Grammar of Graphics

Intro to the grammar, ggplot2 + interactive graphics

SURF: Justin Castagna

9/8/2017

What is Grammar?

Most people think of grammar in terms of language.

A Definition:

The structural rules governing the syntax and composition of clauses, phrases and words. Words can be broadly classified as Adjective (descriptive), Noun (objects) or , Verb (actions).

Punctuation marks are used to improve comprehension of the subject.

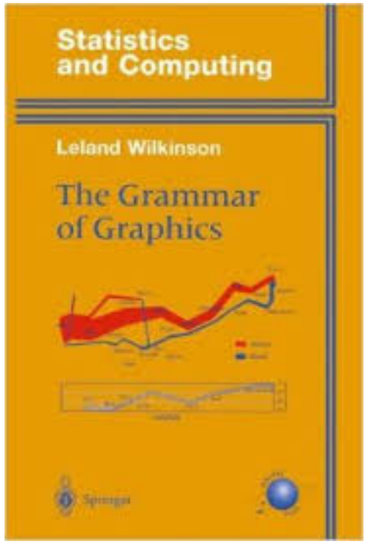
"Let's eat Grandma" vs. "Let's eat , Grandma"

Grammar

Why is a grammar of graphics important?

To understand, discuss and think deeply about graphs and their components we need a common language and thus a graphic grammar to facilitate this.

"The Grammar of Graphics" by Leyland Wilkinson, Annad & Grossman (2005).



What is a Graphic?

A graph is an abstract idea.

It becomes a graphic through the process of rendering.

A graph is made up of an aggregate of layered collections of components. Each layer has its own aesthetic properties that can be used to further describe the layer.

Leyland proposed that there were three stages moving from data to a final graphic.

1. Specification: The explicit syntax used to explicitly define the graph.
2. Assembly: The coordination of the specified components to portray the abstract graph.
3. Display: The rendering of the graph and its aesthetic attributes to a final display system graphic.

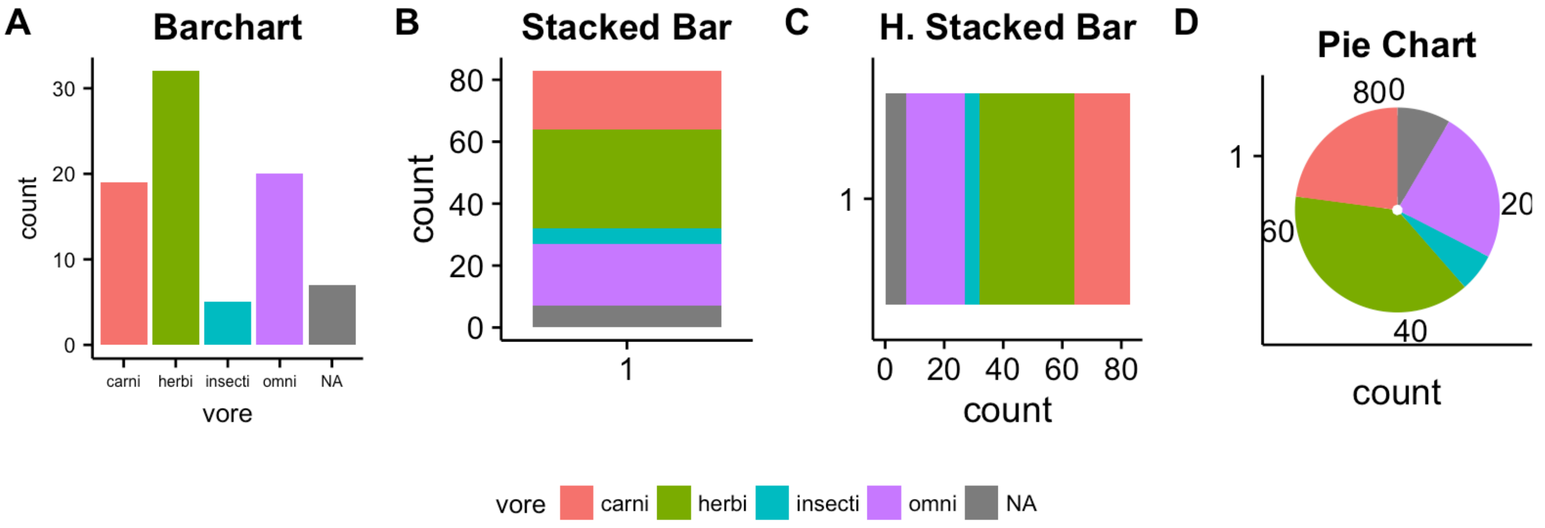
The key stage was "Specification" as computers automated Assembly & Display.

Grammar Specifications

A change in a graph specification results in a different graphic being rendered.

ID	**Specification**	Description	Example
1.00	DATA	A set of data operations that create variables from datasets	
2.00	TRANS	Variable transformations	Rank, bin
3.00	SCALE	Scale transformations	Log
4.00	COORD	A coordinate system	Cartesian, polar
5.00	ELEMENT	Graphs	Points, line
		Aesthetic attributes	Colour, transparency, label
6.00	GUIDE	One or more guides	Axes, legends

Specifications Example



1. Similarities: All use 'Mammals Sleep' dataset and variable vore used (DATA).
   A count of the variable 'vore' (TRANS) and same colours aesthetic (ELEMENT) used with legend (GUIDE).

2. Differences: (A,B,C) use a Cartesian coordinate system. (D) uses a polar coordinate system (COORD). (C) uses a flipped axes to (A,B). Each graphic's title description (GUIDE).

Transitioning from theory to practice

R and it's predecessor language, S, originated in the late 1970's. The existing R graphic systems (such as Base, Grid & Lattice) were built on the ideas of their day.

Enter ggplot2.

GGPlot2 package developed by Hadley Wickham.

Influenced by philosophy of Leland's 'Grammar of Graphics'.

ggplot's specifications syntax, particularly for ELEMENTS, is far richer.

Adopted modern ideas such as object oriented programming, pipes, and a new programming paradigm that had not been widely used before in the R eco system.

Consistent syntax and naming conventions.

Programming

Programming, or writing code in R is how we instruct the computer to complete the assembly and display stage of a graphic's construction.

