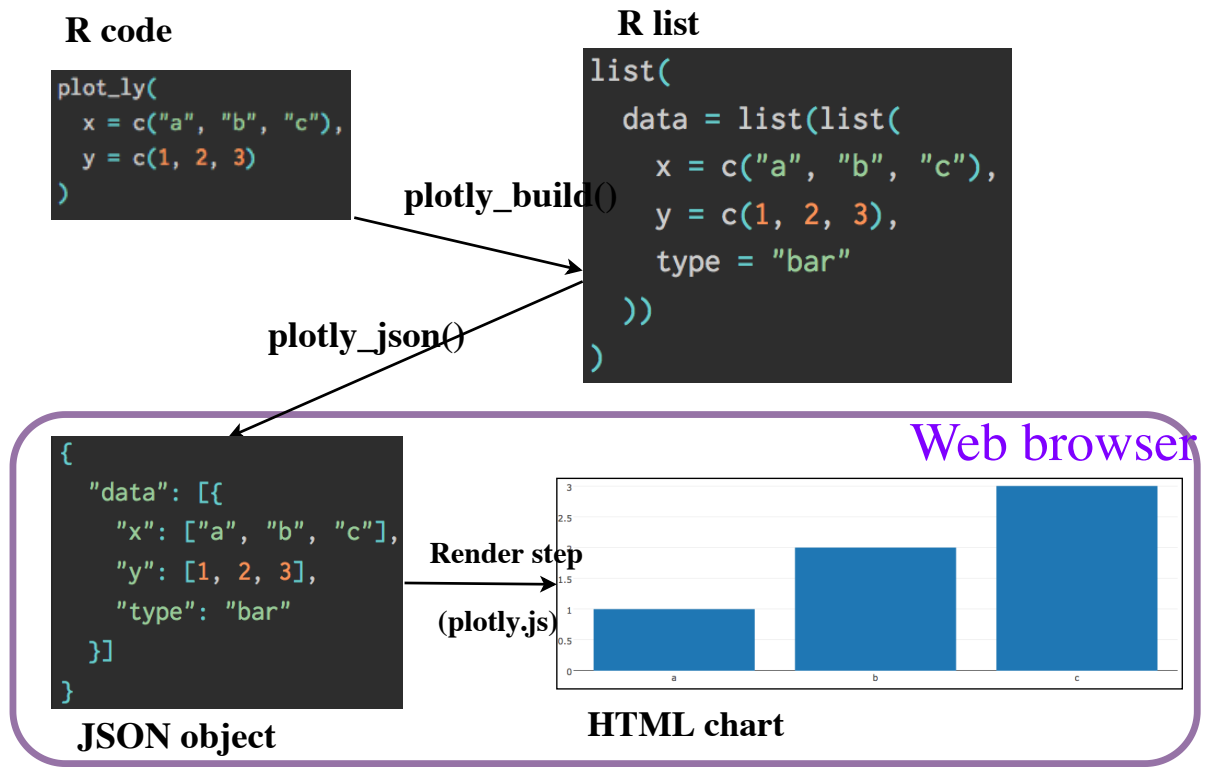


Figure 2.5 from Sievert (2019)

demo data for today

Palmer penguins

Show 10 entries Search:

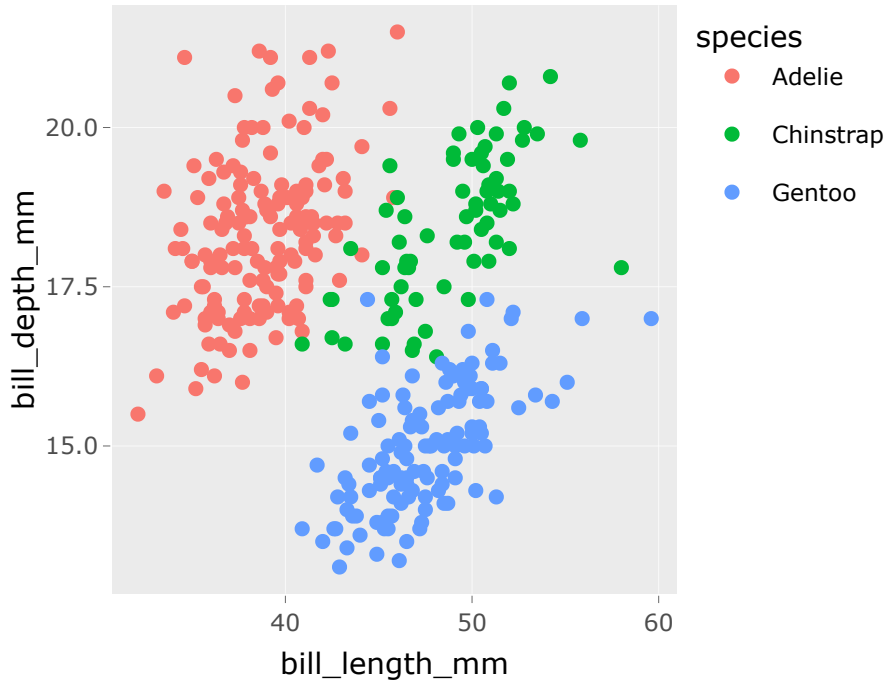
	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
1	Adelie	Torgersen	39.1	18.7	181	3750	male	2007
2	Adelie	Torgersen	39.5	17.4	186	3800	female	2007
3	Adelie	Torgersen	40.3	18	195	3250	female	2007
4	Adelie	Torgersen						2007
5	Adelie	Torgersen	36.7	19.3	193	3450	female	2007
6	Adelie	Torgersen	39.3	20.6	190	3650	male	2007
7	Adelie	Torgersen	38.9	17.8	181	3625	female	2007
8	Adelie	Torgersen	39.2	19.6	195	4675	male	2007
9	Adelie	Torgersen	34.1	18.1	193	3475		2007
10	Adelie	Torgersen	42	20.2	190	4250		2007

Showing 1 to 10 of 344 entries

Previous 1 2 3 4 5 ... 35 Next

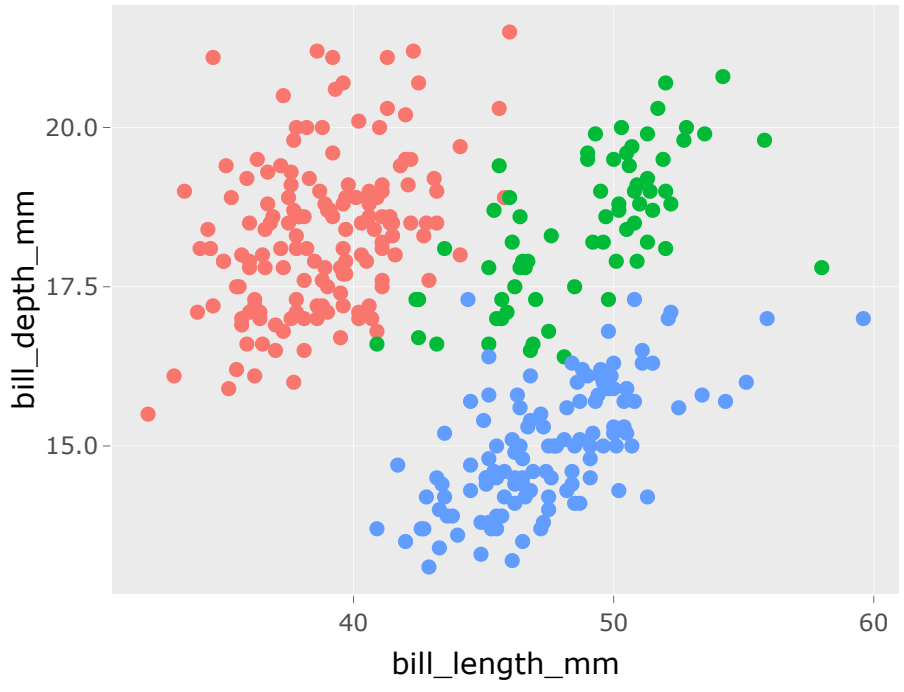
Layering interactivity to a ggplot

```
p <- ggplot(penguins,
       aes(x = bill_length_mm,
           y = bill_depth_mm,
           color = species)) +
  geom_point()

ggplotly(p)
```



