Data visualization

Abhijit Dasgupta

March 25-27, 2019

Starting with ggplot2

Why ggplot2?

We're making the decision to use `ggplot2` for graphics

- Makes pretty good formatting choices out of the box
- Works like pipes!!
- Is declarative (tell it what you want) without getting caught up in minutae
- Strongly leverages data frames (good practice)
- Fast enough
- There are good templates if you want to change the look

Introduction to ggplot2

The `ggplot2` package is a very flexible and (to me) intuitive way of visualizing data. It is based on the concept of layering elements on a canvas.

> This idea of layering graphics on a canvas is, to me, a nice way of building graphs

Introduction to ggplot2

You need:

- A `data.frame` object
- *Aesthetic mappings* (aes) to say what data is used for what purpose in the viz
  - x- and y-direction
  - shapes, colors, lines
- A *geometry object* (geom) to say what to draw
  - You can "layer" geoms on each other to build plots

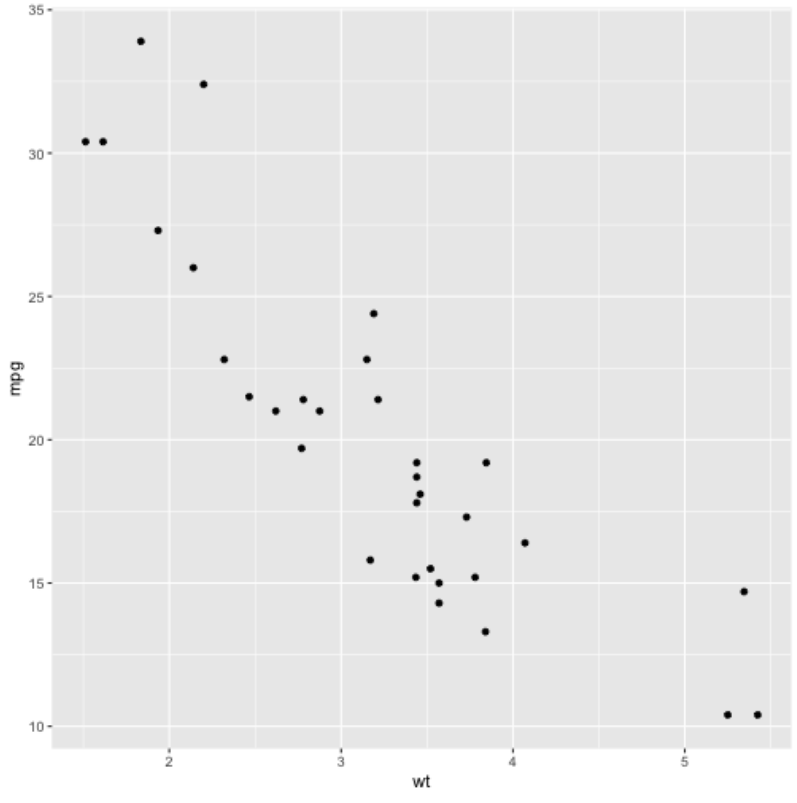
Introduction to ggplot2

`ggplot` used pipes before pipes were a thing.

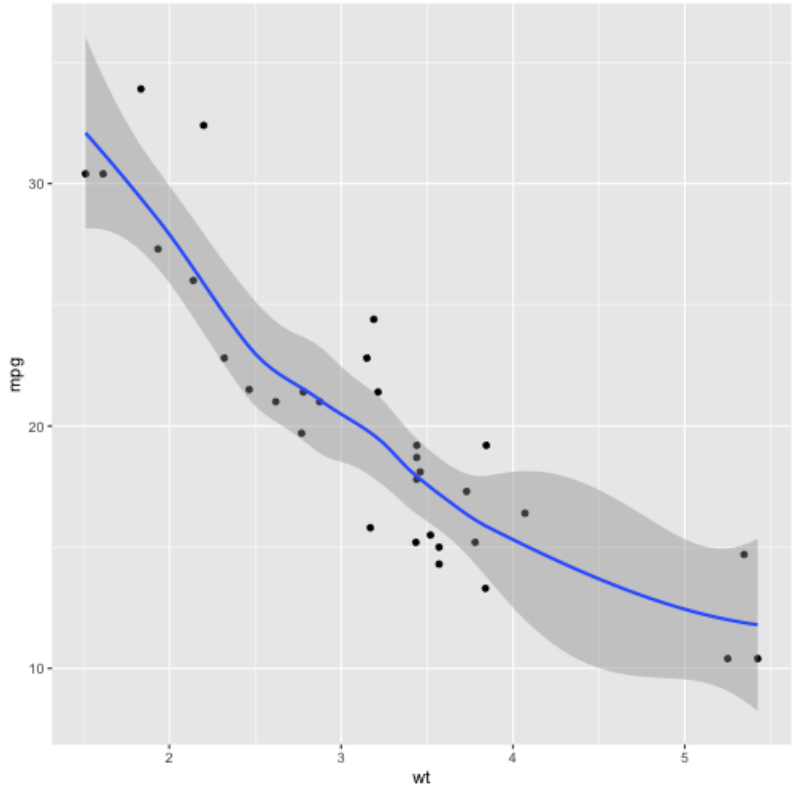
However, it uses the + symbol for piping rather than the %>% operator, since it pre-dates the `tidyverse`

```
library(ggplot2)
theme_set(theme_bw())
ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point()
```

- A `data.frame` object: mtcars
- Aesthetic mapping:
  - x-axis: wt
  - y-axis: mpg
- Geometry:
  - geom_point: draw points

