Animated graphics in R

Abhijit Dasgupta, PhD

Dynamic graphics

Dynamic graphics in R

There are several package in R that provide dynamic graphics meant to be consumed on the web.

Many of these are wrappers around well-known Javascript libraries like D3.js, leaflet.js and others

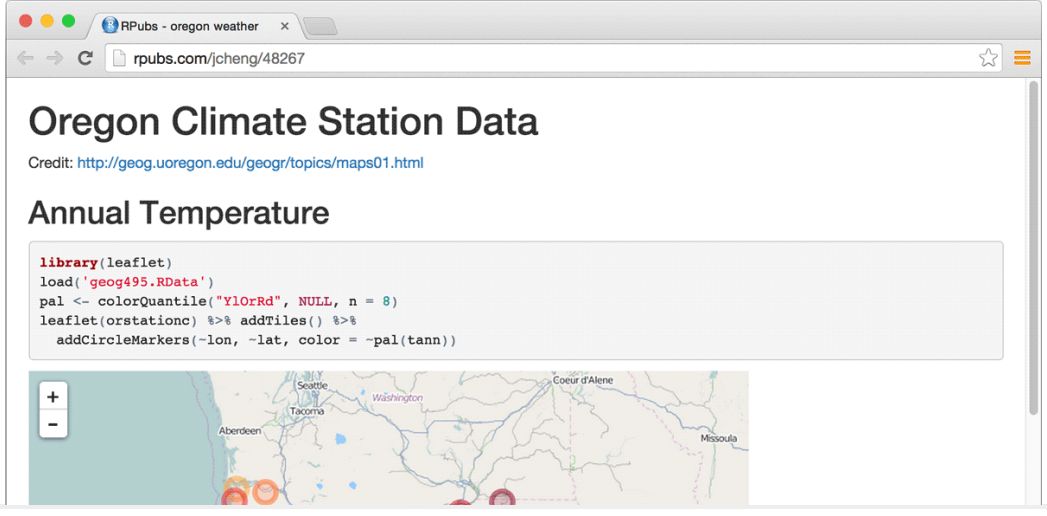
These packages have mostly come under the umbrella of of the `htmlwidgets` package, which allows these HTML-based graphics to be displayed through R and R Markdown

Bring the best of JavaScript data visualization to R

Use JavaScript visualization libraries **at the R console**, just like plots

Embed widgets in **R Markdown** documents and **Shiny** web applications

Develop new widgets using a framework that seamlessly bridges R and JavaScript



Dynamic graphics in R

There are several broad categories of dynamic graphs

General purpose:

Package	Description
r2d3	Interface for D3.js, requires D3.js code
plotly	Interface with plot.ly, direct conversion from ggplot2
highcharter	Using the Highcharts.js package
dygraphs	For time series or longitudinal data

Maps:

```
tribble(
    ~Package, ~Description,
        "leaflet", "Maps using OpenStreetMaps") %>%
    kable() %>%
    kable_styling()
```

Package	Description

Dynamic graphics in R

Networks:

```
tribble(~Package, ~Description,
        'networkD3', 'Dynamic network visualizations using D3',
        'visNetwork', 'Interface to the vis.js pacman::p_load') %>%
    kable() %>% kable_styling()
```

Package	Description
networkD3	Dynamic network visualizations using D3
visNetwork	Interface to the vis.js pacman::p_load