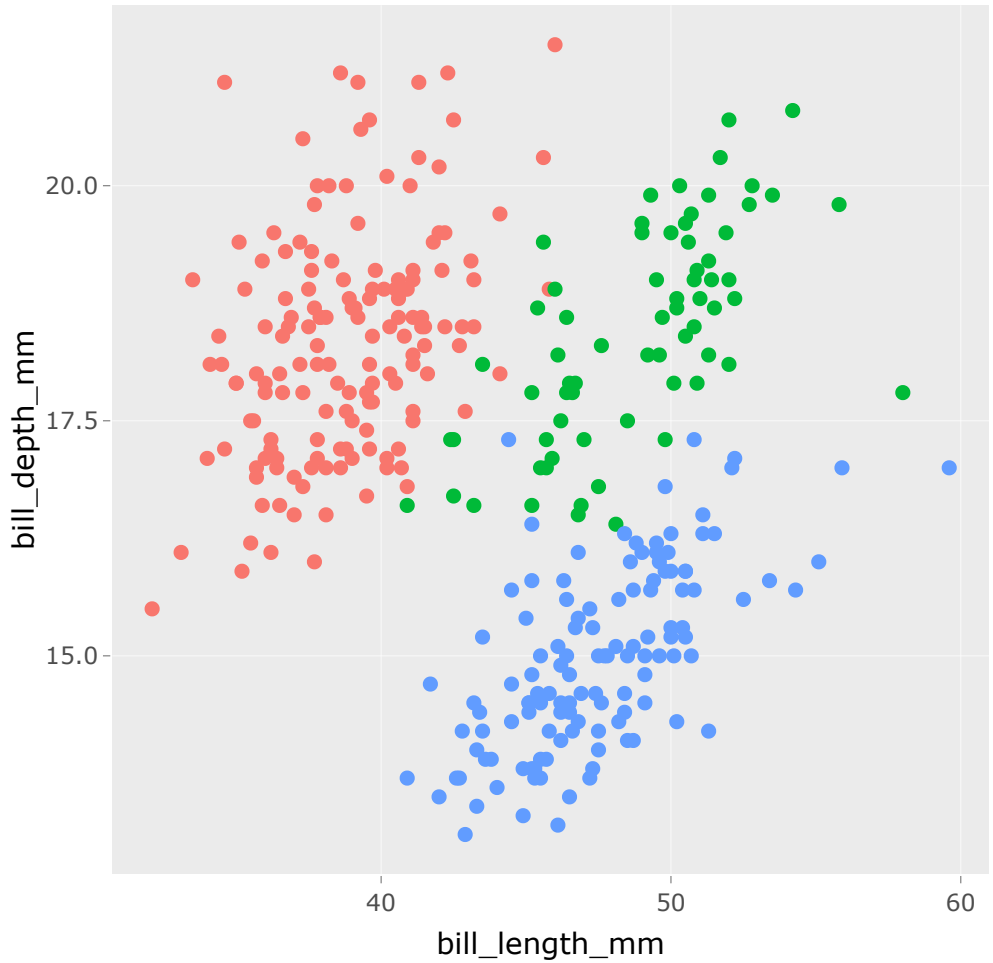
Aspect ratio, and remove legend

```
p2 <- ggplot(penguins,
       aes(x = bill_length_mm,
           y = bill_depth_mm,
           color = species)) +
  geom_point() +
  theme(legend.position="none",
        aspect.ratio=1)

ggplotly(p2)
```

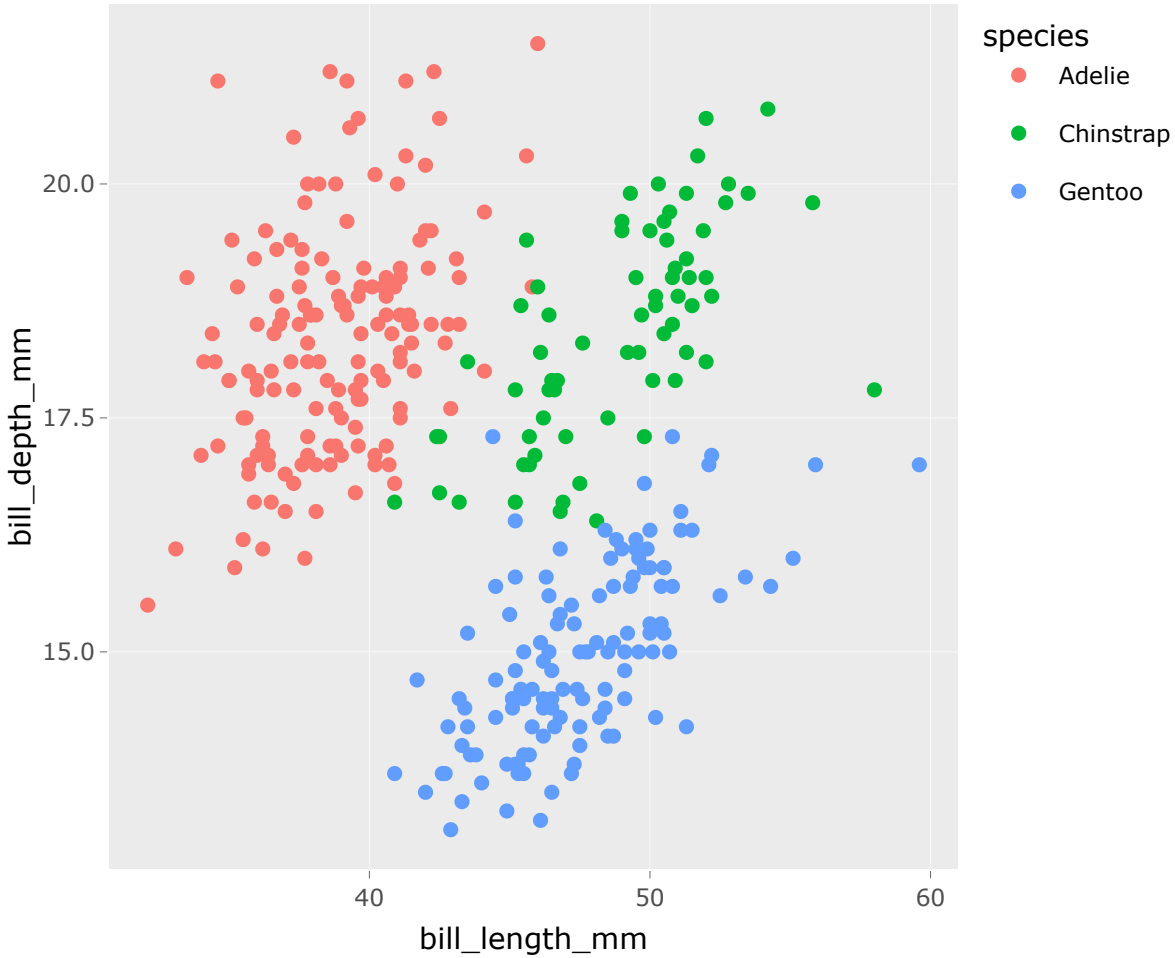


Fix aspect ratio with ggplotly argument

```
p3 <- ggplot(penguins,
       aes(x = bill_length_mm,
           y = bill_depth_mm,
           color = species)) +
  geom_point() +
  theme(legend.position="none",
        aspect.ratio=1)

ggplotly(p3, width=500, height=500)
```