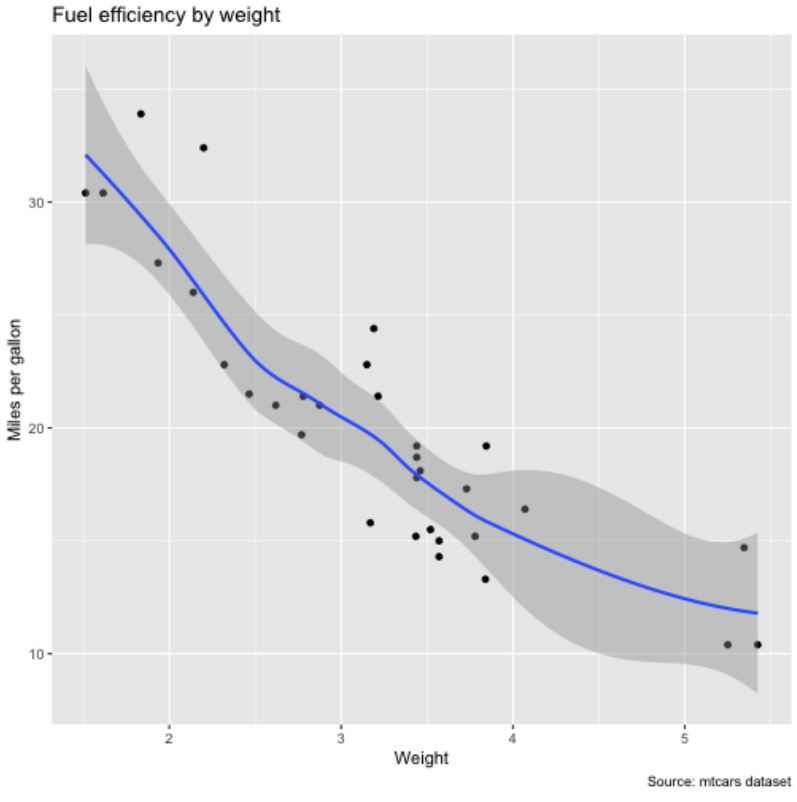


```
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()+
  geom_smooth()
```



- A `data.frame` object: mtcars
- Aesthetic mapping:
  - x-axis: wt
  - y-axis: mpg
- Geometry:
  - geom_point: draw points
  - geom_smooth: Add a layer which draws a best-fitting line

```
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()+
  geom_smooth()+
  labs(x = 'Weight',
       y = 'Miles per gallon',
       title = 'Fuel efficiency by weight',
       caption = "Source: mtcars dataset")
```

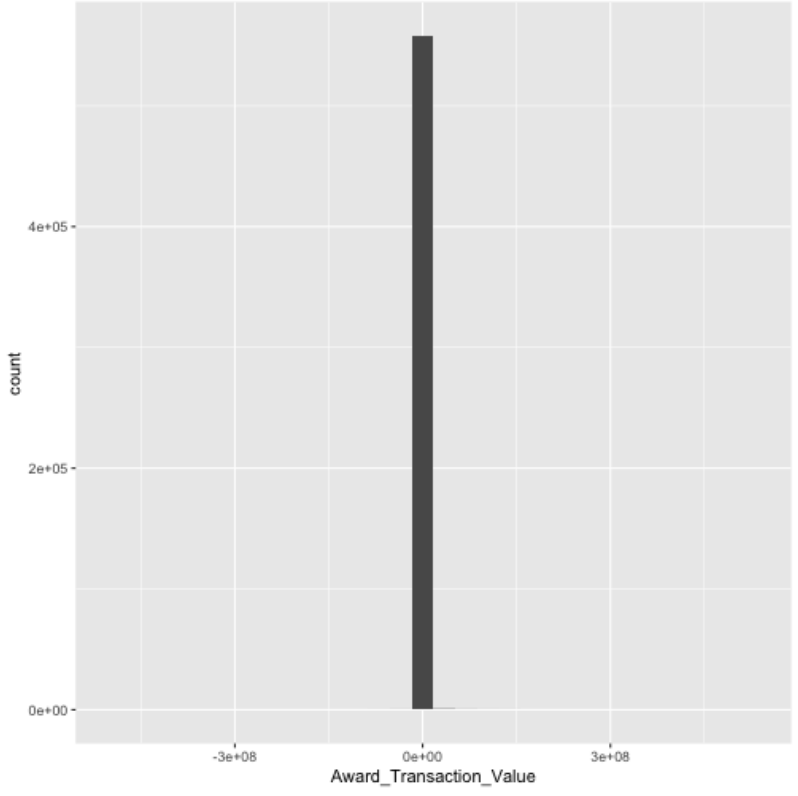


- A `data.frame` object: mtcars
- Aesthetic mapping:
  - x-axis: wt
  - y-axis: mpg
- Geometry:
  - geom_point: draw points
  - geom_smooth: Add a layer which draws a best-fitting line

Basic plots

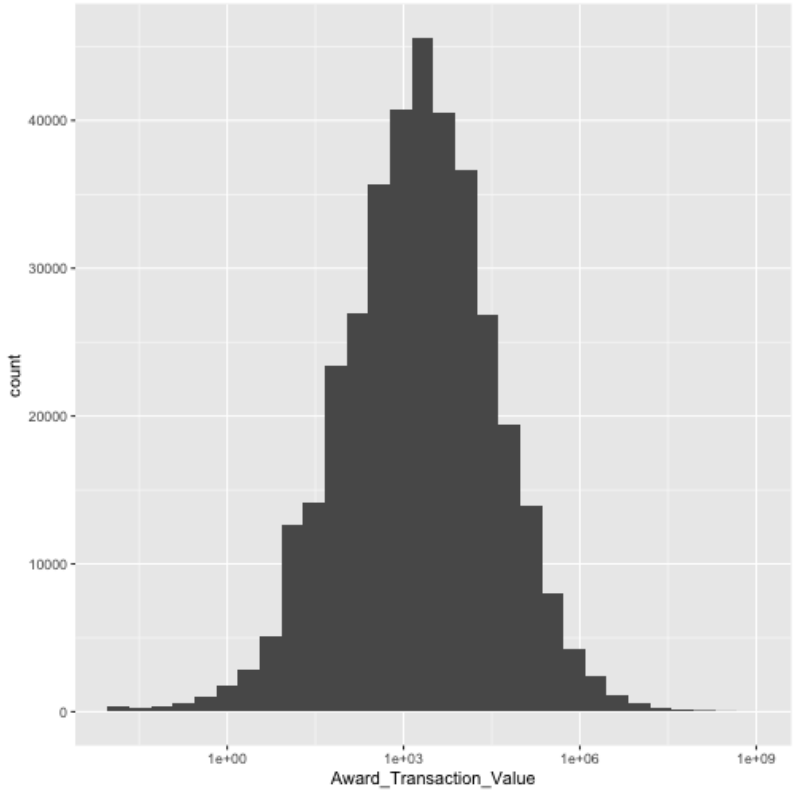
Histograms

```
dos <- import('Data/FSI/Department of State.csv')
dos %>%
  ggplot(aes(x = Award_Transaction_Value)) + geom_histogram()
```