Plotly

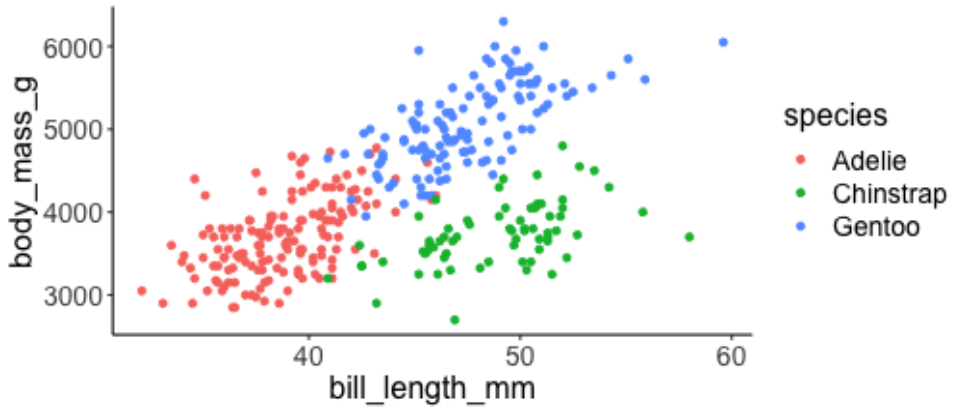
https://plotly.com/graphing-libraries

Plotly

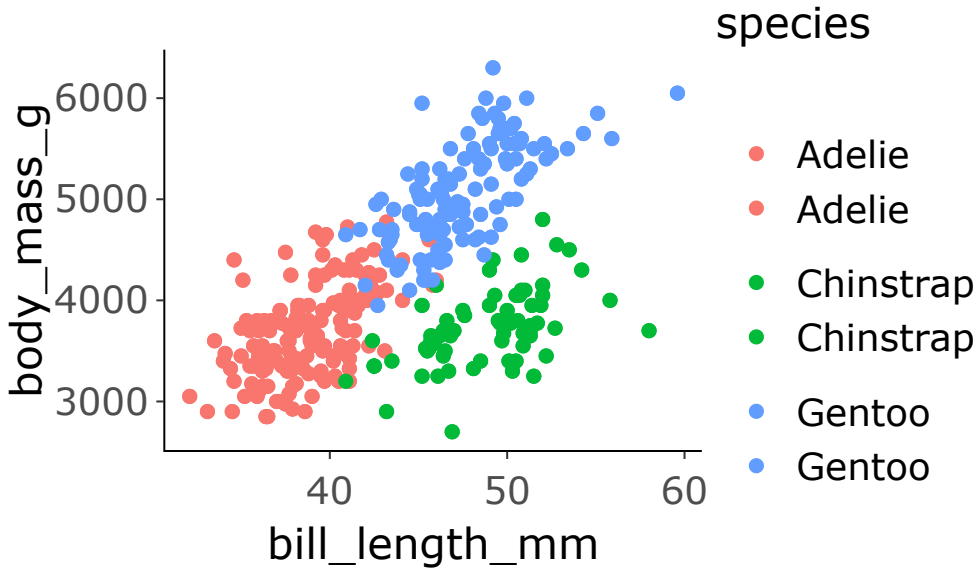
Plotly is a company that developed the `plotly.js` graphing pacman::p_load, as well as packages for R and Python.

For the R package, it developed a turnkey method to convert `ggplot2` graphics into interactive graphs.

```
plt <- ggplot(penguins,
              aes(bill_length_mm, body_mass_g,
                  color = species))+
    geom_point()
plt
```



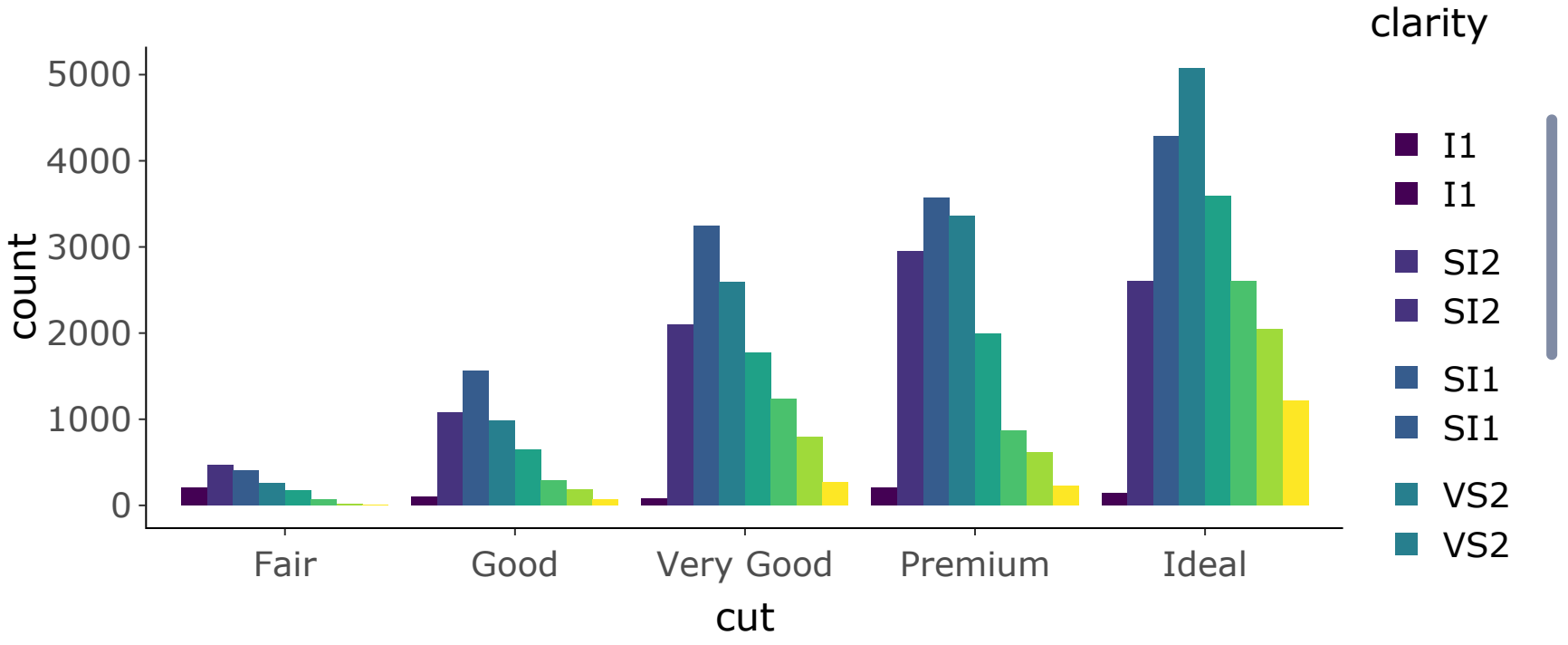

```
ggplotly(plt)
```



Plotly

You can do some customization on the tooltips (what shows up when you put your mouse over a point).

