Visualizations in R

PSS SUMMER SCHOOL

Anton Badev, Daniel Nikolic, Justin Skillman

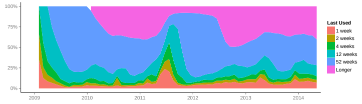
June 28, 2017

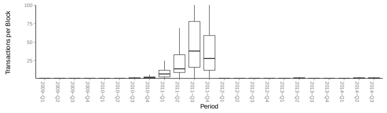
What is ggplot2

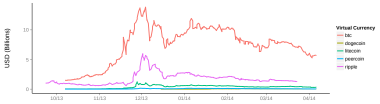
The package uses ideas outlined in the paper *Grammar of Graphics* by Leland Wilkison to create a general approach to statistical plotting. A few of the major benefits of ggplot over conventional graphics plotting systems are:

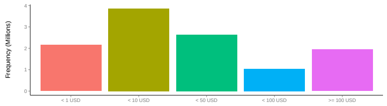
1. Ability to generate almost any type of conventional 2D graph
2. Building blocks allow for as much customization as desired
3. Uniform grammar allows for a consistent way of plotting (once you understand it)

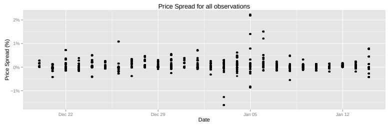
Example of Graphics

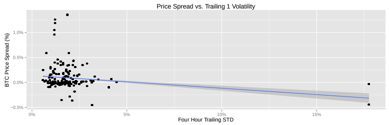












General Philosophy

Plotting in this package takes a layered approach. We can think of this as a step by step procedure in which each step contributes to the final result.

- Aesthetics - Mapping input data to what is displayed
- Geom - Uses the mappings to generate shapes on the graph
- Stats - Aggregates measures to generate statistical objects on the graph

Quick Plotting

Quick way to generate graphs without the full power of customization allowed in ggplot. Uses ggplot in the back end with common default arguments.

- **data** - input data frame
- **x, y** - x and y coordinates of graph
- **geom** - geometric object to display data
- **main, xlab, ylab** - graph labels
- **stat** - Summary statistics to graph

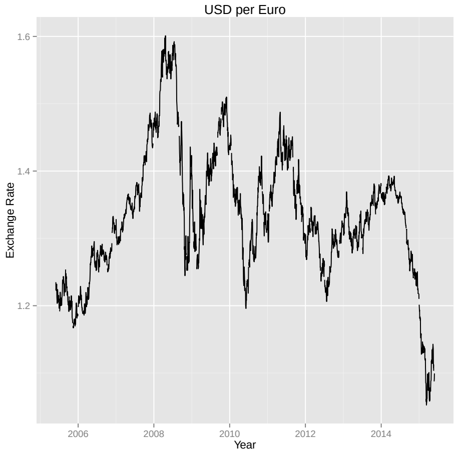
Example: Euro USD Exchange Rate

```
> str(fx_rate)
```

```
'data.frame': 2513 obs. of  2 variables:
 $ date    : Date, format: "2005-05-31" "2005-06-01" ...
 $ usd_euro: num  1.23 1.22 1.23 1.22 1.23 ...
```

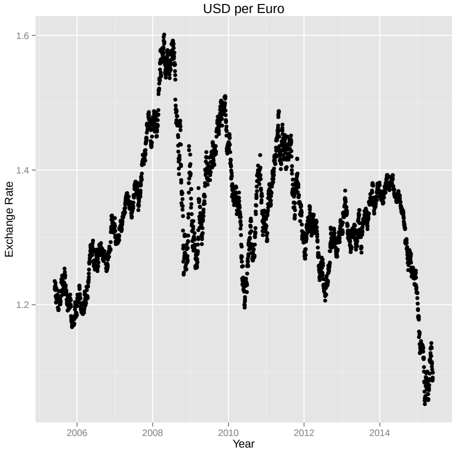
Line Graph

```
qplot(data=fx_rate, x=date, y=usd_euro, geom="line",
      main='USD per Euro', ylab='Exchange Rate', xlab='Year')
```



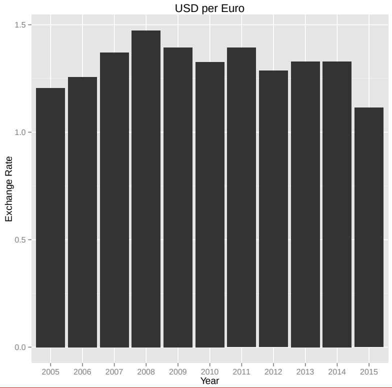
Scatter Plot

```
qplot(data=fx_rate, x=date, y=usd_euro, geom="point",
      main='USD per Euro', ylab='Exchange Rate', xlab='Year')
```