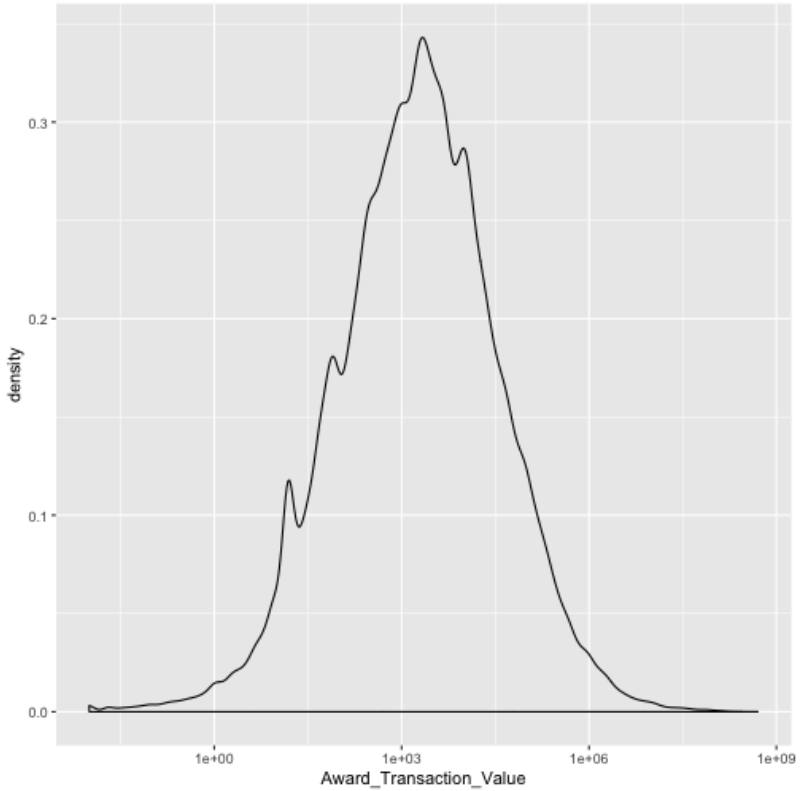
Histograms

```
dos %>%
  ggplot(aes(x = Award_Transaction_Value)) + geom_histogram()+
  scale_x_log10() # x-axis on log scale
```



Density plots

```
dos %>%
  ggplot(aes(x = Award_Transaction_Value)) + geom_density()+
  scale_x_log10()
```



Bar plots

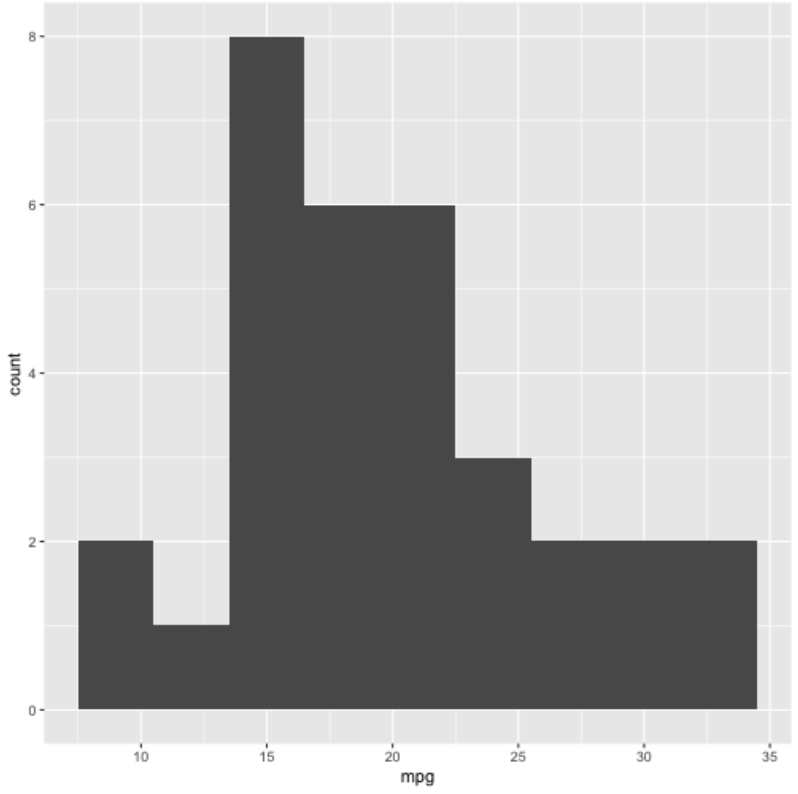
```
library(lubridate)
dos %>%
  group_by(year = year(as_date(Award_Start_Date))) %>%
  summarize(amount = sum(Award_Transaction_Value)) %>%
  ggplot(aes(x = year, y = amount)) + # Note change in pipe operator
    geom_bar(stat='identity')
```

Exercise

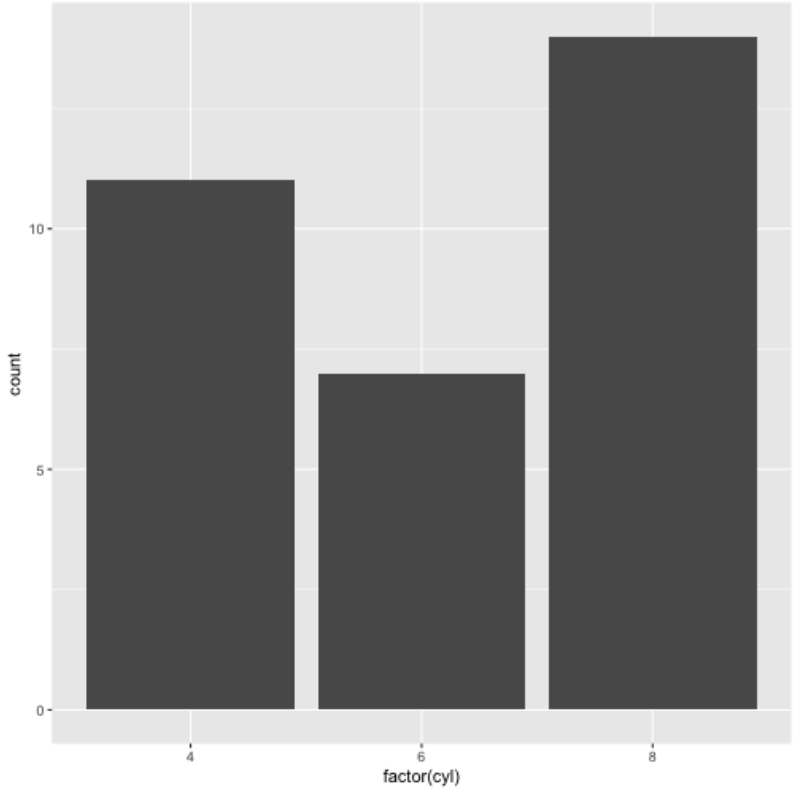
Using the `mtcars` dataset in R, create:

1. A histogram of the fuel efficiences (`mpg`) in the data set
2. A bar plot of frequencies of number of cylinders (`cyl`) in the car

```
ggplot(mtcars, aes(x = mpg)) + geom_histogram(binwidt
```




```
ggplot(mtcars, aes(x = factor(cyl))) + geom_bar()
```

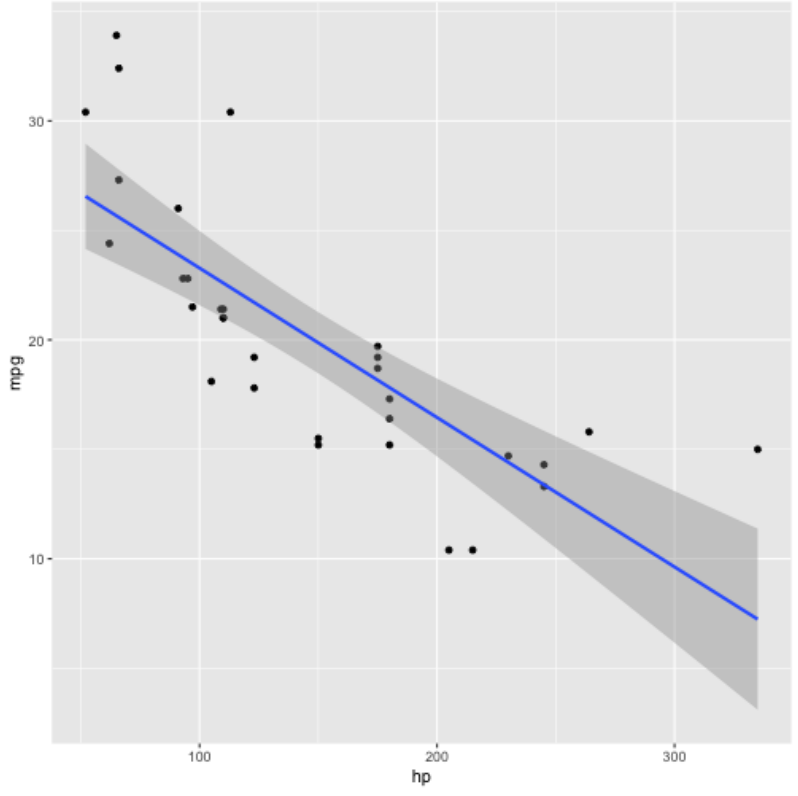



```
# ggplot(mtcars) + geom_histogram(aes(x = mpg), binwi
```

Two continuous variables

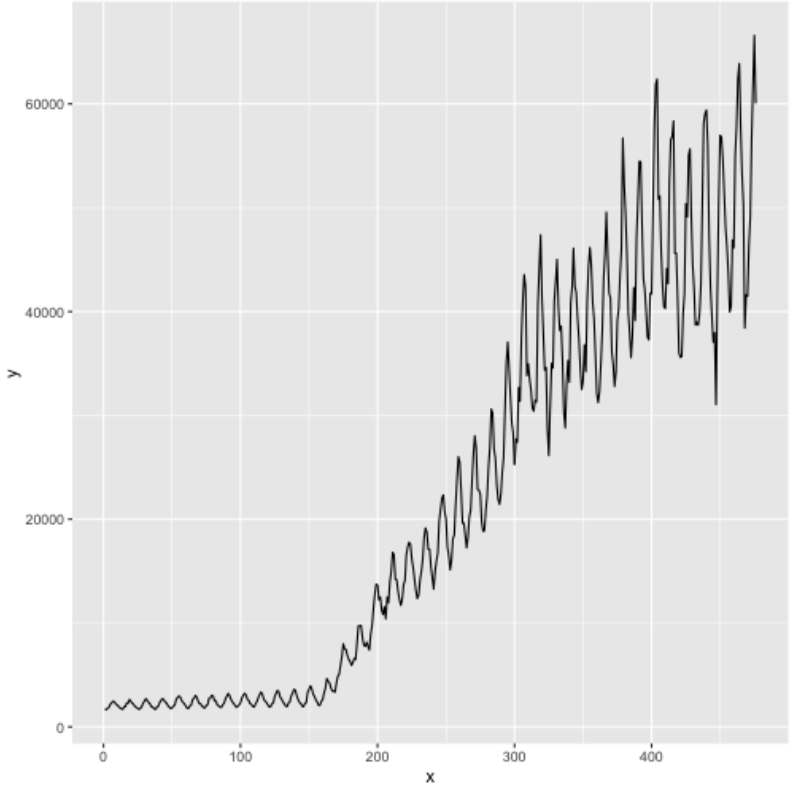
Scatter plot with best-fitting line

```
ggplot(mtcars, aes(x = hp, y = mpg))+
  geom_point()+
  geom_smooth(method = 'lm')
```



Line plot (for time series)

```
library(forecast)
d <- data.frame(x = 1:length(gas), y = gas) # Australian monthly gas production
ggplot(d, aes(x, y)) + geom_line()
```

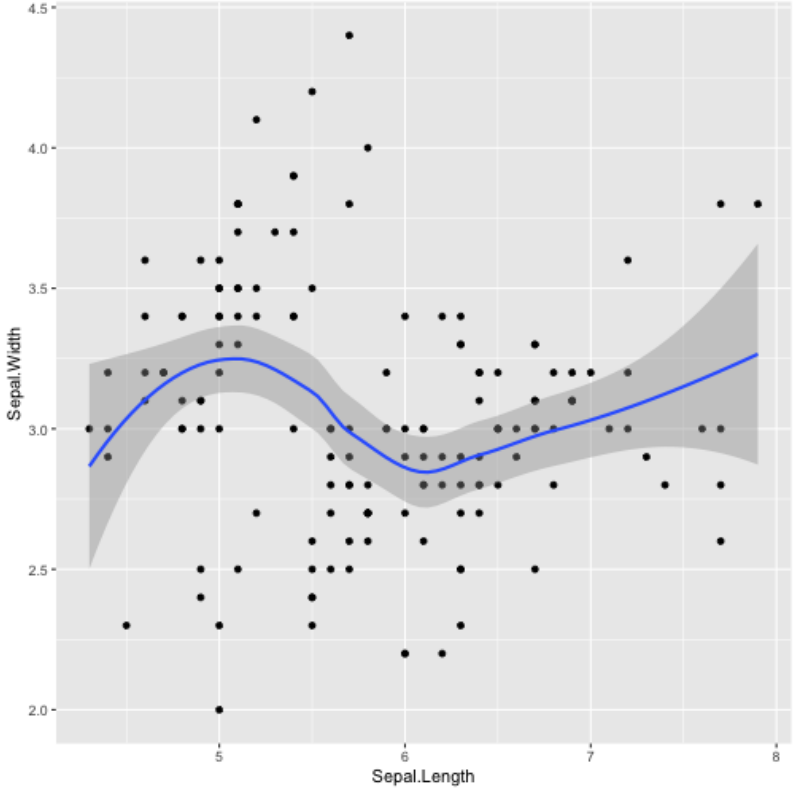


Exercise

1. Create a scatter plot of sepal length and sepal width from the `iris` dataset, and add a smooth line through it

Solution

```
ggplot(iris, aes(Sepal.Length, Sepal.Width)) + geom_point() + geom_smooth()
```