plotly features



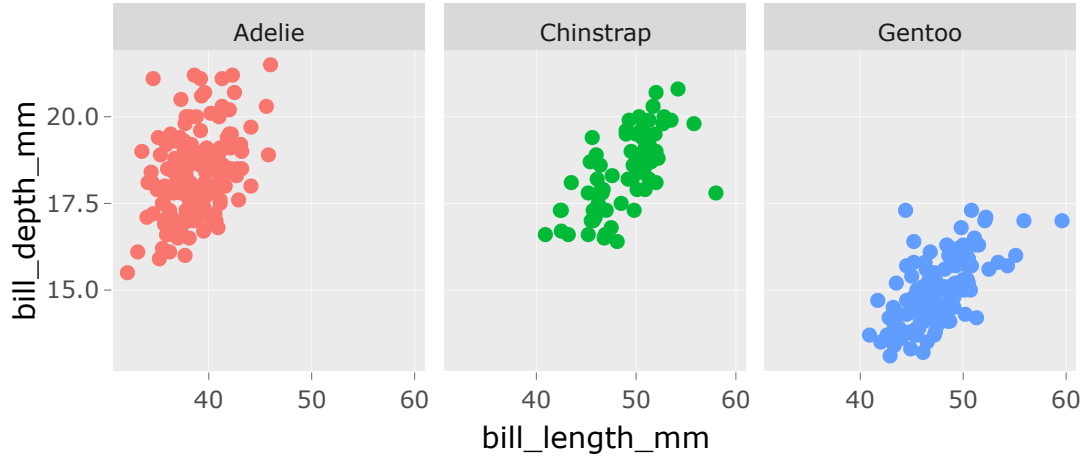
pan/zoom

legend click

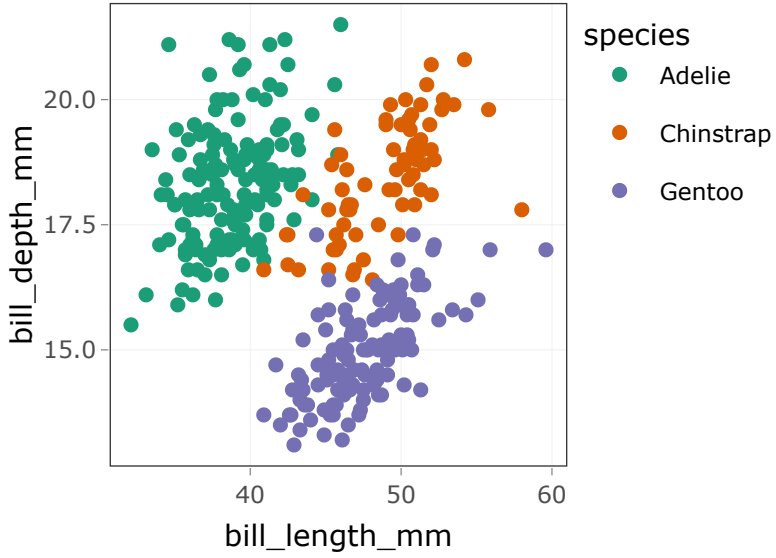
brush

Most but not all ggplot features are supported

```
p5 <- p3 +
  facet_wrap(~species)

ggplotly(p5, width=550, height=250)
```

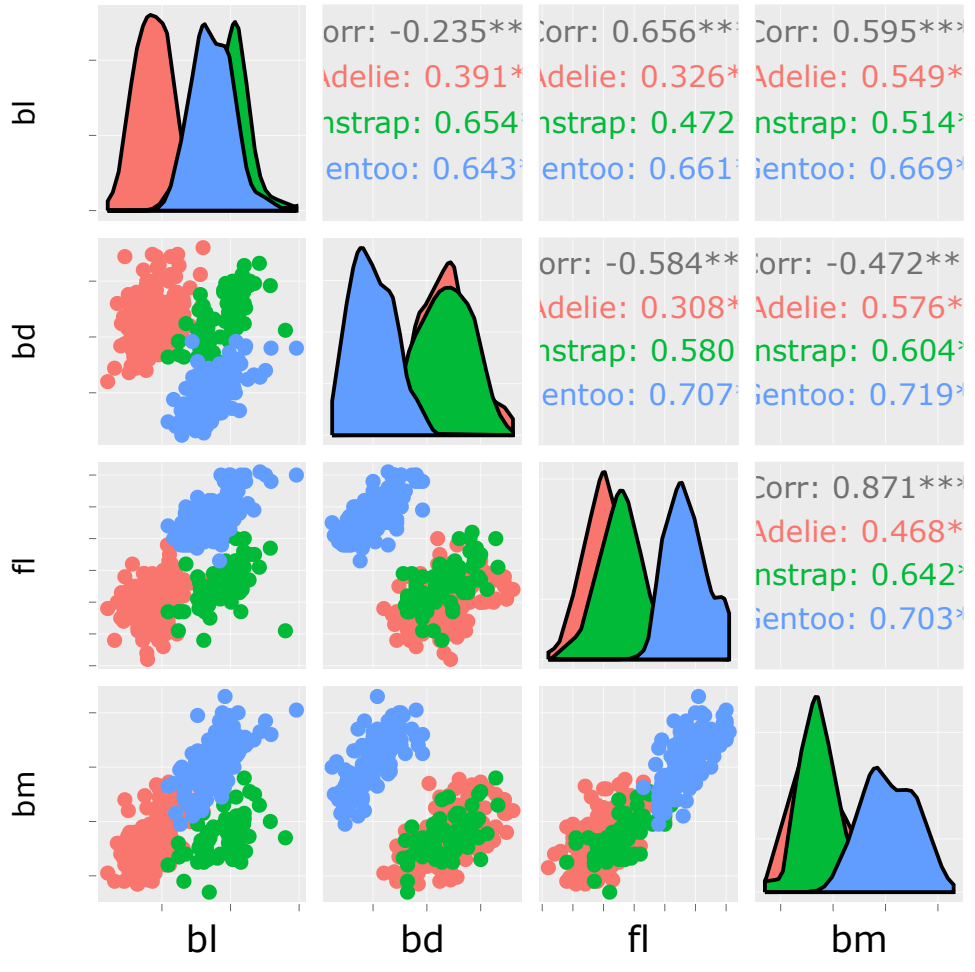



```
p6 <- p3 +
  scale_color_brewer(
    palette = "Dark2") +
  theme_bw()

ggplotly(p6, width=400, height=300)
```



