ggplot2

Grammar of Graphics

2 principles

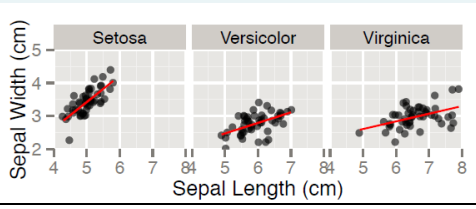
1. Graphics = distinct layers of grammatical elements
2. Meaningful plots through aesthetic mapping

Element	Description
Data	The dataset being plotted.
Aesthetics	The scales onto which we *map* our data.
Geometries	The visual elements used for our data.
Facets	Plotting small multiples.
Statistics	Representations of our data to aid understanding.
Coordinates	The space on which the data will be plotted.
Themes	All non-data ink.

Data	*{variables of interest}*				
Aesthetics	*x-axis* *y-axis*	*colour* *fill*	*size* *labels*	*alpha* *shape*	*line width* *line type*
Geometries	*point*	*line*	*histogram*	*bar*	*boxplot*
Facets	*columns*	*rows*			
Statistics	*binning*	*smoothing*	*descriptive*	*inferential*	
Coordinates	*cartesian*	*fixed*	*polar*	*limits*	
Themes	*non-data ink*				

```
ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point()
ggplot(mtcars, aes(x = wt, y = mpg, color = disp)) + geom_point()
ggplot(mtcars, aes(x = wt, y = mpg, size = disp)) + geom_point()
```

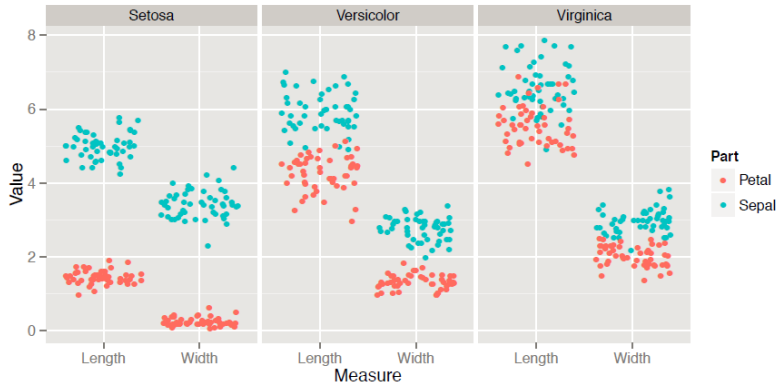

```
> levels(iris$Species) <- c("Setosa", "Versicolor", "Virginica")
> ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
    geom_jitter(alpha = 0.6) +
    facet_grid(. ~ Species) +
    stat_smooth(method = "lm", se = F, col = "red") +
    scale_y_continuous("Sepal Width (cm)",
                       limits = c(2,5),
                       expand = c(0,0)) +
    scale_x_continuous("Sepal Length (cm)",
                       limits = c(4,8),
                       expand = c(0,0)) +
    coord_equal()
```



Data

iris.wide (pg15) & iris.tidy (pg23) & facet_grid()

```
> ggplot(iris.tidy, aes(x = Measure, y = Value, col = Part)) +
    geom_jitter() +
    facet_grid(. ~ Species)
```



Aesthetics Layer

- Column can be mapped onto visible ***aesthetic***
- Aesthetics in aes(), ***attributes*** in geom_(col="red")
- aes() can also be called in geom_(), but done usually when you want to include multiple data sources
- ggplot(mtcars, aes(x=wt, y=mpg, fill=cyl, col=am)) + geom_point(**shape**=21, **size**=4, **alpha**=0.6)
    - aes has to be associated with columns
    - attributes are given along with geom_*() and don't have columns associated with them
- ggplot(mtcars, aes(x = wt, y = mpg, fill = cyl, **label**=rownames(mtcars))) + geom_**text**(color='red')
- **Modifying Aesthetics**
    - geom_bar(**postion**="< stack, fill, dodge, … >")
    - **scale_*** functions
        - scale_x/y_continuous/discrete("title", limits, breaks, expand, ….)
        - ex: scale_x_continuous("x-axis", limits=c(0,10), breaks=seq(0,10,2))
    - **labs**(x,y,col, …)

Aesthetics for Continuous Variables

Aesthetic	Description
x	X axis position
y	Y axis position
size	Diameter of points, thickness of lines
alpha	Transparency
colour	Colour of dots, outlines of other shapes
fill	Fill colour

Aesthetics for Categorical Variables

Aesthetic	Description
labels	Text on a plot or axes
fill	Fill colour
shape	Shape of point
alpha	Transparency
linetype	Line dash pattern
size	Diameter of points, thickness of lines

Geometry Layer

Scatter Plots: geom_point()