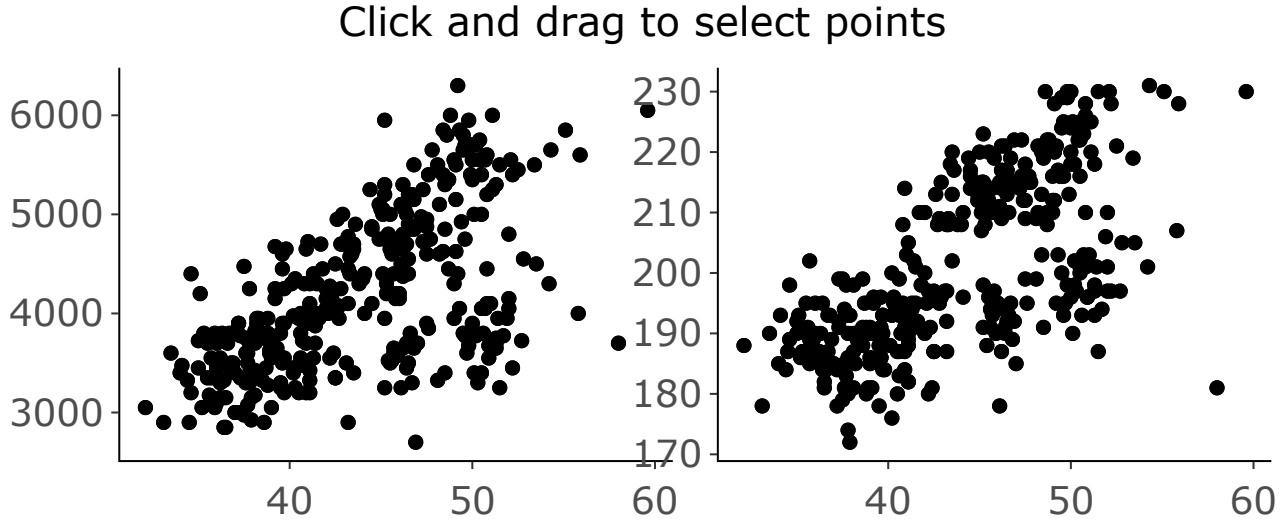
```
p <- ggplot(data = diamonds, aes(x = cut, fill = clarity))+
    geom_bar(position = 'dodge')
ggplotly(p, tooltip = c('cut'))
```



Plotly

You can do linked plots in plotly, so interactions in one plot are reflected in a second plot. This is called *brushing*. The key here is to use `highlight_key`, which allows a data frame to be shared between multiple plots at the same time.

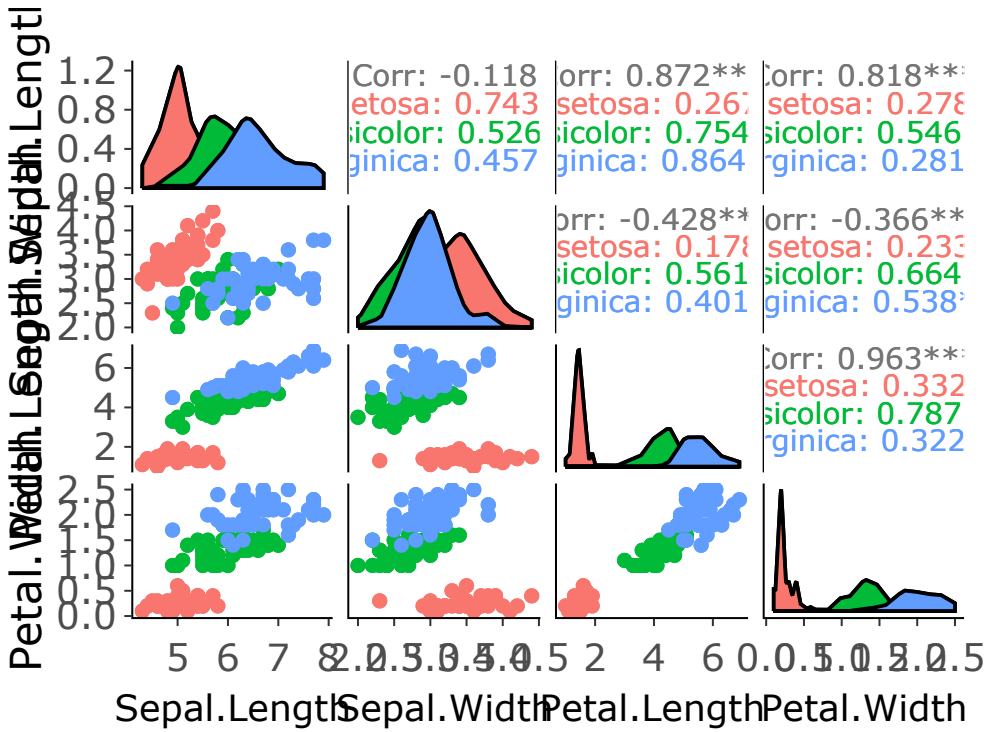
```
d <- highlight_key(penguins)
plt1 <- ggplot(d, aes(x = bill_length_mm, y = body_mass_g))+geom_point()
plt2 <- ggplot(d, aes(x = bill_length_mm, y = flipper_length_mm))+geom_point()
subplot(plt1, plt2) %>%
    layout(title = "Click and drag to select points") %>%
    highlight("plotly_selected")
```



Plotly

You can also do brushing over multiple plots drawn on the same dataset

```
highlight_key(iris) %>%
      GGally::ggpairs(aes(colour = Species), columns = 1:4) %>%
      ggplotly(tooltip = c("x", "y", "colour")) %>%
      highlight("plotly_selected")
```



GGally is a **ggplot2** extension that provides additional composite plot types like the pairs plot we use here.