Most ggplot2 extensions mostly work

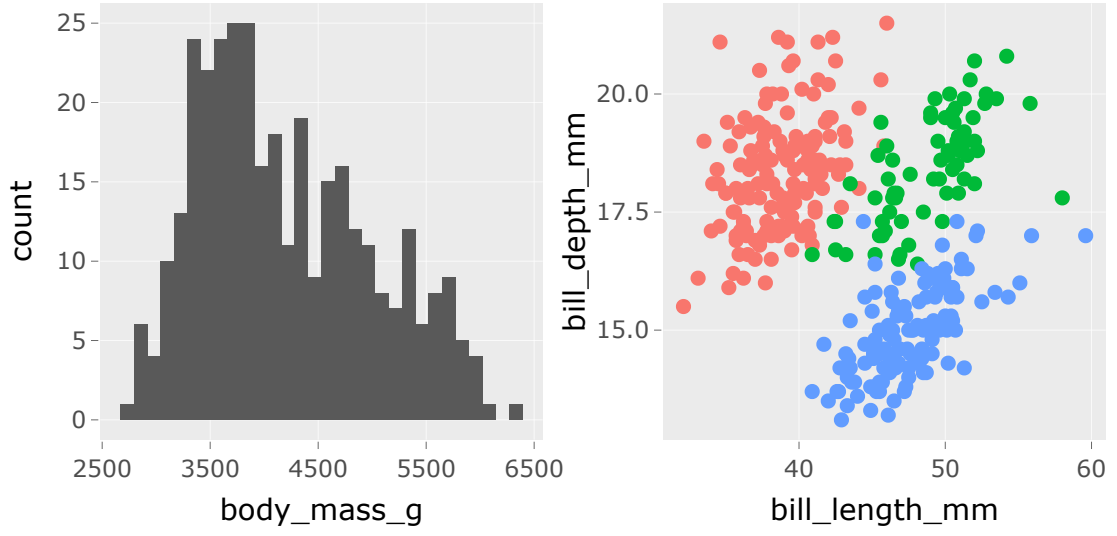
```
library(GGally)
penguins_l <- penguins %>%
  rename(bl = bill_length_mm,
         bd = bill_depth_mm,
         fl = flipper_length_mm,
         bm = body_mass_g)
splom <- ggpairs(penguins_l,
                 columns = 3:6,
                 aes(color = species)) +
  theme(axis.text = element_blank())

ggplotly(splom, width=500, height=500)
```



Multiple plotly plots with subplot

A layout of different plots

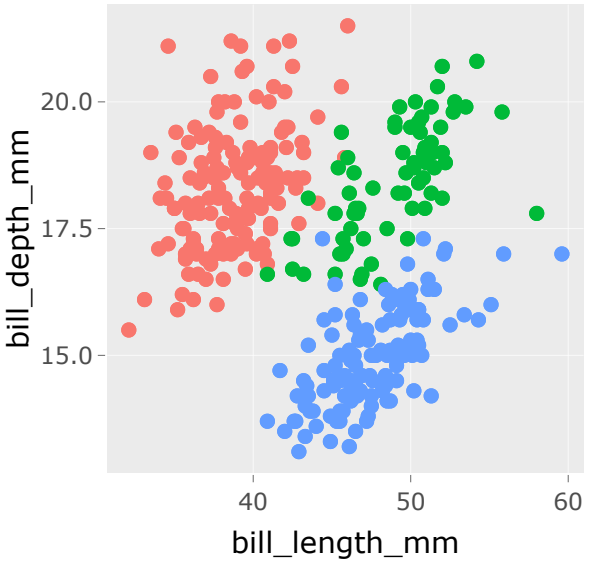
```
p_h <- ggplot(penguins,
          aes(x = body_mass_g)) +
        geom_histogram()

gp_h <- ggplotly(p_h)
gp_s <- ggplotly(p3)

subplot(gp_h,
        gp_s,
        titleX = TRUE,
        titleY = TRUE,
        margin=c(0.06, 0.06, 0, 0),
        nrows = 1)
```



Tooltips can be customised

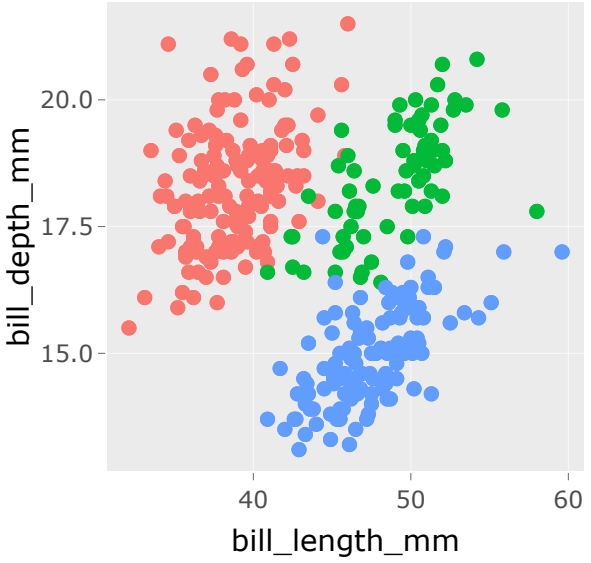
Custom tooltips, from the ggplot

```
p_tooltip <- p3 +
  geom_point(aes(text = island))

ggplotly(p_tooltip,
         width=300, height=300)
```



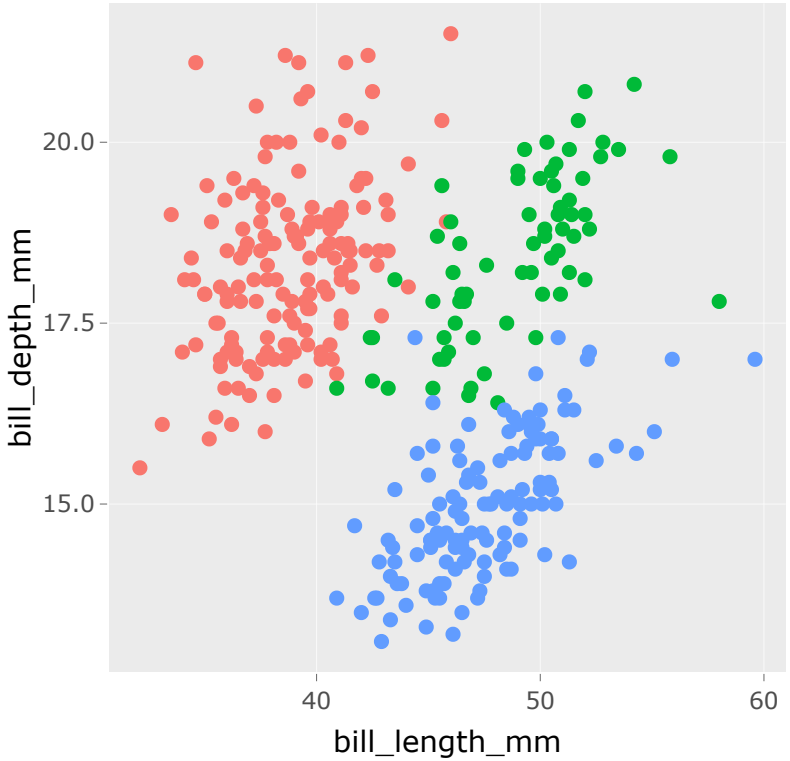
or in ggplotly, either aesthetics or column name in plot data

```
ggplotly(p_tooltip,
         tooltip = "text",
         width=300, height=300)
```



Events can be turned off

```
ggplotly(p3, width=400, height=400) %>
  style(hoverinfo = "none")
```