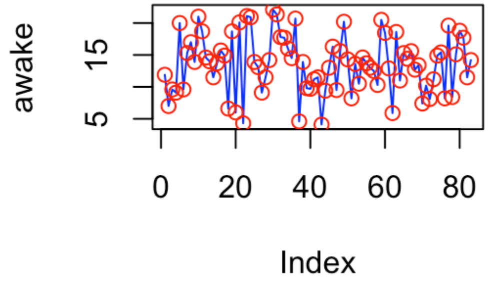
Two Programming paradigms:

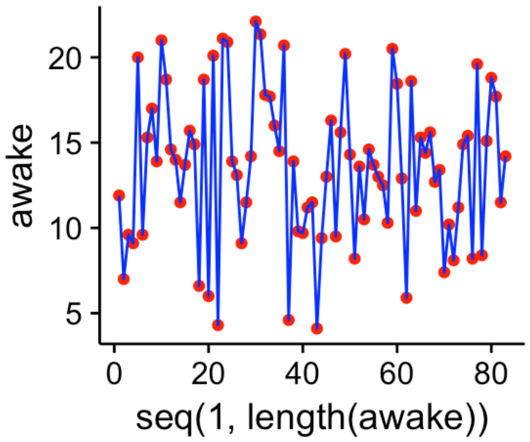
Imperative - Code step by step to create a graphic. Complicated graphics require programming techniques such as for loops.

Declarative - You describe and code what your graph actually is.

Imperative vs Declarative

```
# Imperative;
plot(awake)

# Plot with lines and points      # Alternate method
plot(awake,type="p",col="red")      plot(awake,type="l",col="blue")
lines(awake, col="blue")            points(awake,col="red")

# Declarative
ggplot(data=msleep,aes(x=seq(1,length(awake)),y=awake)) +
        geom_point(colour="red") +
        geom_line(colour="blue")
```





ggplot2 specifications

ggplot2's far richer specifications.

1. **Geoms:** Short for geometric objects, describe the type of plot produced eg. Points, maps, lines, polygons.
2. **Statistics:** Transform your data before plotting. eg. Unique, summary, quantile.
3. **Scales:** Control the mapping between data and aesthetics.eg. Guides, colour, linetype, shape, size.
4. **Coordinate:** Adjust the mapping from coordinates system to the 2d plane of the computer screen. eg. Cartesian, fixed, flip, polar
5. **Faceting:** Display subsets of the dataset in different panels. eg. Grid, wrap, null, labeller
6. **Position:** Used to adjust positioning of objects to achieve effects. eg. Jitter, dodge, fill, stacking.
7. **Annotation:** Specialised functions for adding annotations to a plot. eg. Annotate, map, raster, logticks.
8. **Themes:** Control non-data components of the plot eg. Classic, Tufte

ggplot2 Layers

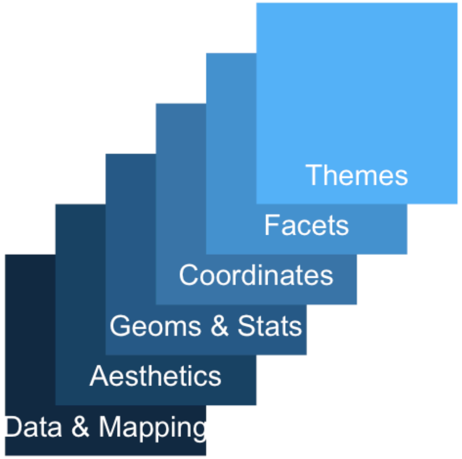
A plot in ggplot can be described a

Plot = (Data + Mapping) + Layer where a
Layer = Geom + Geom(Parameters) + Stat + stat:parameters + Position

Each graphic contains at a minimum Data, Geometric object (Geom), Statistical transformation (Stat), Scales and a Co-ordinate System.

A graphic can have multiple layers.

In practice each geom has a default statistic associated with it and vice versa.



Using ggplot2

ggplot2 has sensible defaults built in.

ggplot2 code can written / used in 3 main ways.

1. As a traditional function
2. As an object
3. Using pipes (%>%)

Layers are added by use '+' to link.

Consistent syntax for usage "specification_description" eg geom_point()

Aesthetics

The key to ggplot2's success.

Aesthetics often shortened to 'aes' refers to both mapping and the visual aspect of a graph.

Aesthetic describes how variables are mapped or assembled into the graphs coordinates system.

Aesthetics also include colour, scale, size, fill and shape and transparency (called alpha).

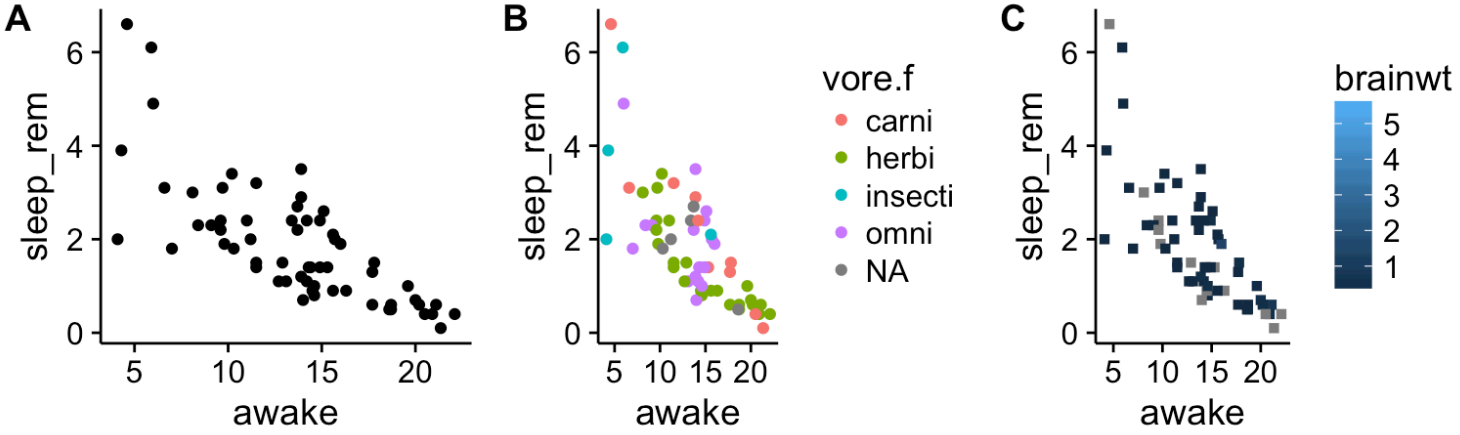
Aesthetics allow additional information to encoded into plots.

Allows user to recognise differences between discrete & continuous data.

Mapping & aesthetics can be 'inherited' or passed through to each layer automatically if described in initial mapping aesthetic.