Continuous variable with discrete variable

Box plots

```
dos %>%
ggplot(aes(x = factor(year(as_date(Award_Start_Date))),
                y = Award_Transaction_Value))+
  geom_boxplot() +
  scale_y_log10()+
  labs(x = 'Year')
```

Violin plots

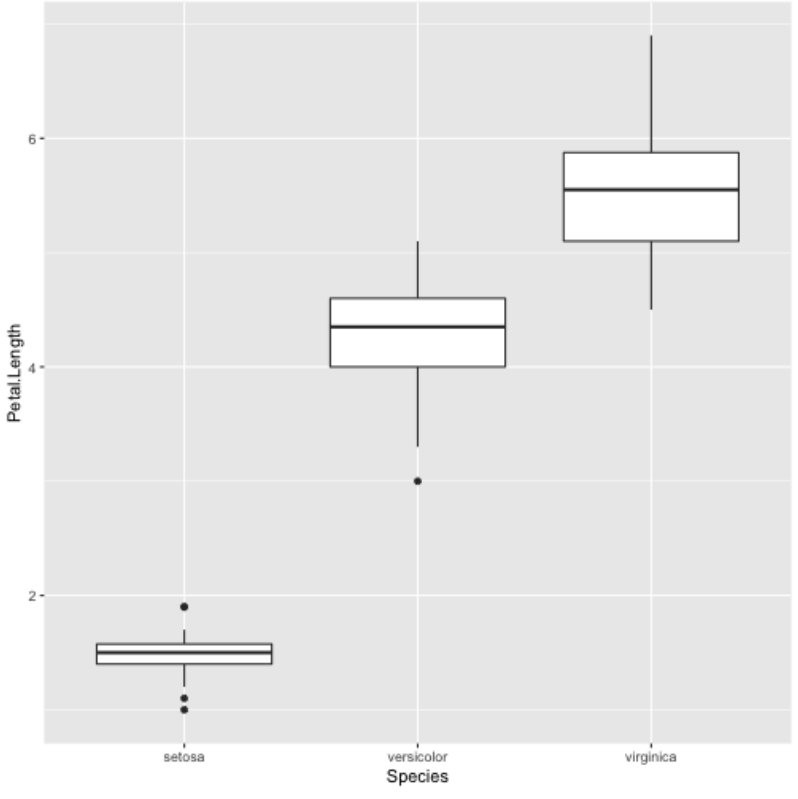
```
dos %>%
ggplot(aes(x = factor(year(as_date(Award_Start_Date))),
                y = Award_Transaction_Value))+
  geom_violin() +
  scale_y_log10()+
  labs(x = 'Year')
```

Exercise

1. Plot a boxplot of petal length by species using the `iris` dataset

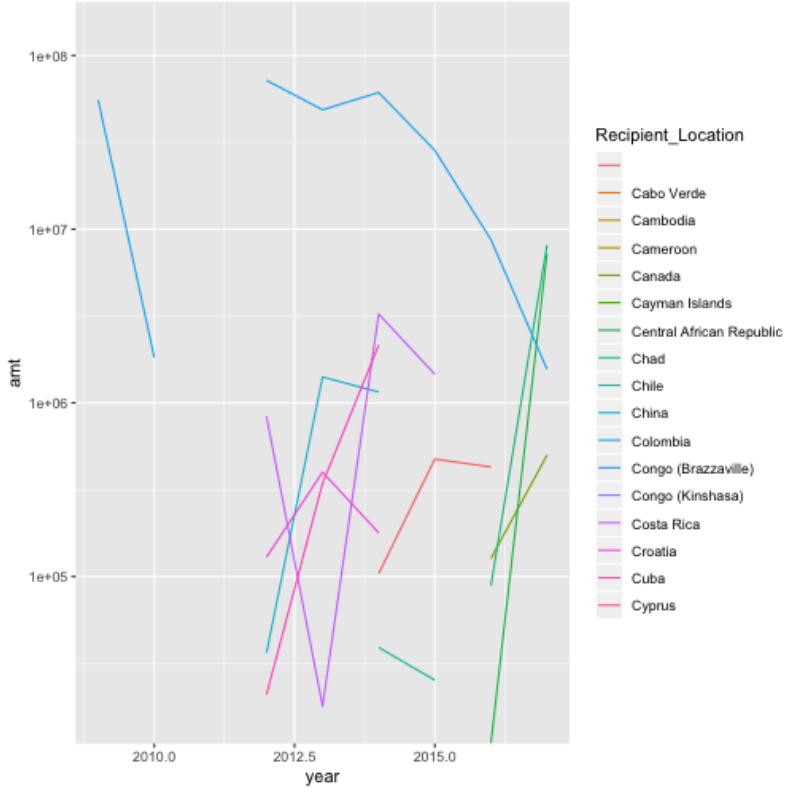
Solution

```
ggplot(iris, aes(x = Species, y = Petal.Length))+geom_boxplot()
```