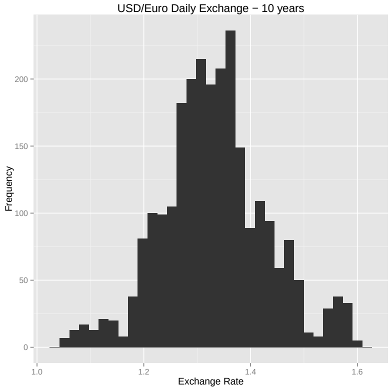
Bar Graph

```
fx_rate$year <- format(fx_rate$date, "%Y")
df <- summarise(group_by(fx_rate, year),
                musd_euro=mean(usd_euro, na.rm=T))
qplot(data=df, x=year, y=musd_euro, geom="bar", stat="identity",
      main='USD per Euro', ylab='Exchange Rate', xlab='Year')
```



Histogram

```
qplot(data=fx_rate, x=usd_euro, geom="histogram",
      main='USD/Euro Daily Exchange - 10 years',
      xlab='Exchange Rate', ylab='Frequency')
```



ggplot Function

Syntax

```
g <- ggplot(data, mapping) + [Options]

pdf([Output Path])
plot(g)
dev.off()
```

- Create the graph object first, then pass it to the generic plot function
- To output the graph as a file specify a standard output function (pdf,jpeg,png,etc) with the path
- The output function also takes arguments to customize the output size of the figure

Graph Object

Composed of a set of layers which each define different parts of the visualization. Some common parameters defining the components of a layer are as follows:

- **mapping** - Aesthetic mapping between data and visual objects
- **data** - input data frame to be visualized
- **geom** - geometric object or shape used to draw the graphic
- **stat** - the statistical transformation to use
- **position** - used to adjust positions, overlap or stack objects

Note: not all of these need to be used at once, for example, only 1 could be used. Also they may reference different datasets within one graphic!

Example: Cars Dataset

```
> data(mtcars)
> str(mtcars)
```

```
'data.frame': 32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 ...
 $ drat: num  3.9 3.9 3.85 3.08 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Aesthetic Mapping

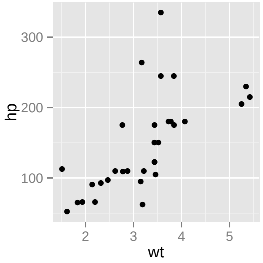
Command that maps data to how it is going to be plotted. Here we tag which variables are going to be plotted as x, y coordinates and colours. Colours are groupings of data that we want to display differently.

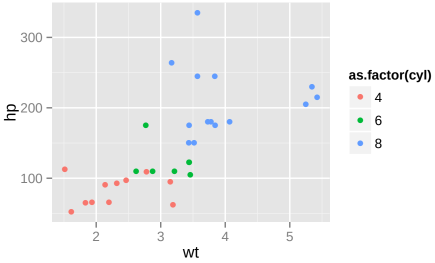
Syntax

```
aes(x, y, colour, group)
```

Aesthetic Examples 1

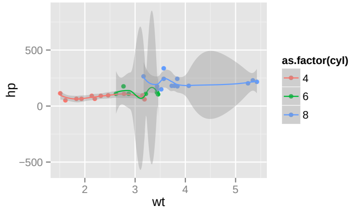
```
# Left Plot
g <- ggplot(data=mtcars, aes(x=wt, y=hp)) + geom_point()

# Right Plot
g <- ggplot(data=mtcars, aes(x=wt, y=hp,
            colour=as.factor(cyl))) + geom_point()
```

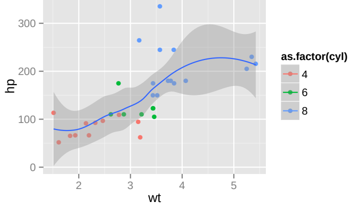




Aesthetic Examples 2

```
# Left Plot
g <- ggplot(data=mtcars, aes(x=wt, y=hp, colour=as.factor(cyl))) +
            geom_point() + geom_smooth()

# Right Plot
g <- ggplot(data=mtcars,
            aes(x=wt, y=hp, colour=as.factor(cyl), group=1)) +
            geom_point() + geom_smooth()
```

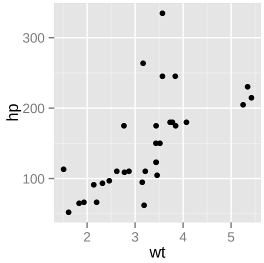




Additional Layers

Now that we have the mapping between the raw data and our graphic we can add a layer which shows the data.