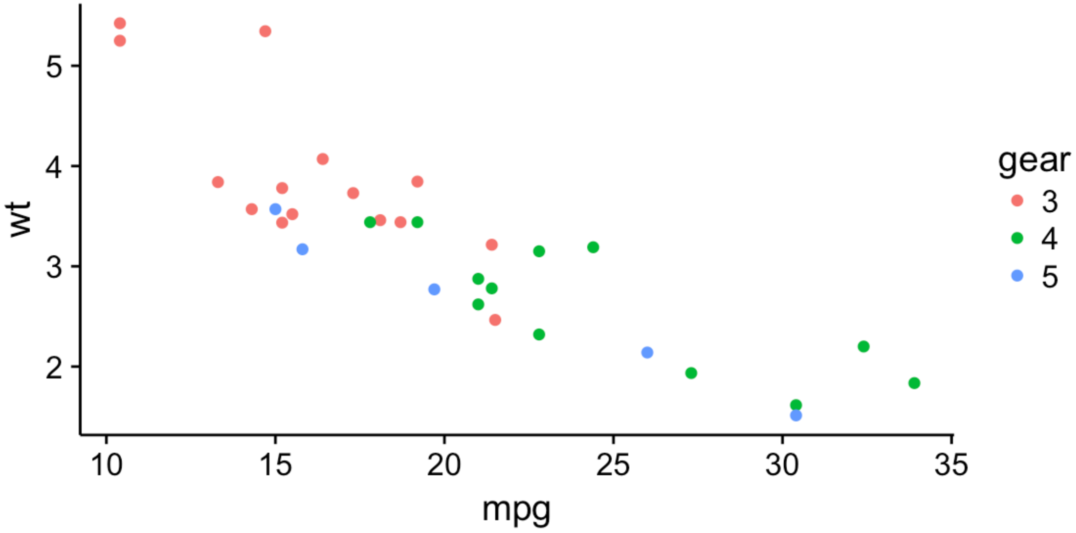
ggplot2 code building blocks

```
# A function
ggplot(mtcars,aes(x=awake, y=sleep_rem)) +
         geom_point()

# B object; Note inheritance of colour
g = ggplot(msleep,aes(x=awake, y=sleep_rem,colour=vore.f))
g +  geom_point()

# C pipes
msleep %>% ggplot(.,aes(x=awake, y=sleep_rem)) + geom_point(shape=15,aes(colour=brainwt))
```



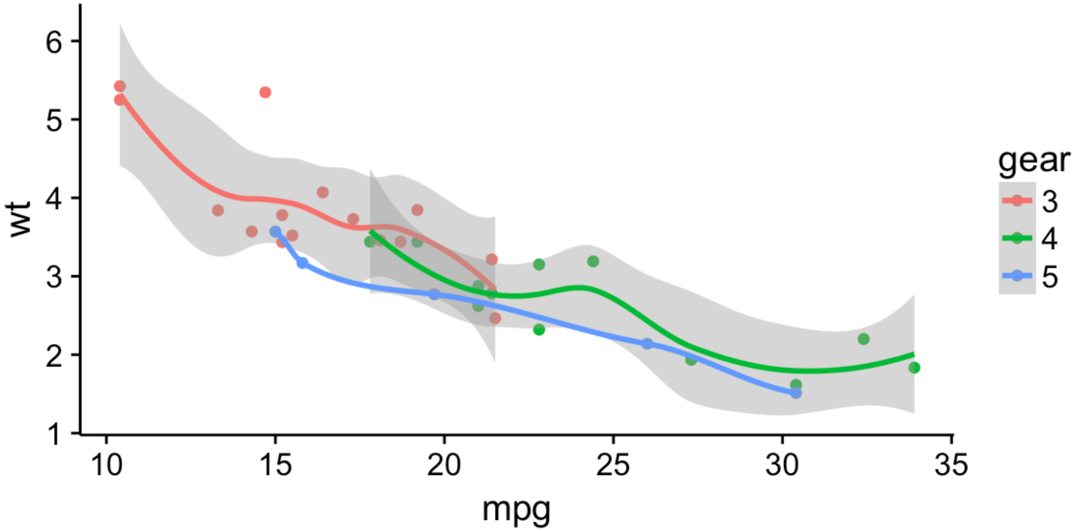
geom_point(1)

```
g = ggplot(mtcars,aes(y=wt,x=mpg,colour=gear))
g +  geom_point()
```



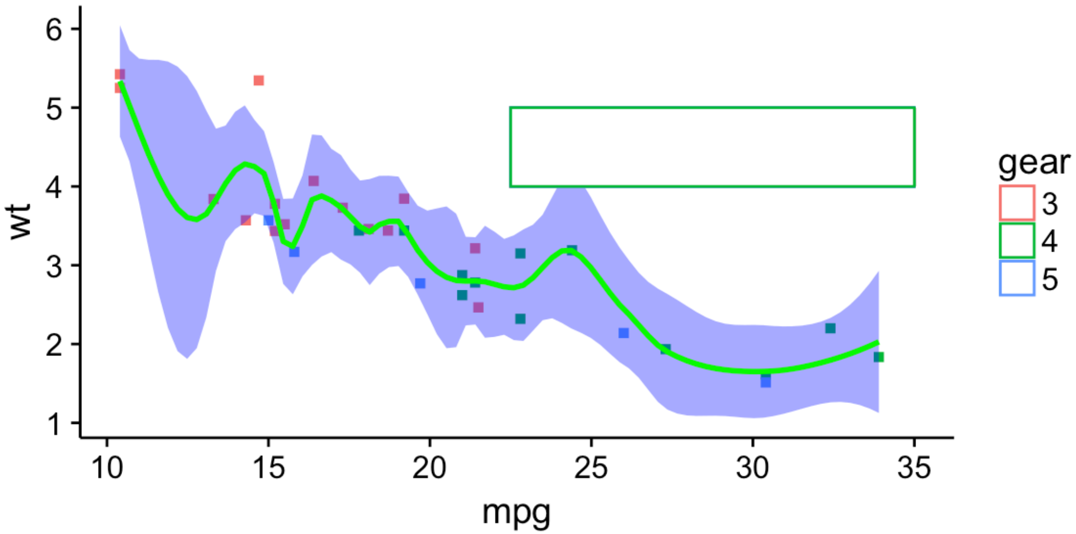
geom_point() + geom_smooth()

```
g + geom_point() +  # Recall that  g = ggplot(mtcars,aes(y=wt,x=mpg,colour=gear))
  geom_smooth()     # note
```