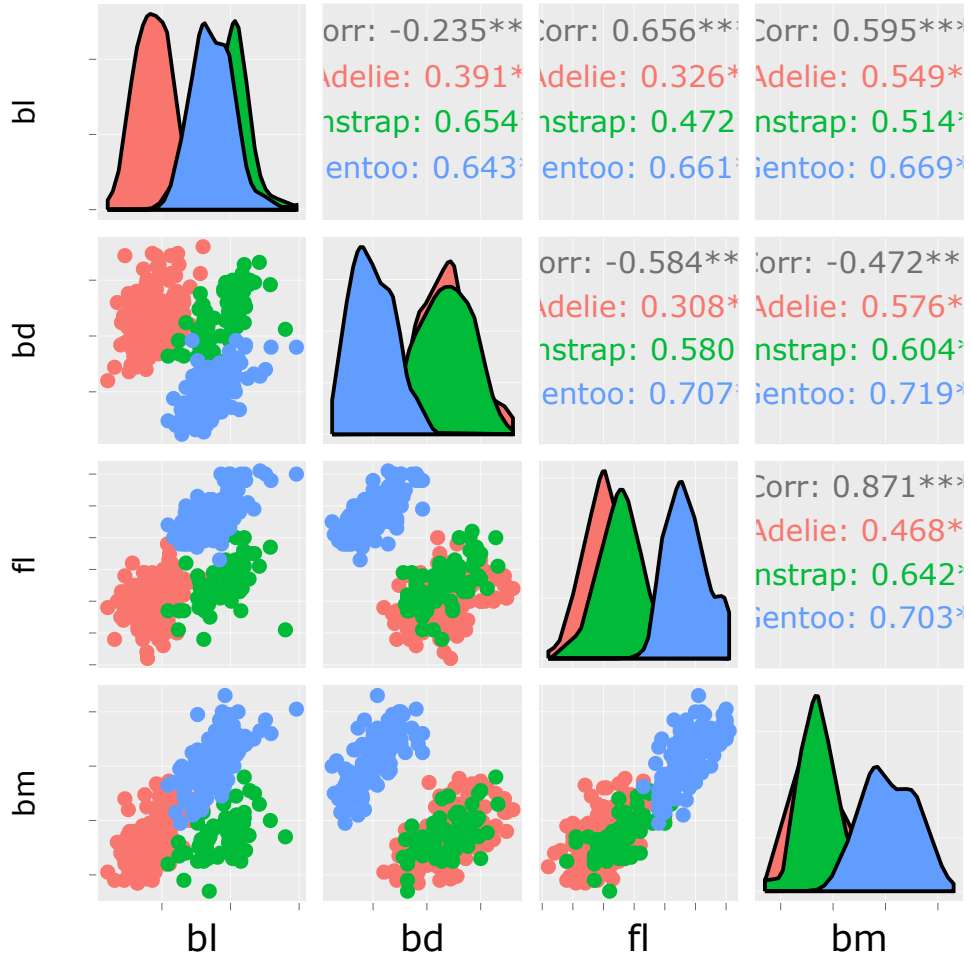


Events can be propagated

Using the `highlight` function with `plotly_selected` shares the selection across views.

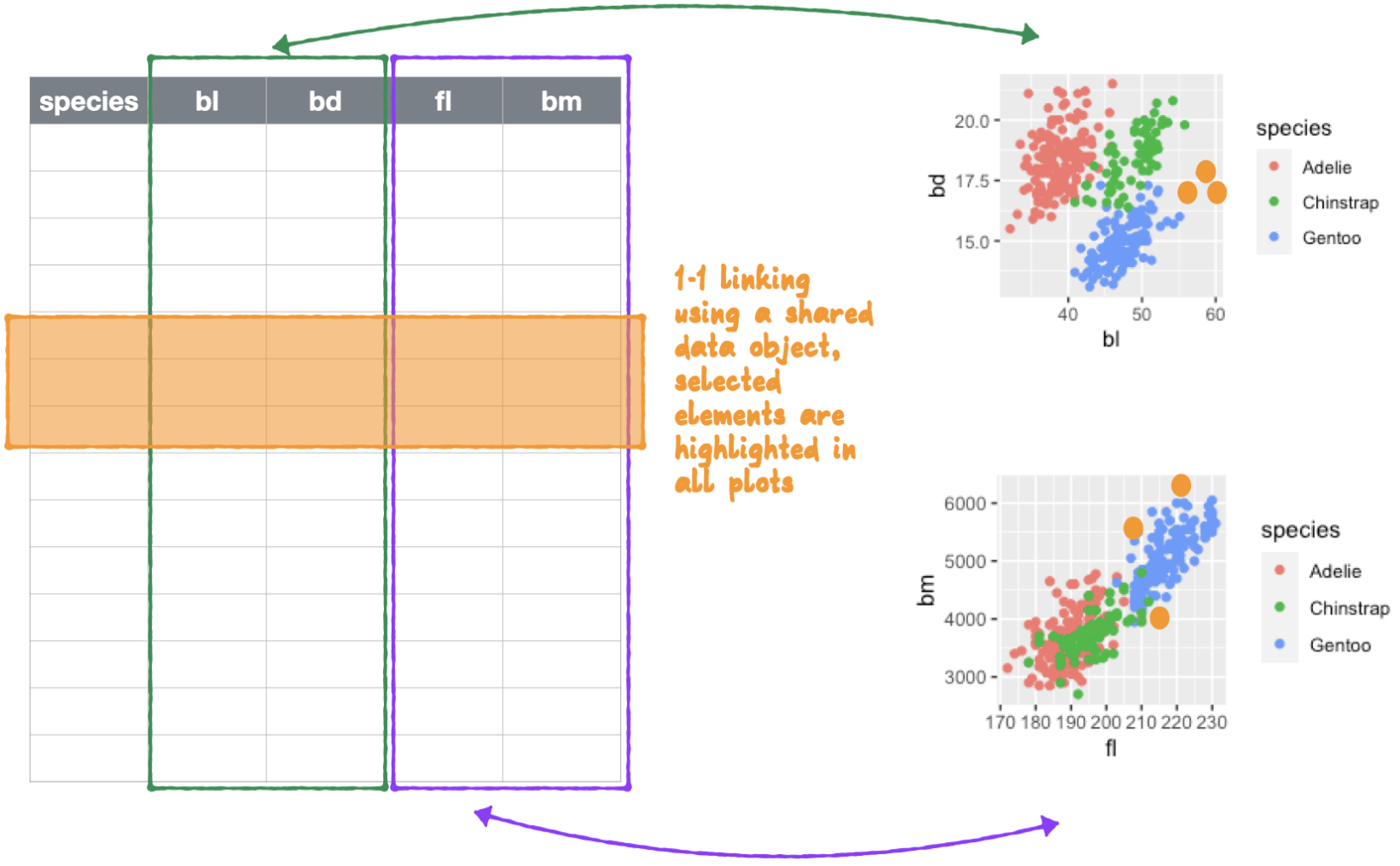
```
ggplotly(splom,
         width=500, height=500) %>%
  highlight(on = "plotly_selected")
```

This is also a first example of linking between plots.



Brush to see how selected points are highlighted in all panels.