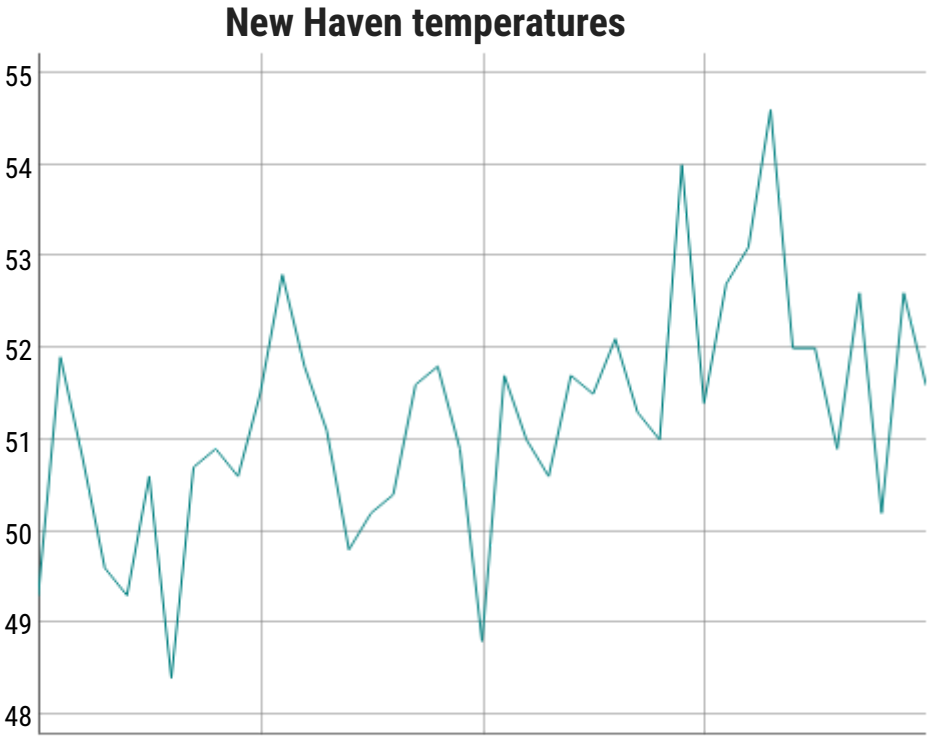
Dygraphs

https://rstudio.github.io/dygraphs

Dygraphs

The *dygraphs* package wraps the `dygraphs` Javascript pacman::p_load, which is build to chart time-series data.

```
pacman::p_load(dygraphs)
dygraph(nhtemp, main = "New Haven temperatures") %>%
    dyRangeSelector(dateWindow = c("1920-01-01","1960-01-01"))
```