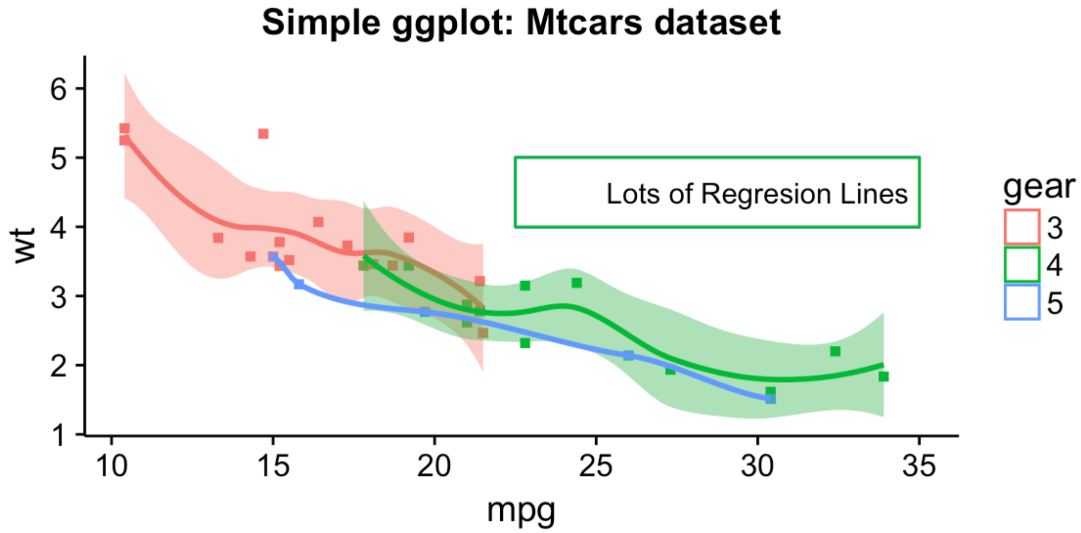
Three geoms layered

```
g + geom_point(shape=15) +
  geom_smooth(colour="green",fill="blue",span=.25) +
  geom_rect(aes(xmin=22.5,xmax=35,ymin=4,ymax=5),fill="white")
```



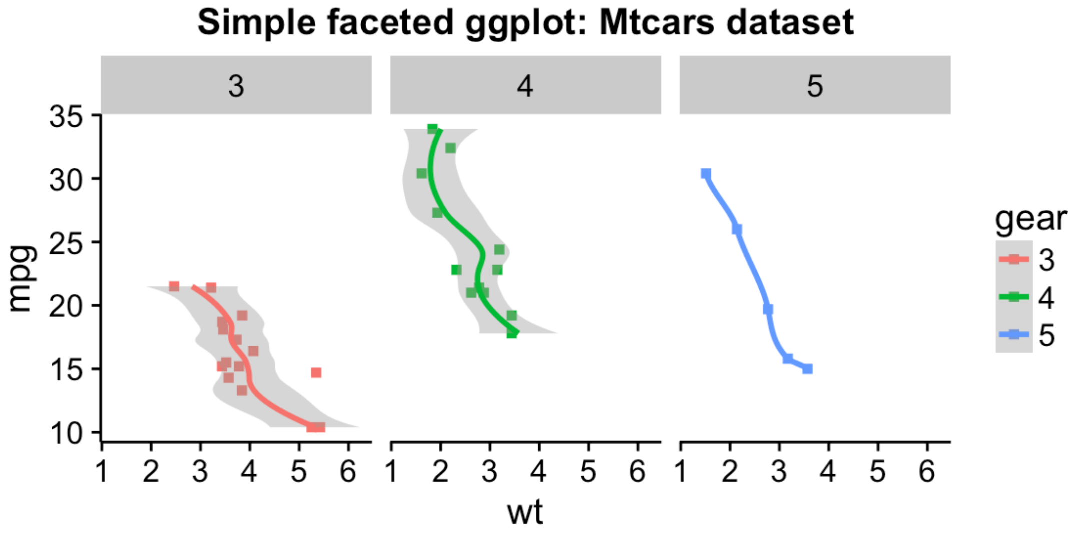
Three geoms layered + fine-tuned

```
g + geom_point(shape=15) +
  geom_smooth(aes(fill=gear)) +
  geom_rect(aes(xmin=22.5,xmax=35,ymin=4,ymax=5),fill="white")+
  annotate("text", x=30,y=4.5,label="Lots of Regresion Lines") +
  labs(title="Simple ggplot: Mtcars dataset")
```



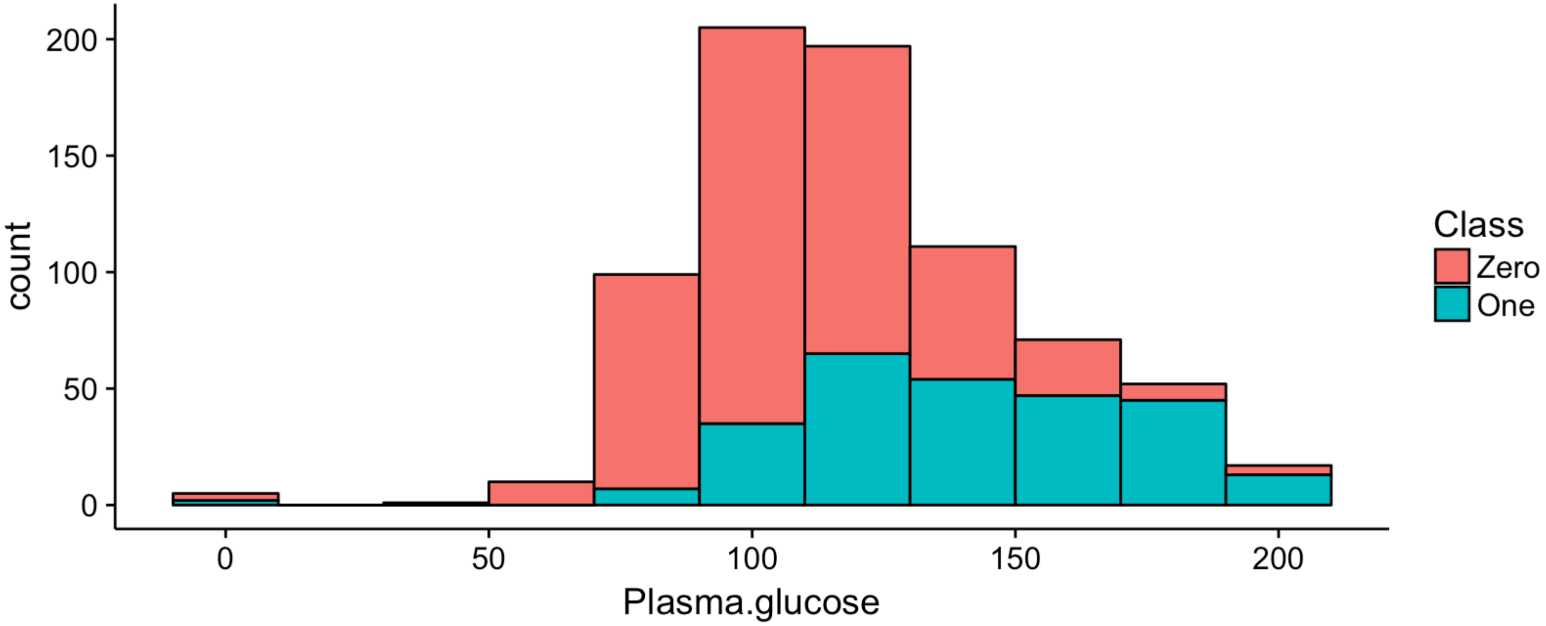
Layers together + Change perspective

```
g + geom_point(shape=15) +
  geom_smooth() +
  coord_flip() +
  facet_wrap(~gear) +
  labs(title="Simple faceted ggplot: Mtcars dataset")
```