How linking works here



Interactive maps

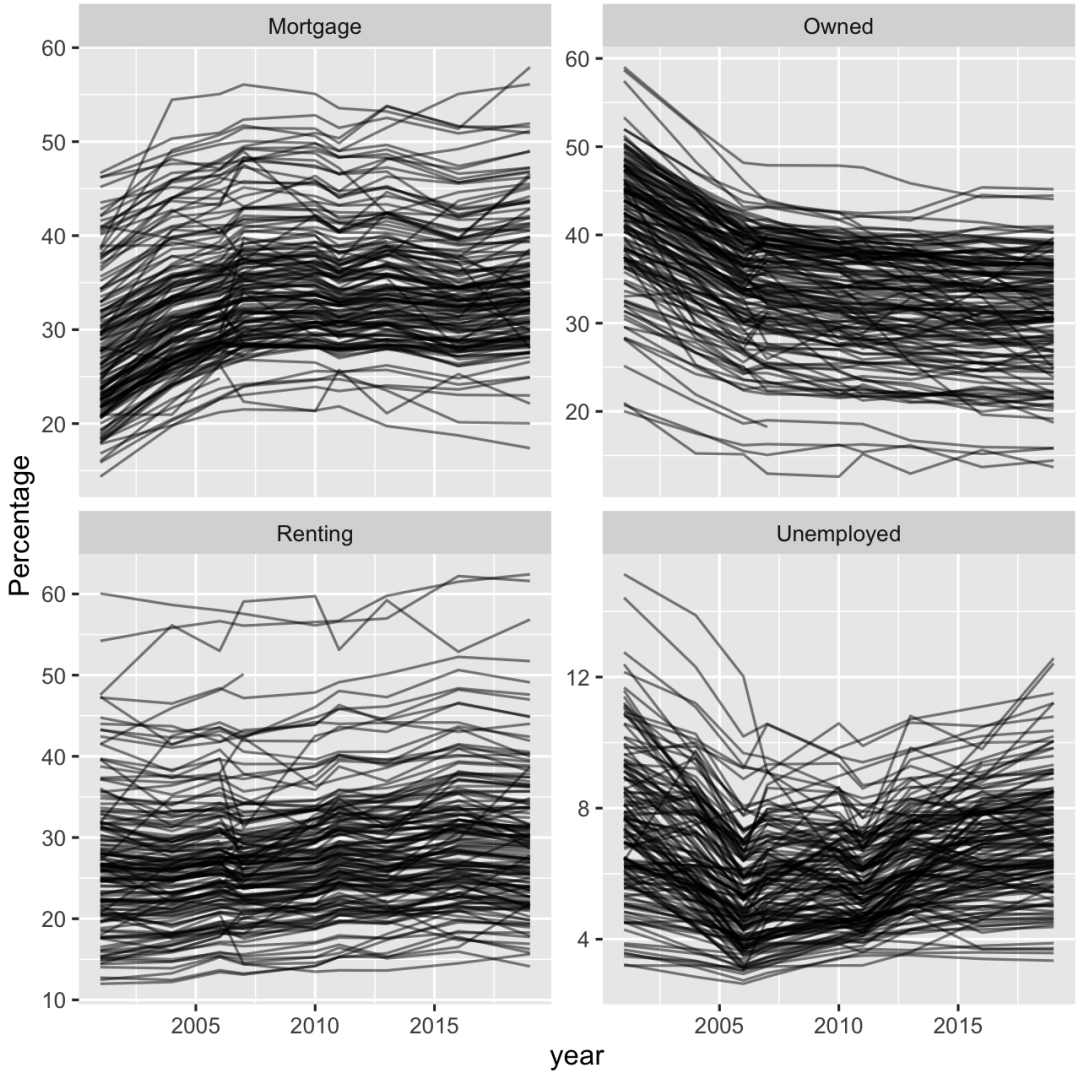
Tips for mapping

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

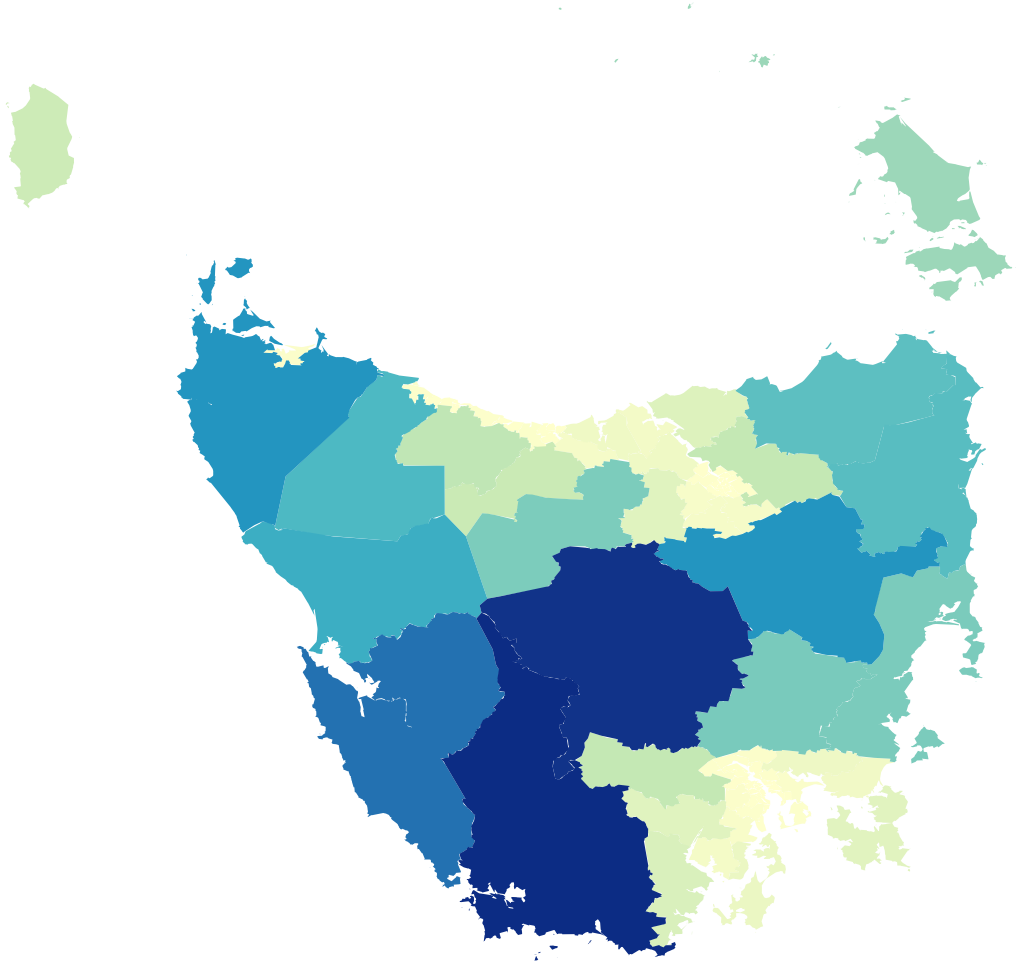
Note: It is important when converting spatial objects from a mapping software to a data analysis project is "thinning" the map to make it smaller and efficient to work with. See the `rmapshapr` package to help with this.

Maps operate with tooltips, too

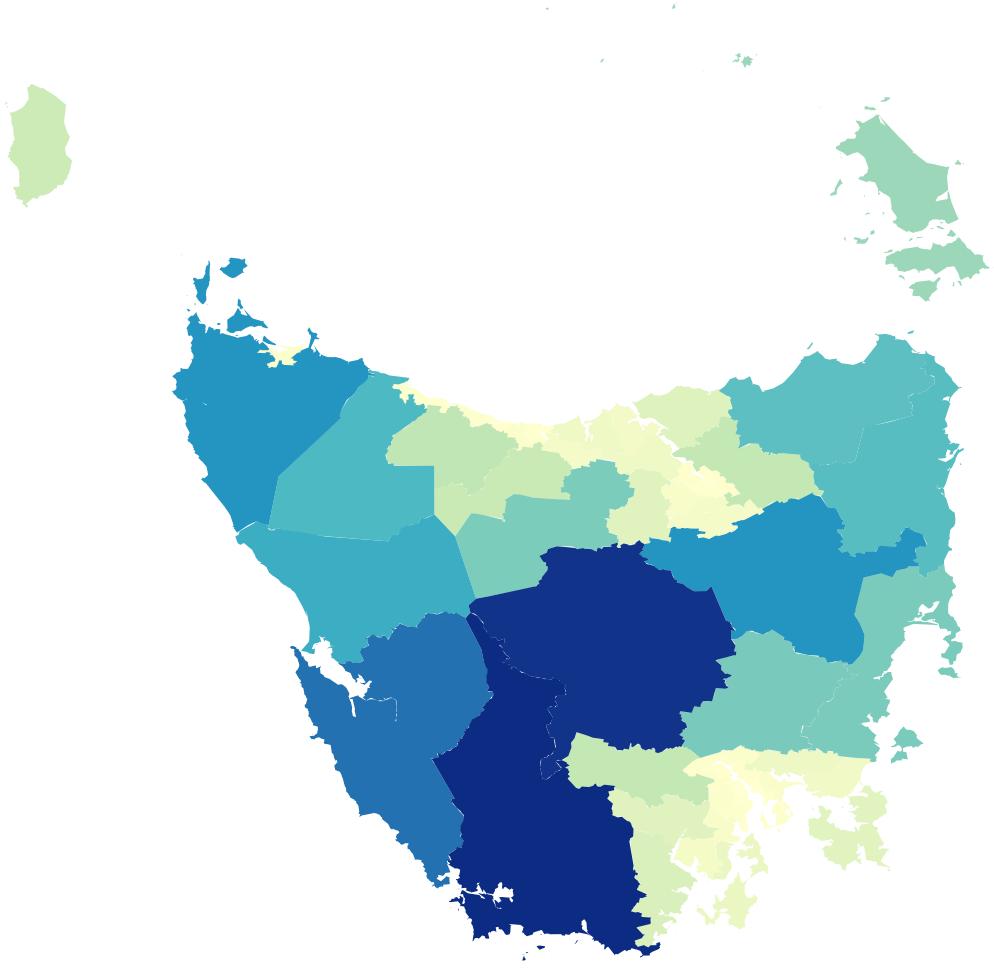
```
library(sugarbag)
library(ggthemes)
tas_tb <- fortify_sfc(tas_sa2)
tas_map <- ggplot(tas_tb,
  mapping = aes(x=long, lat,
    group = interaction(SA2_5DIG16,
                        polygon),
     fill = AREASQKM16),
  colour="white") +
  geom_polygon(aes(text = SA2_NAME16)) +
  scale_fill_distiller("",
    palette="YlGnBu", direction=1) +
  theme_map()
```



```
ggplotly(tas_map +
           theme(legend.position="none")
         tooltip = "text")
```