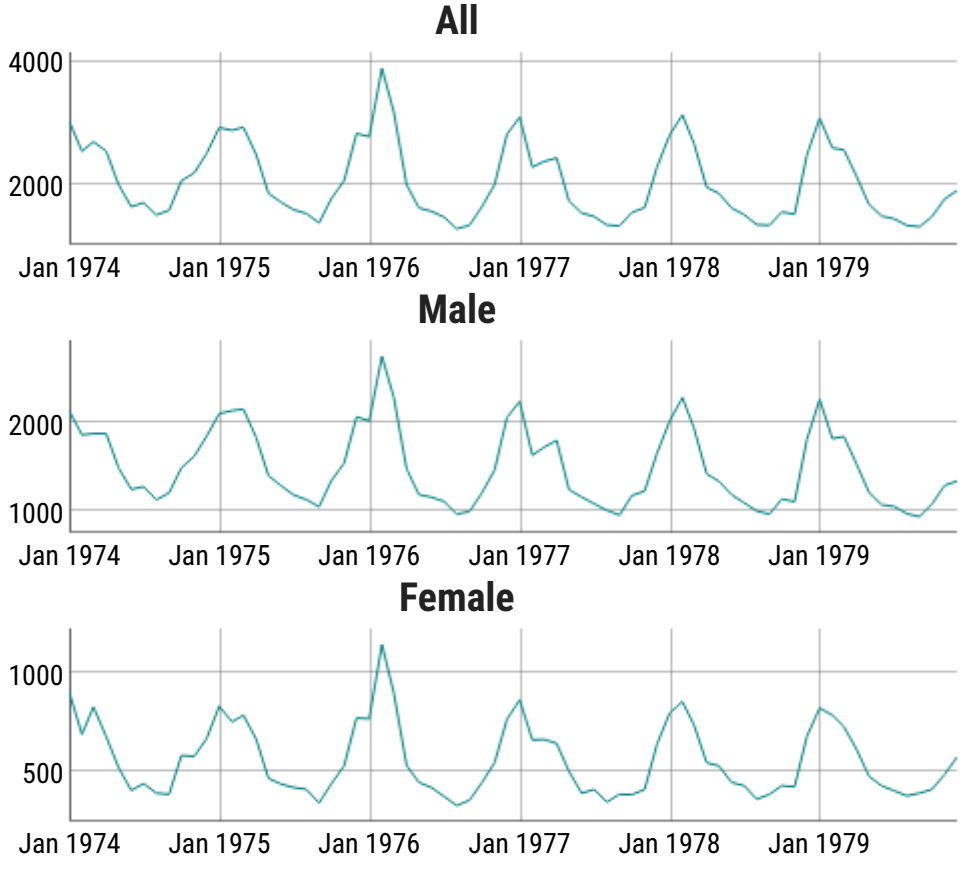


Dygraphs

We can also create multiple linked time series

```
dygraph(ldeaths, main = "All",
        group = "lung-deaths")
dygraph(mdeaths, main = "Male",
        group = "lung-deaths")
dygraph(fdeaths, main = "Female",
        group = "lung-deaths")
```

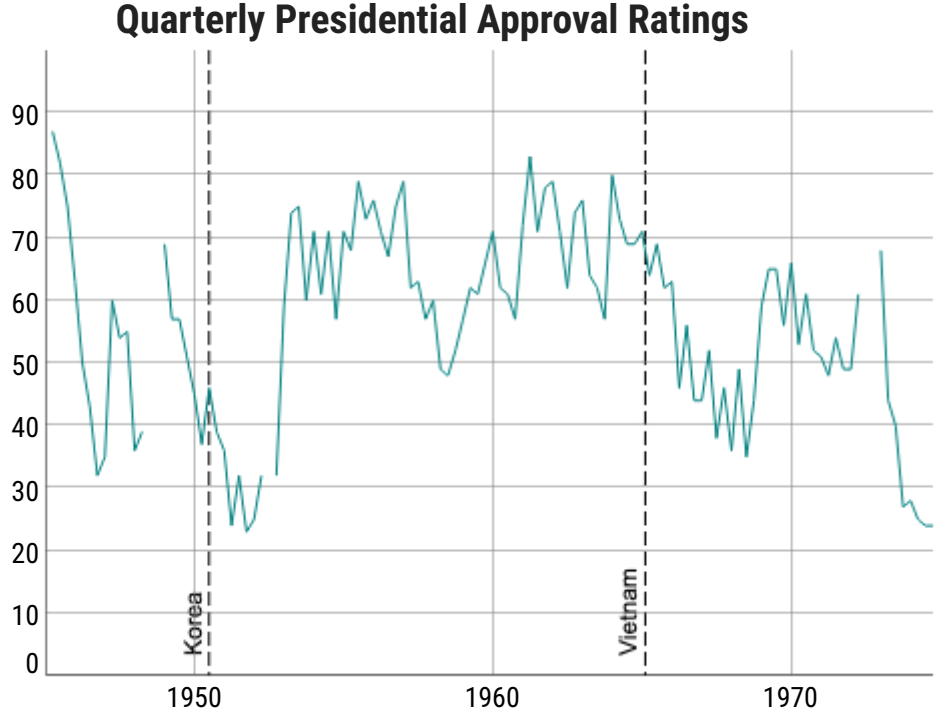
You could name the group anything you like, as long as it's the same across the plots.



Dygraphs

You could also annotate dygraphs

```
dygraph(presidents, main = "Quarterly Presidential Approval Ratings") %>%
  dyAxis("y", valueRange = c(0, 100)) %>%
  dyEvent("1950-6-30", "Korea", labelLoc = "bottom") %>%
  dyEvent("1965-2-09", "Vietnam", labelLoc = "bottom")
```



Highcharter

https://jkunst.com/highcharter/index.html

Highcharter

- The **highcharter** package provides a rich set of plotting functions for dynamic graphics
- The R package is, in spirit, similar to **ggplot2**
  - It uses *mappings*, *aesthetics* and *geometries* (called *types*)
  - Customization can be added using additional functions in a pipe
  - Very rich set of chart types