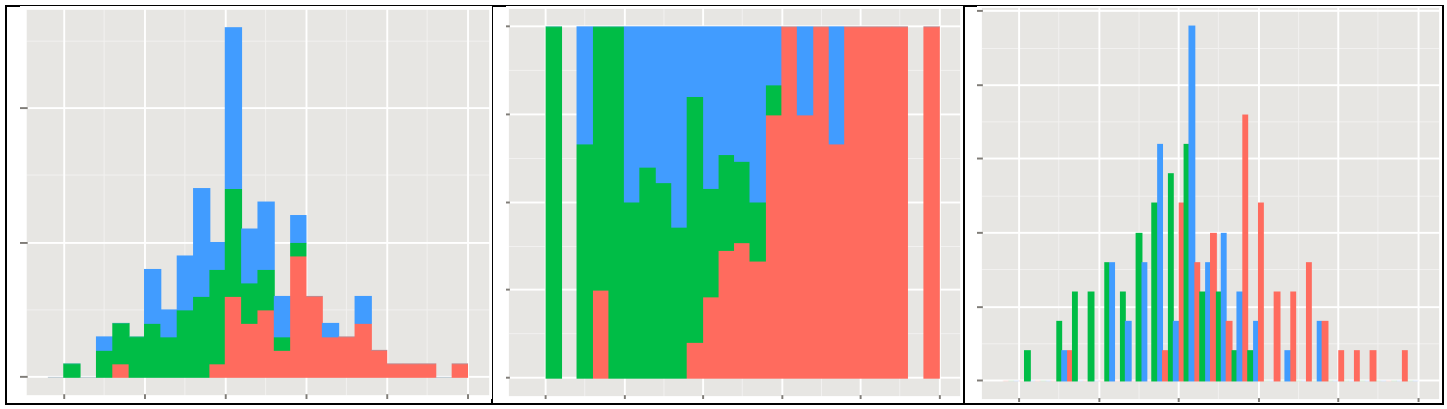
- aes() inside geom_*() can be used to use different aesthetics for different layer of geom_*() . Same goes for different data for different geom_*()
- aes() inside geom_*() is same as aes() in ggplot()

```
> ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, col = Species)) +
    geom_point() +      inherits data and aes from ggplot()
    geom_point(data = iris.summary, shape = 15, size = 5)    different data inherits aes
```

- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, col = Species)) + geom_**jitter**(**shape**=1, **alpha**=0.6)
    - to visualize the density, use jitter along with alpha & shape(hollow shapes preferred)

Bar Plots:

- **Histogram**: geom_histogram() : x-axis : continuous variables
    - ggplot(df, aes(x=x1)) + geom_histogram(**binwidth**=0.1)
    - ggplot(df, aes(x=x1)) + geom_histogram(aes(y=**..density..**), binwidth=0.1)
    - ggplot(df, aes(x=x1, fill=cat_var)) + geom_histogram(binwidth=0.1, **position**="stack/fill/dodge")



- **Bar Plot**: geom_bar() : x-axis = categorical variables
    - ggplot(df, aes(x=cat_var)) + geom_bar(stat="bin")
    - Custom Color Palettes

```
blues <- brewer.pal(9, "Blues")
blue_range <- colorRampPalette(blues)
ggplot(Vocab, aes(x = education, fill = vocabulary)) +
    geom_bar(position = "fill") +
    scale_fill_manual(values=blue_range(11))
```

    - Overlapping bar plots

```
posn_d <- position_dodge(width=0.2)
ggplot(mtcars, aes(x = cyl, fill = am)) + geom_bar(position=posn_d)
```

- **Line Plots:** geom_line()
    - Plotting different categories

```
ggplot(df, aes(x=Year, y=Capture, linetype=Species)) + geom_line()
```

    - Proportional Trends

```
ggplot(df, aes(x=Year, y=Capture, fill=Species)) + geom_area(position="fill")
```

- ggplot(economics, aes(x=date, y=unemploy/pop)) + **geom_rect**(data=recess, aes(xmin=begin, xmax=end, ymin=-Inf, ymax=+Inf), **inherit.aes=FALSE**, fill="red", alpha=0.2) + geom_line()

qplot

- Quick and dirty way for plotting, not very flexible, doesn't follow grammar of graphics
- qqplot(x, y, data, shape/size/col, postion, jitter, alpha=I(value))

Wrap-Up

iris	Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
iris.wide	Species	Part	Length	Width	
iris.mixed	Species.Part	Length	Width		
iris.tidy	Species	Part	Measure	Value	

Choice of data format depends on desired plot!

Statistics Layer

- Two types: called within a geom and called independtly

stat_	description
stat_summary()	Summarise y values at distinct x values
stat_function()	Compute y values from a function of x values
stat_qq()	Perform calculations for a quantile-quantile plot

- **stat_smooth**(method="loess/lm/…", se, aes(group=1, col="text"), span)
  - ex: ggplot(Vocab, aes(x = education, y = vocabulary, col = year, group = factor(year))) + stat_smooth(method = "lm", se = FALSE, alpha = 0.6, size = 2) + scale_color_gradientn(colors = brewer.pal(9, "YlOrRd"))

- **stat_summary**(geom="***errorbar***", fun.data=mean_sdl, fun.args=list(…), witdh) # adds error bars
- **stat_summary**(geom="***point***", fun.y=mean) # adds point for the mean values
- **stat_summary**(geom="***linerange***", fun.data =<custom_function>, position=posn.d, size=3)

- **stat_function**(fun=dnorm, colour, arg=list(…))

Coordinate Layer

- zooming-in : + coord_cartesian(xlim=c(4.5, 5.5))
- aspect ratio : + coord_fixed(0.055)

Facets Layer

- … + facet_grid(row ~ column)

Themes Layer

text	line	rect
title	axis.ticks	legend.background
plot.title	axis.ticks.x	legend.key
legend.title	axis.ticks.y	panel.background
axis.title	axis.line	panel.border
axis.title.x	axis.line.x	plot.background
axis.title.y	axis.line.y	strip.background
legend.text	panel.grid	
axis.text	panel.grid.major	
axis.text.x	panel.grid.major.x	
axis.text.y	panel.grid.major.y	
strip.text	panel.grid.minor	
strip.text.x	panel.grid.minor.x	
strip.text.y	panel.grid.minor.y	

- … + theme(plot.background=element_rect(color=”black”, size=3))
- … + theme(panel.grid=element_line(color=”red”))
- … + theme(axis.text=element_text(color=”red”))
- my.theme <- theme(…)
- theme_update(…)
- theme_set(…)