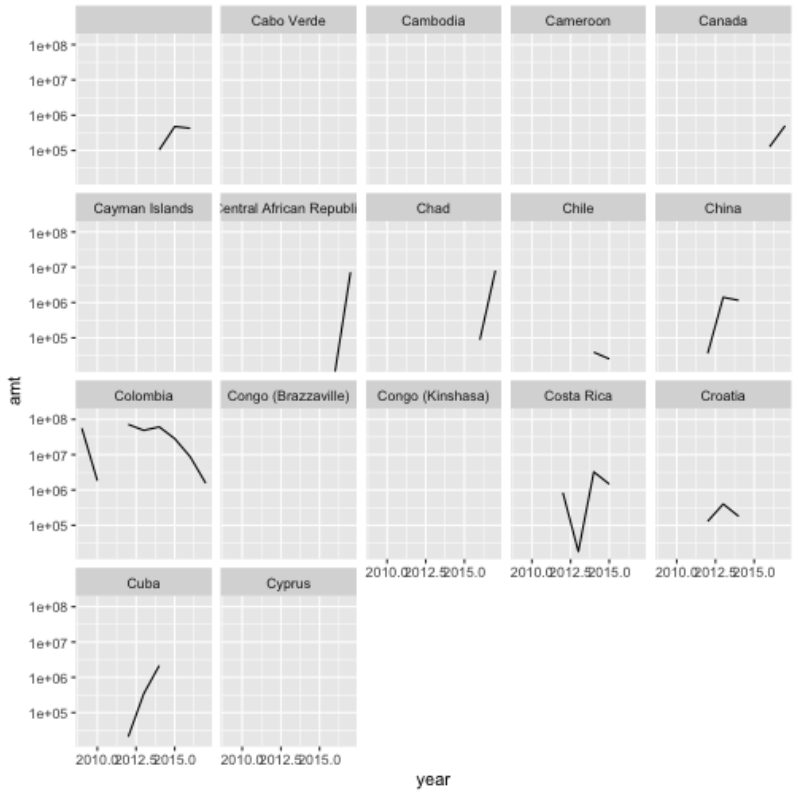


Grouped visualization

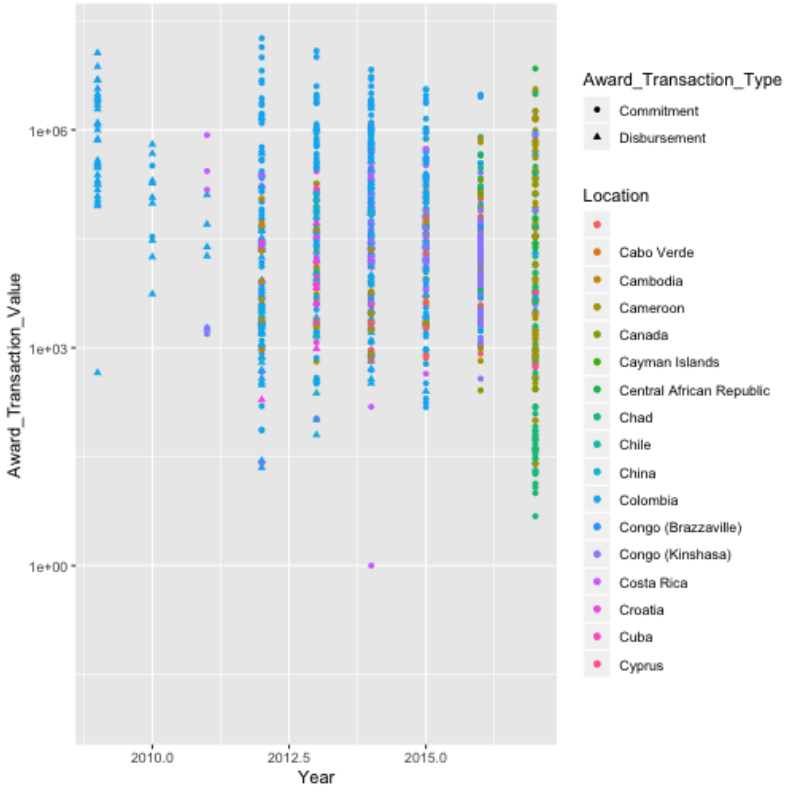
```
grp_data <- dos %>%
  group_by(Recipient_Location, year = year(as_date(Award_Start_Date))) %>%
  summarize(amt = sum(Award_Transaction_Value)) %>%
  filter(str_detect(Recipient_Location, '^C'))
ggplot(grp_data, aes(x = year, y = amt, color=Recipient_Location))+
  geom_line()+
  scale_y_log10()
```



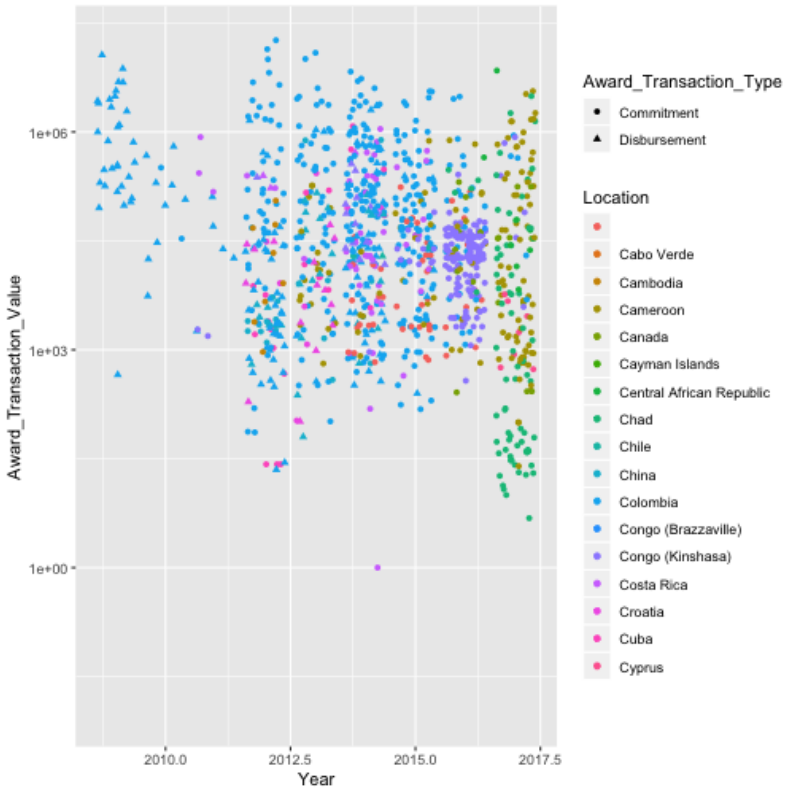
```
ggplot(grp_data, aes(x = year,  y = amt))+
  geom_line()+
  scale_y_log10()+
  facet_wrap(~Recipient_Location)
```



```
dos %>% filter(str_detect(Recipient_Location, '^C'))
ggplot(aes(x = year(as_date(Award_Start_Date)),
           y = Award_Transaction_Value,
           color = Recipient_Location,
           shape = Award_Transaction_Type))+
  geom_point()+
  labs(x = 'Year', color='Location')+
  scale_y_log10()
```



```
dos %>% filter(str_detect(Recipient_Location, '^C'))
ggplot(aes(x = year(as_date(Award_Start_Date)),
           y = Award_Transaction_Value,
           color = Recipient_Location,
           shape = Award_Transaction_Type))+
  geom_jitter()+
  labs(x = 'Year', color='Location')+
  scale_y_log10()
```