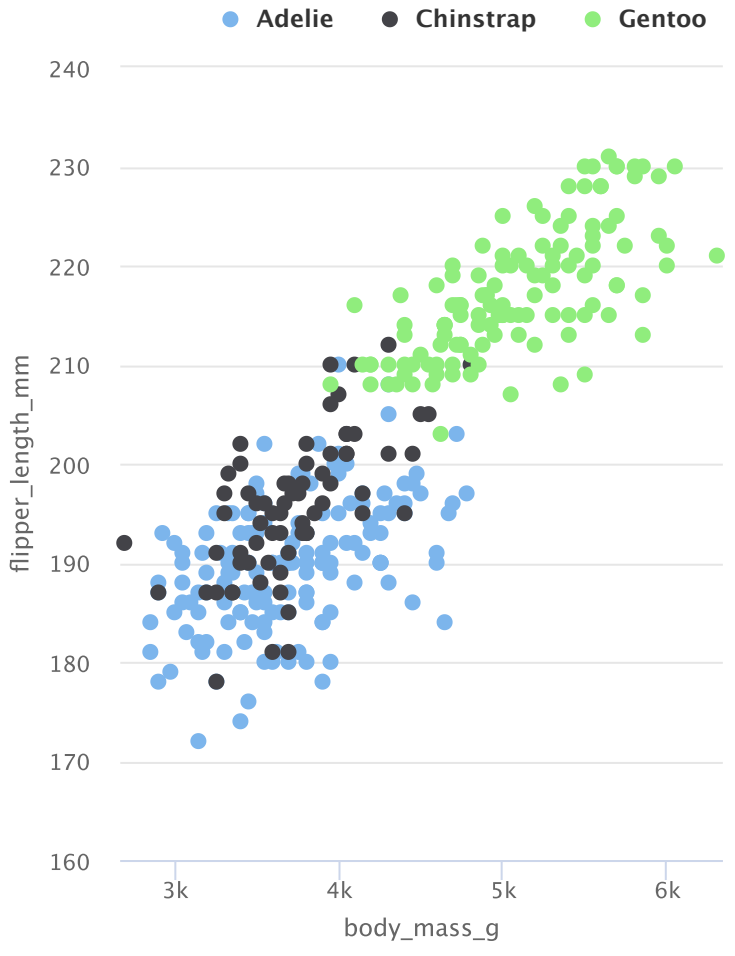
Highcharter

```
pacman::p_load(highcharter)

hchart(object=penguins,
       type="scatter",
       mapping = hcaes(x = body_mass_g,
                       y = flipper_length_mm ,
                       group = species)) %>%
    hc_legend(align='right',
              verticalAlign='top')
```

See how the elements are similar to what you'd put in a **ggplot2** pipe, except they are in a single function

Options can be added with pipes



Highcharter

Unlike **ggplot2**, the object to be plotted doesn't have to be a data frame

```
hc1=hchart(diamonds$cut, type='column')
hc1
```

```
hc1=hchart(diamonds$cut, type='column')
hc1
```

Highcharter

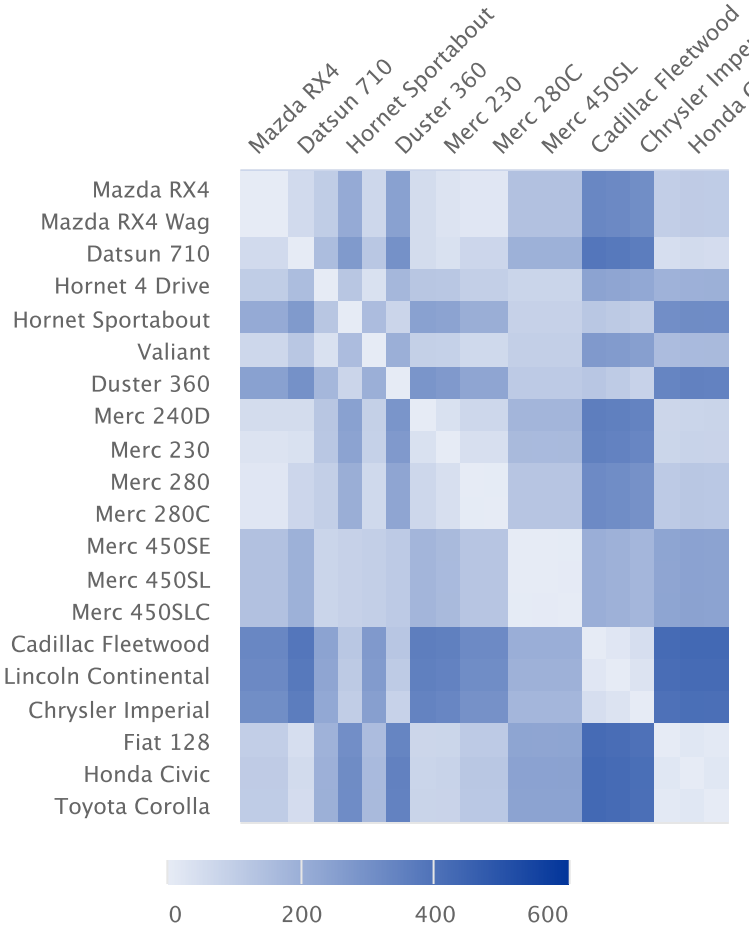
We can make some more complex plots using **highcharter**

```
pacman::p_load(survival)
pacman::p_load(widgetframe)

data(lung)
lung <- dplyr::mutate(lung,
                      sex = ifelse(sex == 1, "Male",
fit <- survfit(Surv(time, status) ~ sex, data = lung)
hchart(fit, ranges=TRUE)
```

Highcharter

```
pacman::p_load(highcharter)
mtcars2 <- mtcars[1:20, ]
x <- dist(mtcars2)
hchart(x)
```