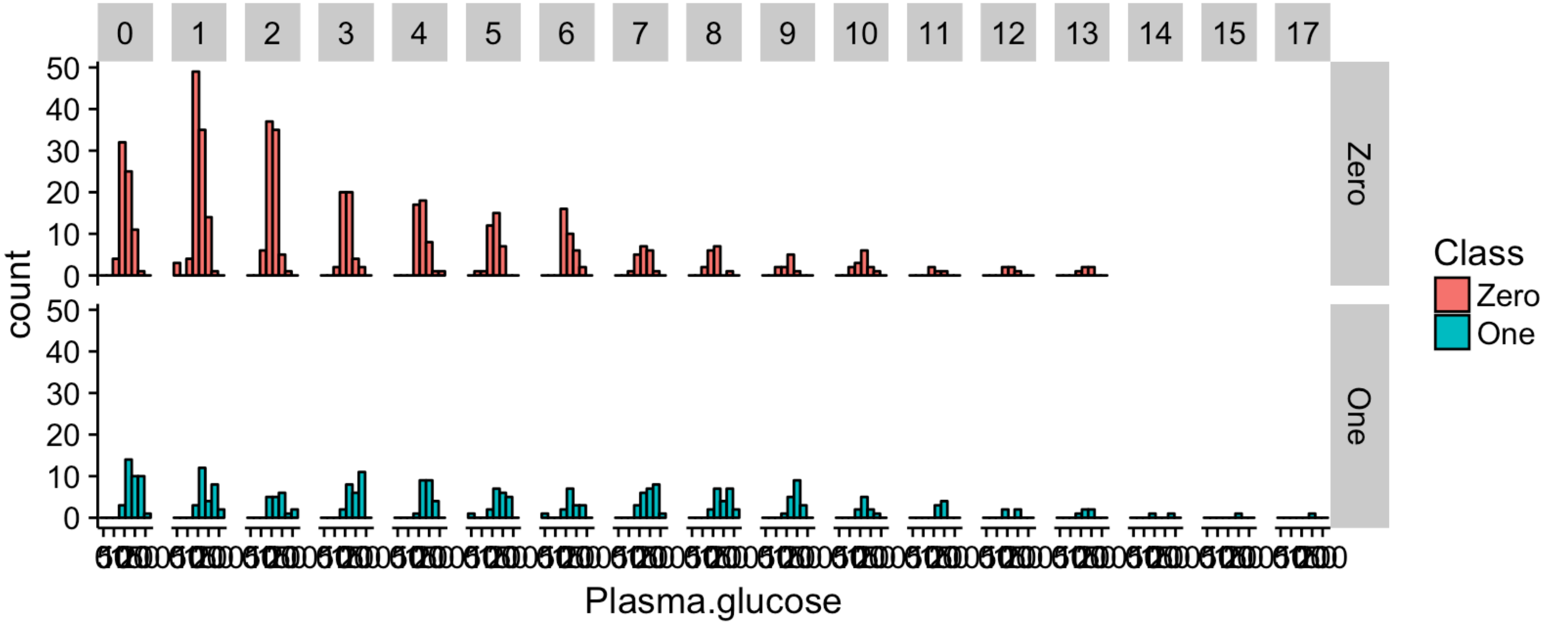
geom_histogram(1)

```
pima %>% ggplot(.,aes(x=Plasma.glucose, fill=Class)) +
  geom_histogram(binwidth=20,colour='black')
```



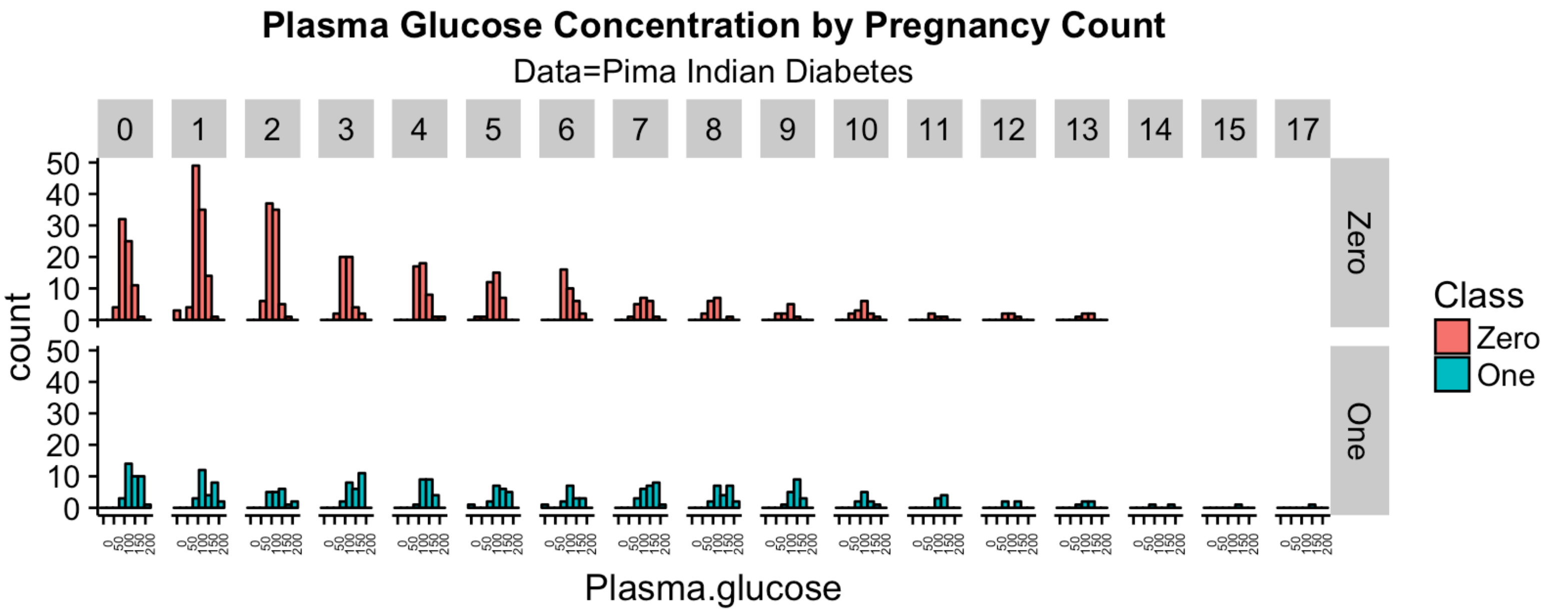
geom_histogram(2)

```
pima %>% ggplot(.,aes(x=Plasma.glucose, fill=Class)) +
  geom_histogram(binwidth=30,colour='black') +
  facet_grid(Class~pregnancy.count)
```