Error bars

```
schools <- rio::import('Data/FSI/schools.rds')
schools %>% filter(tophead=='Elementary schools',
                   head2=="Average hours in school day") %>%
  filter(!is.na(State), State != 'United States') %>%
  ggplot(aes(x = State, y = stats, ymin = stats - 2*se,
             ymax = stats + 2*se)) +
  geom_pointrange()+
  labs(y = 'Avg hours in school day')+
  theme_bw()+
  theme(axis.text.x = element_text(angle=45, hjust = 1))
```

Maps

Chloropleth

Using shapefiles

```
library(sf)
hrr_info <- st_read('~/Downloads/hrr_bdry-1/HRR_Bdry.SHP')
```

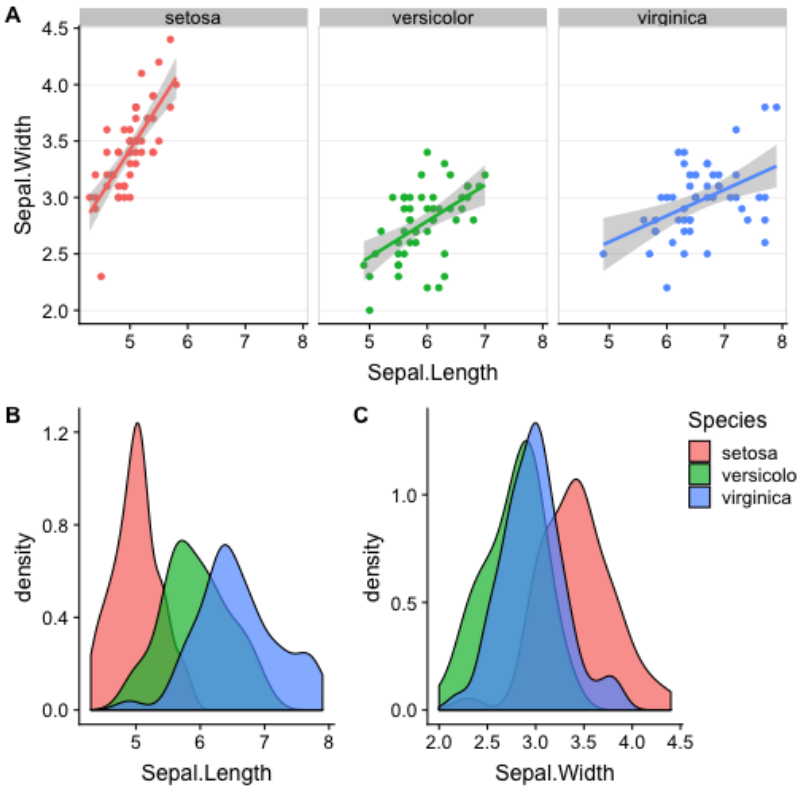
```
Reading layer `HRR_Bdry' from data source `/Users/abhijit/Downloads/hrr_bdry-1/HRR_Bdry.SHP' using driver `ESRI Sh
Simple feature collection with 306 features and 3 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: -124.7319 ymin: 24.54394 xmax: -66.95047 ymax: 49.38308
epsg (SRID):    NA
proj4string:    NA
```

```
head(hrr_info)
```

```
Simple feature collection with 6 features and 3 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: -88.885 ymin: 30.22065 xmax: -84.63694 ymax: 35.42359
epsg (SRID):    NA
proj4string:    NA
  HRR_BDRY_I HRRNUM        HRRCITY                       geometry
1          1      1 AL- BIRMINGHAM MULTIPOLYGON (((-85.89658 3...
2          2      2     AL- DOTHAN MULTIPOLYGON (((-86.194 31....
3          3      5 AL- HUNTSVILLE MULTIPOLYGON (((-86.69474 3...
4          4      6     AL- MOBILE MULTIPOLYGON (((-87.55554 3...
5          5      7 AL- MONTGOMERY MULTIPOLYGON (((-86.49925 3...
6          6      9 AL- TUSCALOOSA MULTIPOLYGON (((-87.31229 3...
```

Putting together multiple graphs

```
# install.packages('cowplot')
library(cowplot)
p1 <- ggplot(iris, aes(Sepal.Length, Sepal.Width, col
  geom_point() + facet_grid(. ~ Species) + stat_smoot
  background_grid(major = 'y', minor = "none") +
  panel_border() + theme(legend.position = "none")

# plot B
p2 <- ggplot(iris, aes(Sepal.Length, fill = Species))
  geom_density(alpha = .7) + theme(legend.justificati
p2a <- p2 + theme(legend.position = "none")

# plot C
p3 <- ggplot(iris, aes(Sepal.Width, fill = Species))
  geom_density(alpha = .7) + theme(legend.position =

# legend
legend <- get_legend(p2)

# align all plots vertically
plots <- align_plots(p1, p2a, p3, align = 'v', axis =

# put together bottom row and then everything
bottom_row <- plot_grid(plots[[2]], plots[[3]], legen
plot_grid(plots[[1]], bottom_row, labels = c("A"), nc
```