Highcharter

Licensing issues

Highcharter has a dependency on Highcharts, a commercial JavaScript charting pacman::p_load. Highcharts offers both a commercial license as well as a free non-commercial license. Please review the licensing options and terms before using this software, as the highcharter license neither provides nor implies a license for Highcharts. Highcharts (http://highcharts.com) is a Highsoft product which is not free for commercial and Governmental use.

rbokeh

http://hafen.github.io/rbokeh/

rbokeh

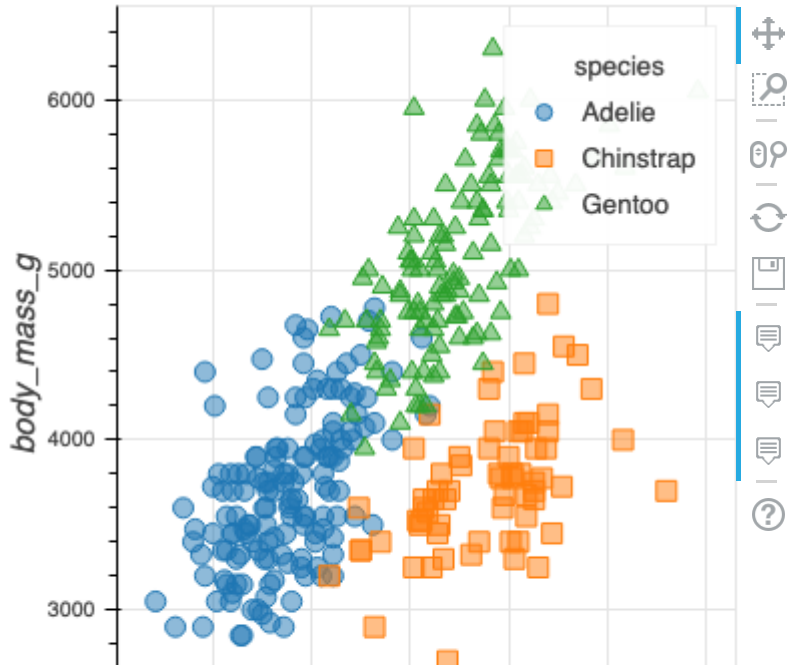
rbokeh wraps the **bokeh** plotting pacman::p_load in Python

It uses a layering concept similar to **ggplot2**

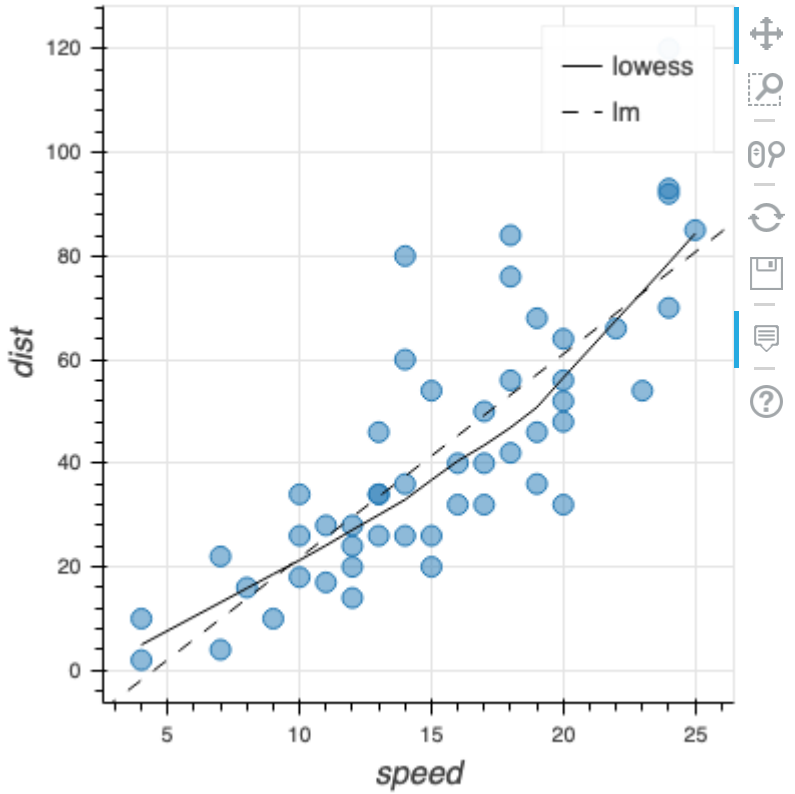
```
pacman::p_load(rbokeh)
```

```
The downloaded binary packages are in
    /var/folders/k4/xvcmx4yx76xdbl41zy3hq8rc0000gn/T/
```

```
p <- figure(width=400,height=400) %>%
    ly_points(bill_length_mm, body_mass_g,
              data=penguins,
              color=species, glyph = species,
              hover=list(bill_length_mm,body_mass_g))
p
```




```
z <- lm(dist ~ speed, data = cars)
p <- figure(width = 400, height = 400) %>%
  ly_points(cars, hover = cars) %>%
  ly_lines(lowess(cars), legend = "lowess") %>%
  ly_abline(z, type = 2, legend = "lm")
p
```