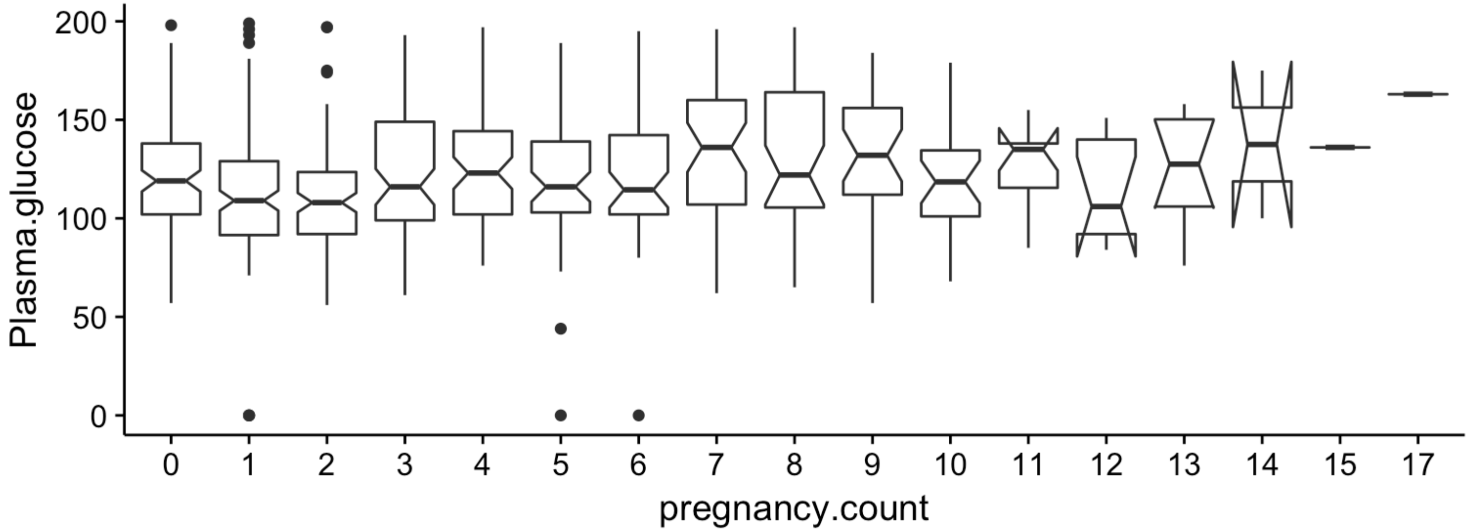
geom_histogram(3)

```
pima %>% ggplot(.,aes(x=Plasma.glucose, fill=Class)) +
  geom_histogram(binwidth=30,colour='black') +
  facet_grid(Class~pregnancy.count) +
  labs(title="Plasma Glucose Concentration by Pregnancy Count",
       subtitle="Data=Pima Indian Diabetes") +
  theme(axis.text.x = element_text(size=5,angle=90),
        plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```



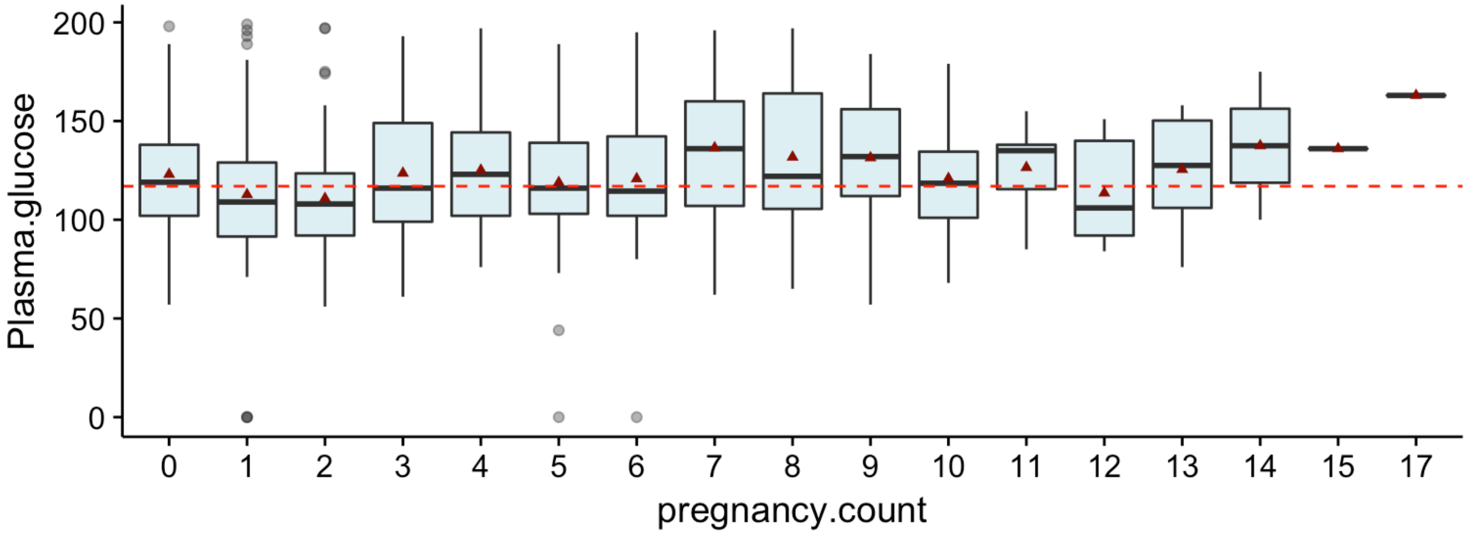
geom_boxplot(1)