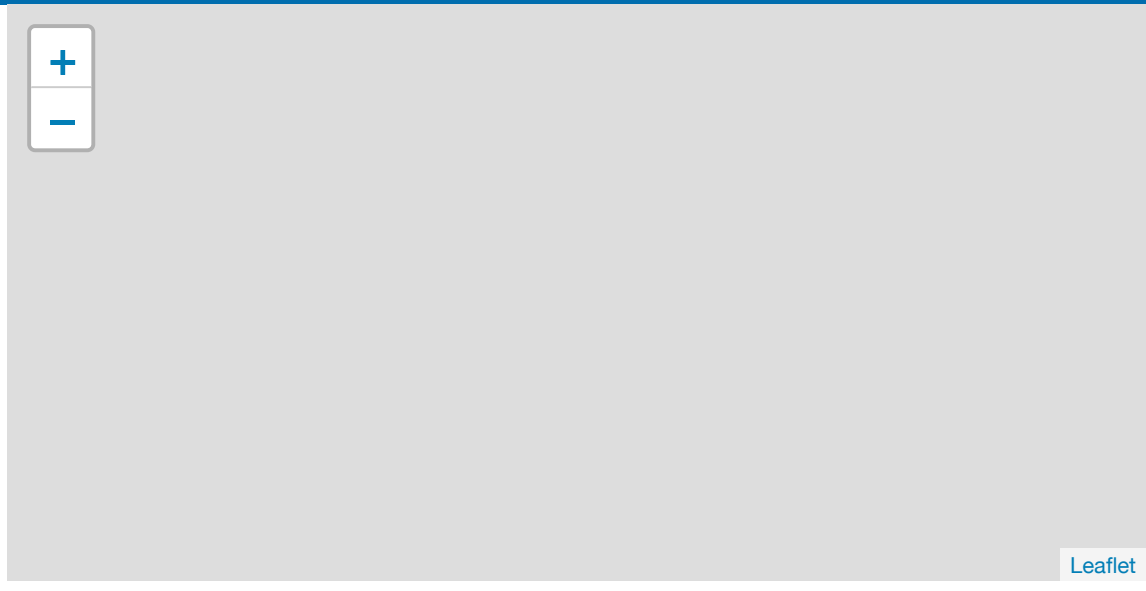


```
ggplotly(tas_map +
           theme(legend.position="none"
         tooltip = c("text", "fill"))
```

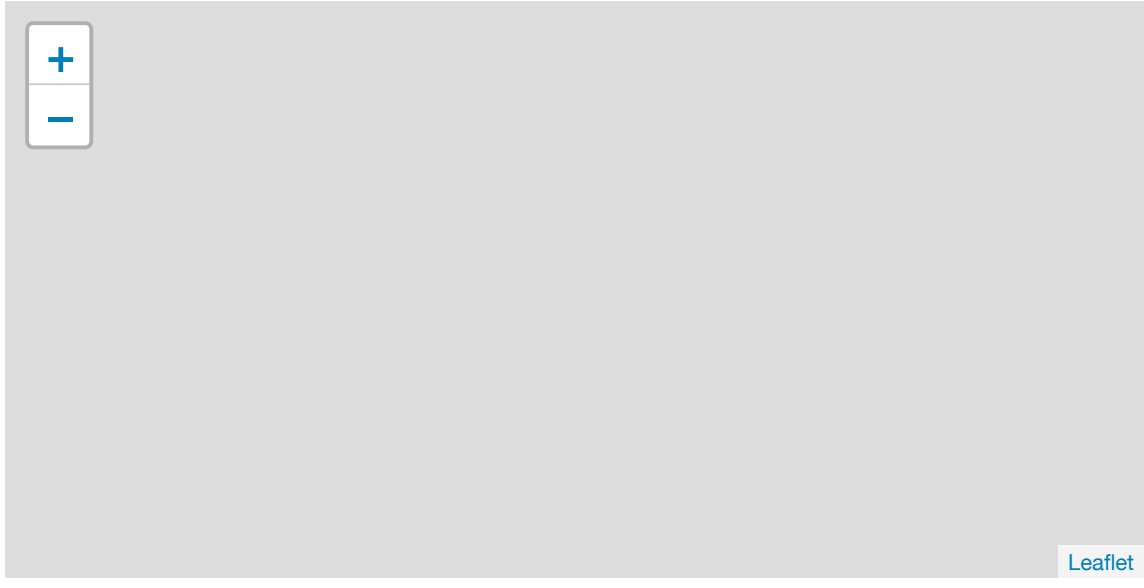


Maps with leaflet

```
library(leaflet)
leaflet(tas_sa2) %>%
  addPolygons(color = "#444444",
              weight = 1,
              smoothFactor = 0.5,
    opacity = 1.0, fillOpacity = 0.5,
    fillColor = ~colorQuantile("YlGnBu",
          c(0, max(AREASQKM16)))(AREASQK
    highlightOptions = highlightOptions(
      color = "white", weight = 2,
      bringToFront = TRUE))
```



```
tas_labels <- sprintf(
  "<strong>%s</strong><br/>%g km<sup>2</
  tas_sa2$SA2_NAME16, tas_sa2$AREASQKM16
) %>% lapply(htmltools::HTML)

leaflet(tas_sa2) %>%
  addPolygons(color = "#444444",
              weight = 1,
              smoothFactor = 0.5,
    opacity = 1.0, fillOpacity = 0.5,
    fillColor = ~colorQuantile("YlGnBu",
          c(0, max(AREASQKM16)))(AREASQK
    highlightOptions = highlightOptions(
      color = "white", weight = 2,
      bringToFront = TRUE),
    label = tas_labels,
    labelOptions = labelOptions(
      style = list("font-weight" = "norm
                   padding = "3px 8px"),
    textsize = "15px",
    direction = "auto"))
```