Interactive graphs

Interactive graphs

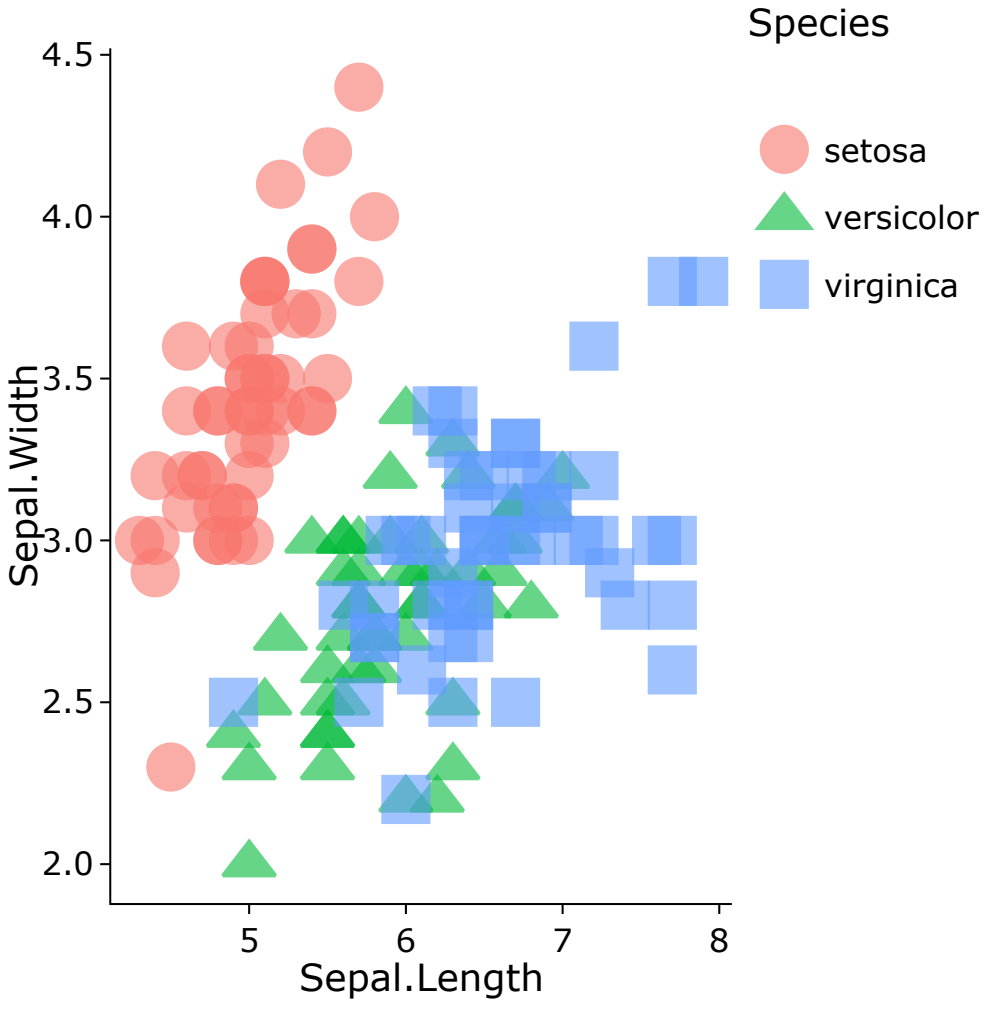
There are several packages available in R to do interactive graphs.

- plotly: A general plotting library from the folks at Plot.ly
- highchart: A port of the highcharter Javascript library
- dygraphs: A port of the dygraph Javascript library
- leaflet: Interactive maps

There are others as well. See http://www.htmlwidgets.org for more details

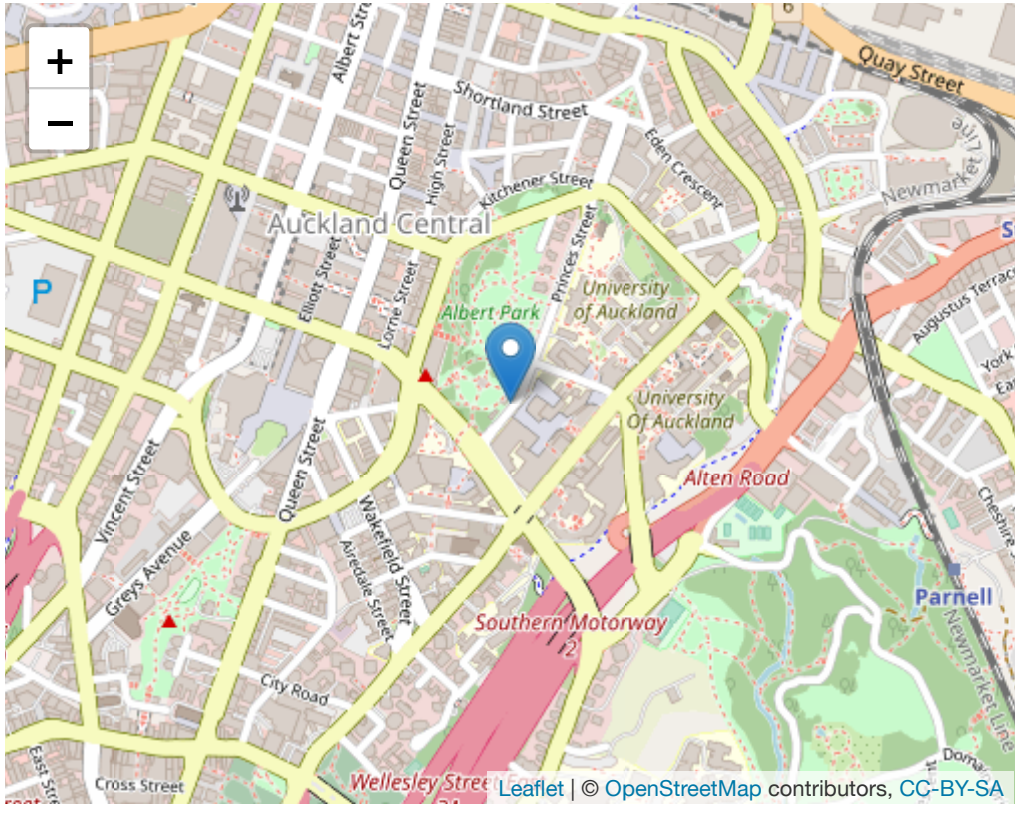
Plotly

```
library(ggplot2)
library(plotly)
p=ggplot(iris, aes(x=Sepal.Length,
                   y=Sepal.Width,
                   color=Species,
                   shape=Species)) +
    geom_point(size=6, alpha=0.6)
mytext=paste("Sepal Length = ", iris$Sepal.Length,
             "\n" , "Sepal Width = ", iris$Sepal.Widt
             "\n", "Row Number: ",rownames(iris),  se
pp=plotly::plotly_build(p)
style( pp, text=mytext,
       hoverinfo = "text",
       traces = c(1, 2, 3) )
```



Leaflet

```
library(leaflet)

m <- leaflet() %>%
  addTiles() %>%  # Add default OpenStreetMap map til
  addMarkers(lng=174.768, lat=-36.852, popup="The bir
m
```



Dygraph

```
library(dygraphs)
library(xts)          # To make the convertion data-f

# Create data + verify it is date format + change the
data=data.frame(time=seq(from=Sys.Date()-40,
                         to=Sys.Date(), by=1 ),
                value=runif(41))
str(data$time)
data=xts(x = data$value, order.by = data$time)

# Add points
dygraph(data) %>%
  dyOptions( stemPlot = TRUE)
```

```
Date[1:41], format: "2019-02-15" "2019-02-16" "2019-
```