```
pima %>% ggplot(.,aes(x=pregnancy.count,y=Plasma.glucose)) +
  geom_boxplot(notch=TRUE)
```



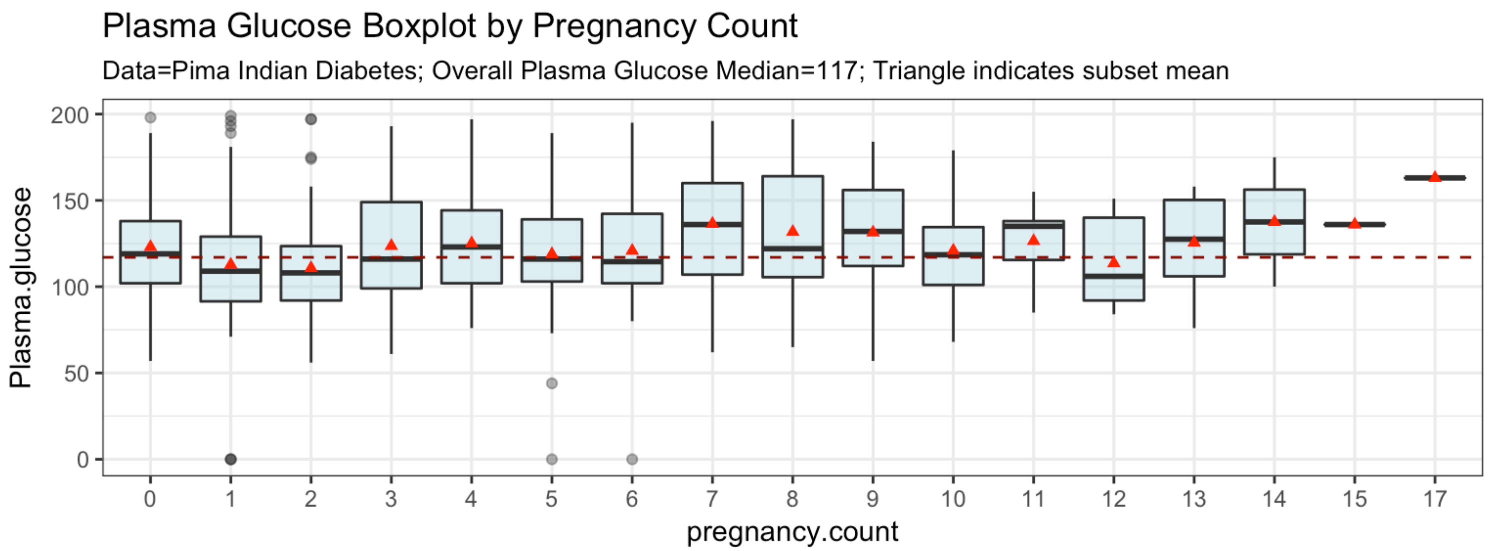
geom_boxplot(2)

```
pg.median <- pima[,2] %>% median()

pima %>% ggplot(.,aes(x=pregnancy.count,y=Plasma.glucose)) +
  geom_boxplot(fill='lightblue',alpha=0.4,notch=FALSE)+
  geom_hline(yintercept = pg.median,colour='red',linetype="dashed") +
  stat_summary(fun.y = "mean", geom = "point", size= 1.3,shape=17,colour="red4")
```



geom_boxplot(3)

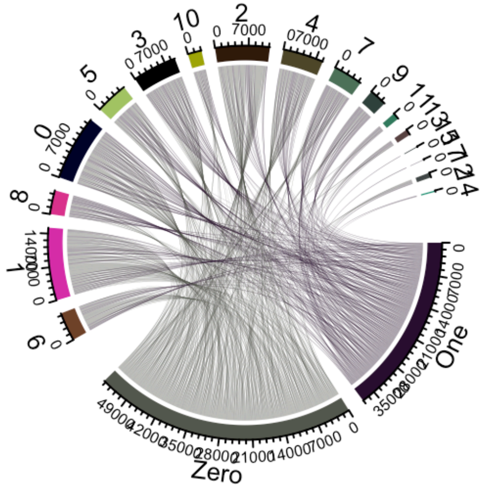
```
pima %>% ggplot(.,aes(x=pregnancy.count,y=Plasma.glucose)) +
  geom_boxplot(fill='lightblue',alpha=0.4,notch=FALSE)+
  geom_hline(yintercept = pg.median,colour='red4',linetype="dashed") +
  stat_summary(fun.y = "mean", geom = "point", size= 1.5,shape=17,colour="red")+
  theme_bw() +
  labs(title="Plasma Glucose Boxplot by Pregnancy Count",
       subtitle="Data=Pima Indian Diabetes; Overall Plasma Glucose Median=117; Triangle indicates subset mean")
```



Chord Diagram

How would you describe?
Imagine a different co-ordinate system.