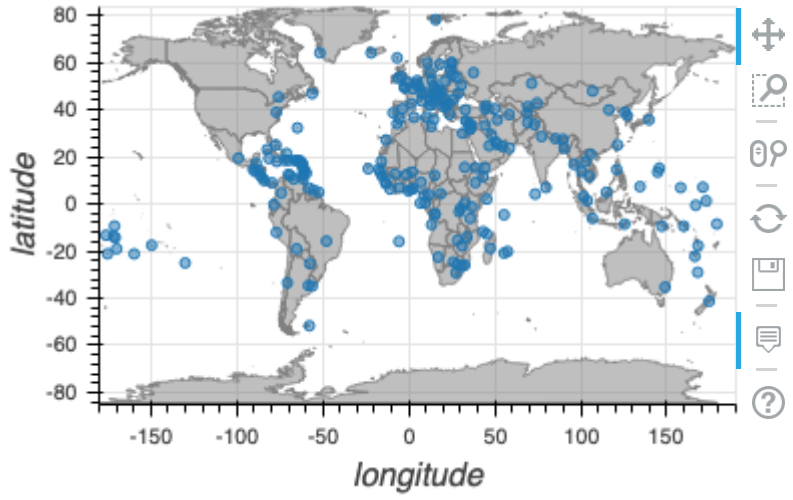
Maps

Maps

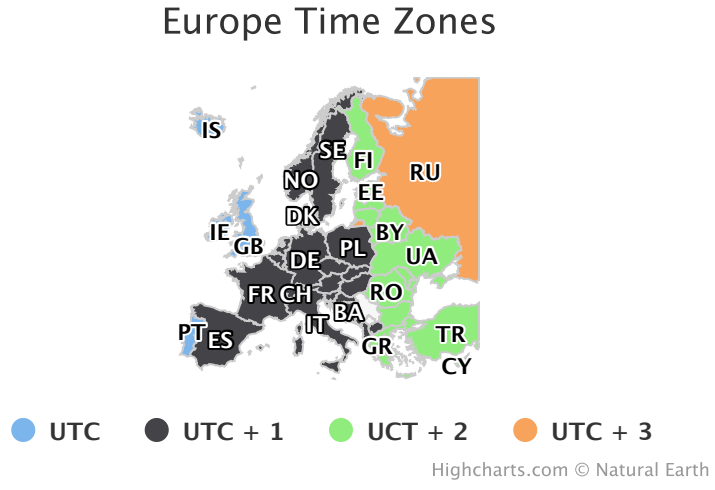
We've already seen maps with **leaflet** that allow data to be overlayed over cartographic maps

Using the packages introduced here, we also have mapping capabilities. See the respective websites for details

rbokeh



highcharter



Using D3.js in R

r2d3

D3.js is a state-of-the-art Javascript pacman::p_load for data-driven graphics

It is widely used in data journalism, including the New York Times and the Guardian

However, it controls each aspect of a chart and so can require rather long JS scripts

r2d3

If you know D3.js, you can use it pretty easily embed them in R using the **r2d3** package

```
pacman::p_load(r2d3)

r2d3(data =  read_csv('../data/flare.csv'), d3_version = 4, script = 'bubbles.js')
```

```
Error: forcegraph is not defined
ReferenceError: forcegraph is not defined
```

```
r2d3(data = jsonlite::read_json('../data/miserables.json'), d3_version = 4, script = 'forcegraph.js')
```

networkD3

You can also draw networks slightly more easily using the **networkD3** package.

```
pacman::p_load(networkD3)
lemis <- jsonlite::fromJSON('../data/miserables.json'
lemis$links <- lemis$links %>%
    mutate(ID = 1:n()) %>%
    gather(location, nm, source, target) %>%
    mutate(nm = as.integer(as.factor(nm))-1) %>%
    spread(location, nm)

forceNetwork(Links = lemis$links, Nodes = lemis$nodes
             Source='source', Target='target',
             Value = 'value',
             NodeID = 'id', Group = 'group', opacity=
             zoom = TRUE, fontSize = 20)
```

Crosstalk

Crosstalk is a package that allows multiple HTML widgets to share data and interact together.

Not all packages in the **htmlwidgets** family are **crosstalk**-compatible.

```
load('data/exdata.rda')
pacman::p_load(leaflet); pacman::p_load(crosstalk); pacman::p_load(d3scatter)

gpx1 <- gpx %>%
   mutate(Minutes = as.numeric(difftime(Time, min(Time), units = 'mins'))) %>%
   filter(Minutes <= 50) %>%
   left_join(run_data)

 shared_gpx1 <- SharedData$new(gpx1)
 bscols(
   leaflet(data = shared_gpx1, width="100%", height=450) %>% addTiles() %>%
     addCircleMarkers(~Longitude, ~Latitude, radius=1, color='blue'),
   list(
   d3scatter(shared_gpx1, ~Minutes, ~ HR, width="100%", height=450)
   )
 )
 bscols(
   filter_slider("Minutes","Time", shared_gpx1, ~Minutes, width="100%")
 )
```

`leaflet` + `d3scatter` + `crosstalk`

Linked graphs

Activity Monitoring with Sensors and R

Abhijit Dasgupta

1 / 24

Conclusions

- There are many many resources today to create interactive graphics
- For many of them, there are wrappers in R
- You can explore the respective packages for many more details about the different kinds of charts and customizations that are possible.