Reflection on leaflet

Advantages

fast, scalable, reliable

many map formats

Disadvantages

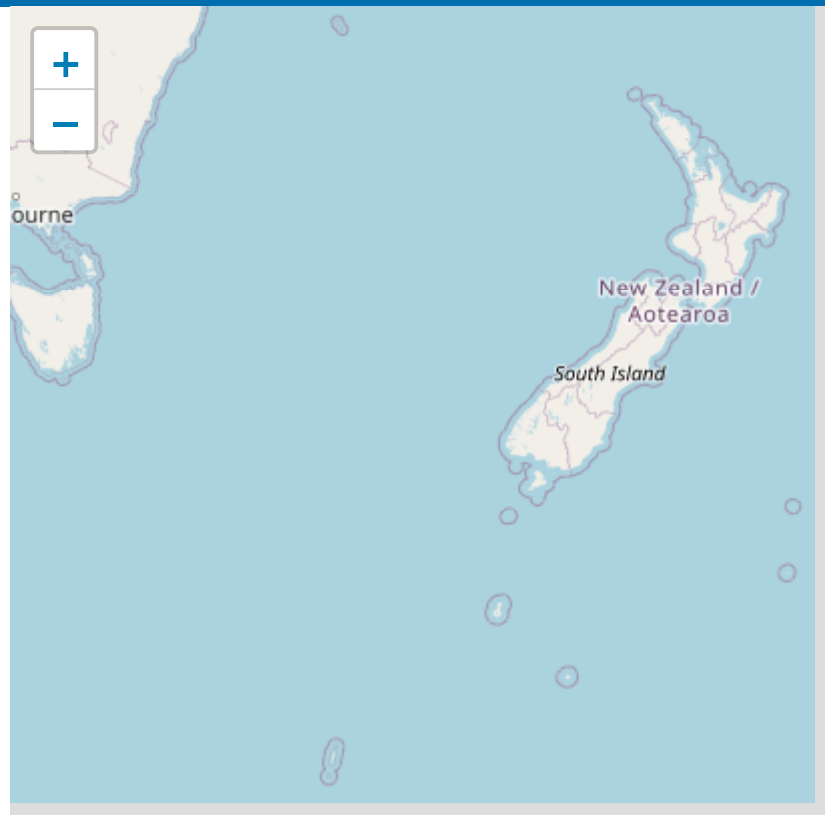
specialist syntax

limited capabilities

map is fast but overlays, like polygons are slow

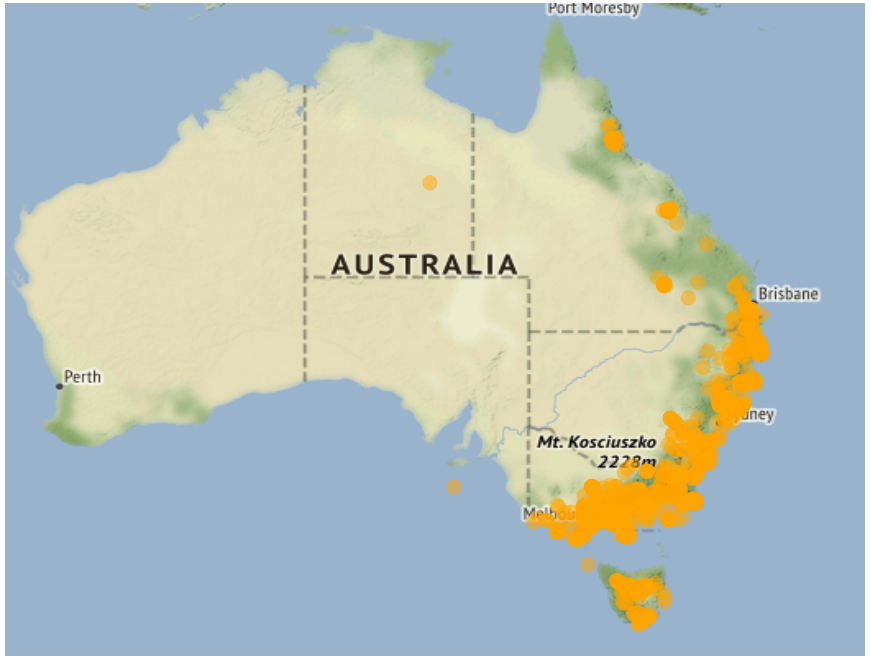
Maps as background: leaflet

```
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)
```



Maps as background: ggmap/plotly

```
# Use ggmap::get_map() to download an OS
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")
```



Interactive tables

Tables with DT

```
penguins %>%
  DT::datatable(width=1150, height=100)
```

Show 10 entries Search:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
1	Adelie	Torgersen	39.1	18.7	181	3750	male	2007
2	Adelie	Torgersen	39.5	17.4	186	3800	female	2007
3	Adelie	Torgersen	40.3	18	195	3250	female	2007
4	Adelie	Torgersen						2007
5	Adelie	Torgersen	36.7	19.3	193	3450	female	2007
6	Adelie	Torgersen	39.3	20.6	190	3650	male	2007
7	Adelie	Torgersen	38.9	17.8	181	3625	female	2007
8	Adelie	Torgersen	39.2	19.6	195	4675	male	2007
9	Adelie	Torgersen	34.1	18.1	193	3475		2007
10	Adelie	Torgersen	42	20.2	190	4250		2007

Showing 1 to 10 of 344 entries Previous 1 2 3 4 5 ... 35 Next

Tables with reactable

species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
Adelie	Torgersen	39.1	18.7	181	3750	male	2007
Adelie	Torgersen	39.5	17.4	186	3800	female	2007
Adelie	Torgersen	40.3	18	195	3250	female	2007
Adelie	Torgersen						2007
Adelie	Torgersen	36.7	19.3	193	3450	female	2007
Adelie	Torgersen	39.3	20.6	190	3650	male	2007
Adelie	Torgersen	38.9	17.8	181	3625	female	2007
Adelie	Torgersen	39.2	19.6	195	4675	male	2007
Adelie	Torgersen	34.1	18.1	193	3475		2007
Adelie	Torgersen	42	20.2	190	4250		2007

1–10 of 344 rows

Previous **1** 2 3 4 5 ... 35 Next

More details of `reactable` can be found at
https://glin.github.io/reactable/index.html

The appearance and display elements in the tables can be modified extensively.

Summary

We've learned basics of interactive plots, primarily using `plotly`.

Keep in mind the variety in types of interactions.

Think about

Interactive plots can be for two very different purposes.

Exploratory

We are yet to understand what the data says. Make available as much of the data as possible, and range of interactions.

Explanatory

We understand the data, and we have identified the key findings that can be communicated to the audience. Here you can optimise the choice of interactions for the best communication.

</> Open `day2-exercise-01.Rmd`

15:00

Learning more

- Sievert (2019) Interactive web-based data visualization with R, plotly, and shiny
- Fidan (2020) Guide to Creating Interactive Maps in R
- RStudio's htmlwidgets gallery
- RStudio's crosstalk guide
- R interface to javascript library DataTables, DT
- Interactive data tables for R, based on React, reactable

Session Information

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

```
## ─ Session info ─────────────────────────────────────────────────────────────
##  setting  value
##  version  R version 4.1.0 (2021-05-18)
##  os       macOS Big Sur 10.16
##  system   x86_64, darwin17.0
##  ui       X11
##  language (EN)
##  collate  en_AU.UTF-8
##  ctype    en_AU.UTF-8
##  tz       Australia/Melbourne
##  date     2021-12-09
##
## ─ Packages ─────────────────────────────────────────────────────────────────
##  package        * version     date       lib
##  anicon           0.1.0       2021-07-14 [1]
##  assertthat       0.2.1       2019-03-21 [1]
##  backports        1.2.1       2020-12-09 [1]
##  base64enc        0.1-3       2015-07-28 [1]
##  bitops           1.0-7       2021-04-24 [1]
##  broom            0.7.9       2021-07-27 [1]
```

These slides are licensed under