```
library(circlize)
# Class vs pregnancy Count
circos.clear()
pima2=pima[,c(9,1,2)]
circos.par("gap.degree" = c(5,15,rep(4,times=16),15))
chordDiagram(pima2)
circos.clear()
```



Assembly

Recall graph vs graphic definition.

The Assembly stage parallels the specifications.

- A frame object is constucted.
    - parse frame model/executing algebra calcs
    - link variables to data
    - associate variable to model dimensions
    - transformations & scales.
- Graph Computations on frames
    - frames can be parallel processed
    - frames cached
- Each graph frame does something different with data

Display

The rendering of the graph and its aesthetic attributes to a final display system graphic.

Rendering may not just be 2 dimensional computer screen...

Not long ago rendering traditionally was paper.

What are rending issues with augmented/virtual reality displays?

Exploring Data

Moving away from static graphics.

With advances in technology - many want to interact with data.

Interaction should be intuitive.

How do you specify a human interaction experience?

Animations, interactive data?

Some solutions -> ggvis, ggraptR, animations, plotly, rbokeh

Interaction

- Filtering
    - Discrete Data > Checkboxes
    - Continuous Data > Sliders
    - Direct Data > Lasso's
- Navigating
    - Zooming > note changing axis values
    - Panning > Changing data region
    - Rotating axis > eg RGL 3dplot
- Manipulating
    - Dragging data points
    - Reordering Categories
- Brushing/Linking
    - Selecting/highlighting a subset of data in one plot - Shown in otherplots
    - iplots package
- Animations

Plotly

Simple interactive plot - one function converts ggplot object.

Filtering discrete data, navigating tols

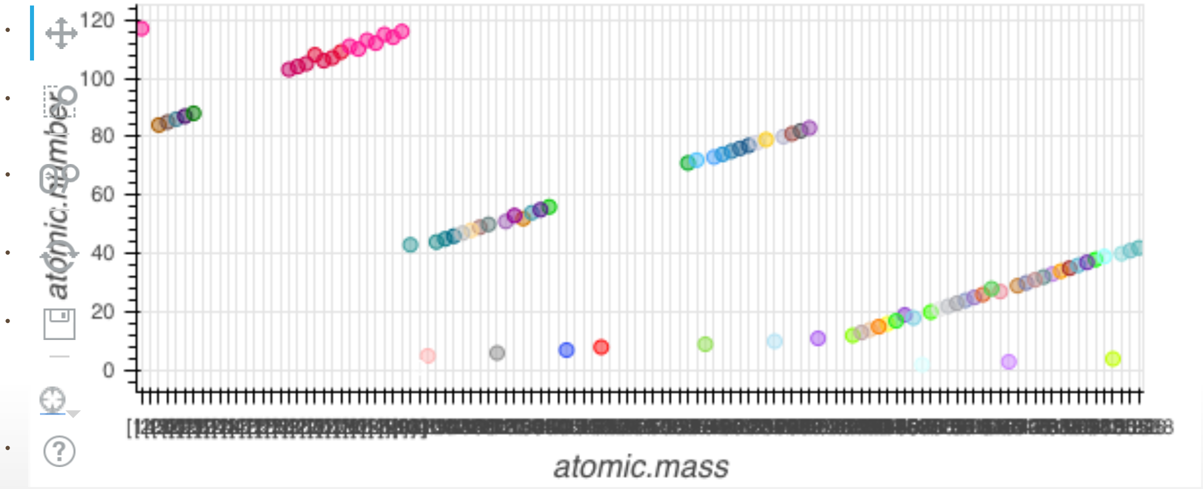
```
library(plotly)
attach(msleep)

ms1 <- ggplot(msleep,aes(x=awake,colour=vore)) + geom_histogram()

ggplotly(ms1)
```

Rbokeh

ggplot style syntax & layers. Uses the term "glyph" for components.

Hover tool extracts non-visualised data values.

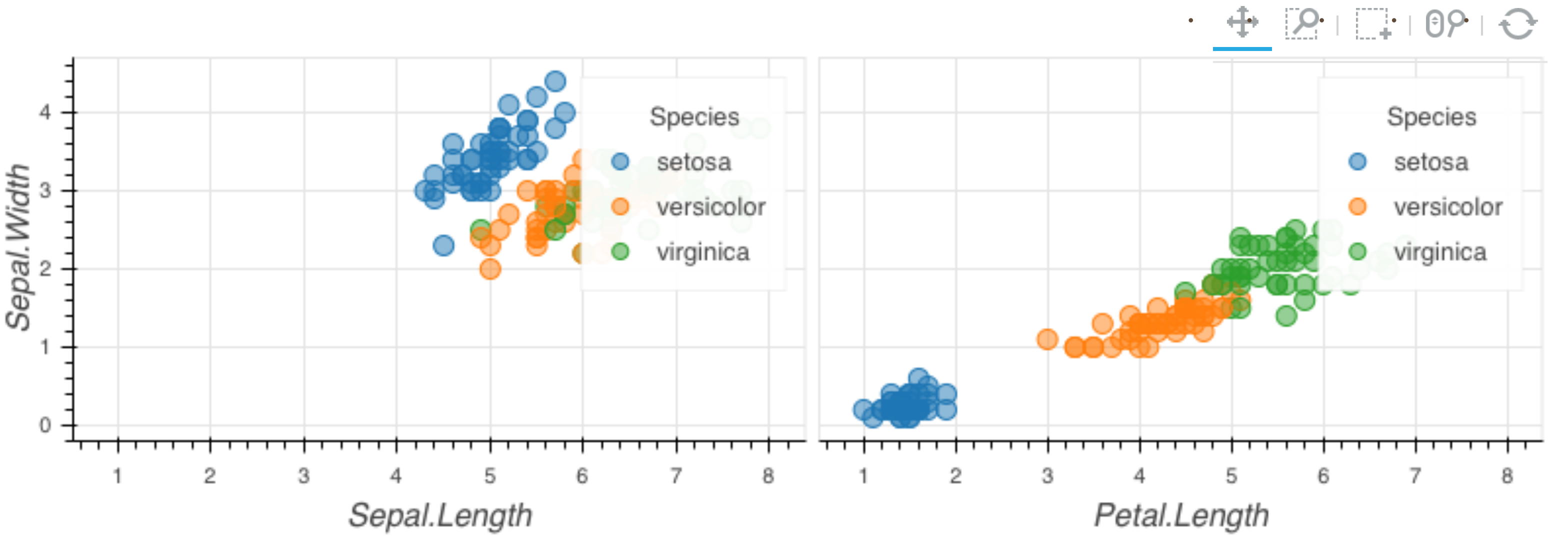
```
library(rbokeh)

elements %>%
  figure(width = 600, height = 250) %>%
  ly_points(x=atomic.mass,y=atomic.number,
            size=7,
            color=CPK,
            hover=list(symbol, name, CPK))
```



Rbokeh II

Linking Plots; tools specified

```
tools <- c("pan", "wheel_zoom", "box_zoom", "box_select", "reset")
p1 <- figure(tools = tools, width = 400, height = 300) %>%
  ly_points(Sepal.Length, Sepal.Width, data = iris, color = Species)
p2 <- figure(tools = tools, width = 400, height = 300) %>%
  ly_points(Petal.Length, Petal.Width, data = iris, color = Species)
grid_plot(list(p1, p2), same_axes = TRUE, link_data = TRUE)
```



R?

PDF of slides will be made available.

http://www.linkedin.com/in/justincastagna

https://github.com/jcastagna/0.Presentations