```
g <- ggplot(data=mtcars, aes(x=wt, y=hp))
            + layer(geom="point")
```



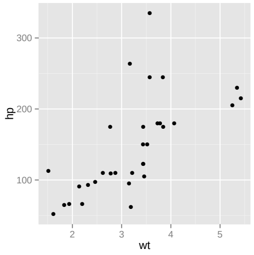
Geoms

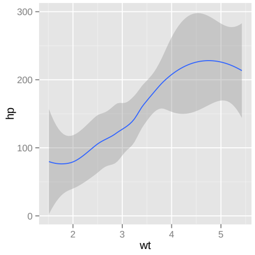
Using geom syntax is really shorthand for adding a layer with some defaults. For instance the following statements will generate the same graph.

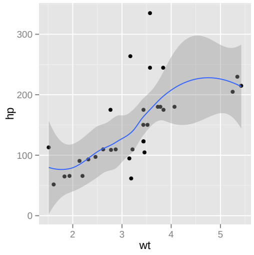
```
# Using the layer function
g <- ggplot(data=mtcars, aes(x=wt, y=hp))
            + layer(geom="point")

#Using the geom_XXX() function
g <- ggplot(data=mtcars, aes(x=wt, y=hp))
            + geom_point()
```

Example: Multiple Geoms

```
g <- ggplot(data=mtcars, aes(x=wt, y=hp))
     + geom_point() + geom_smooth()
```





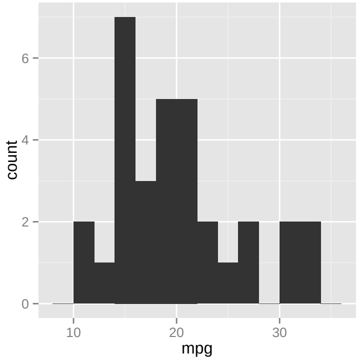


stat

Calculate summary statistics on data and plotting the results. This is another layer with defaults (including geom) which can be overwritten for custom results.

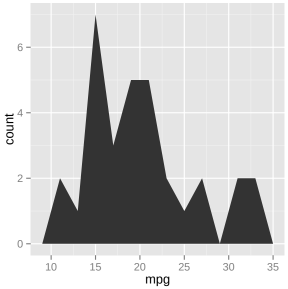
```
# Using stats argument in geom function
g <- ggplot(data=mtcars, aes(x=mpg))
     + geom_bar(stat="bin", binwidth=2)

# Using geom argument in stats function
g <- ggplot(data=mtcars, aes(x=mpg))
     + stat_bin(geom="bar", binwidth=2)

# Default arguements in stats function
g <- ggplot(data=mtcars, aes(x=mpg))
     + stat_bin(binwidth=2)
```

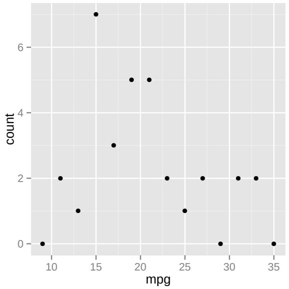
Example: stat



Example: Changing Defaults

```
g <- ggplot(data=mtcars, aes(x=mpg))
     + stat_bin(geom="area", binwidth=2)

g <- ggplot(data=mtcars, aes(x=mpg))
     + stat_bin(geom="point", binwidth=2)
```





Faceting

Create a set of graphs which each split the data based on some variable. This can generate a 2D array of graphs.

Syntax

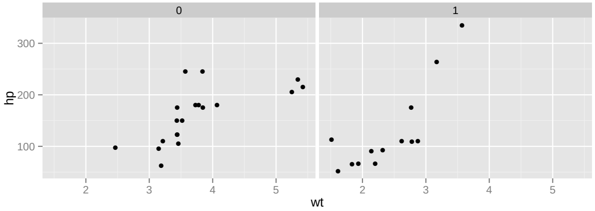
```
+ facet_grid([row var] ~ [col var], scales)

+ facet_wrap([row var] ~ [col var], ncols, scales)
```

Scale Options
free
fixed
free_x
free_y

Example : Faceting Grid

Horse Power (hp) by weight (wt) for manual and automatic transmission

```
g <- ggplot(data=mtcars, aes(x=wt, y=hp))
     + layer(geom="point") + facet_grid(. ~ am)
```

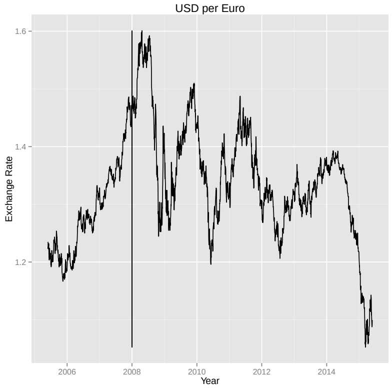


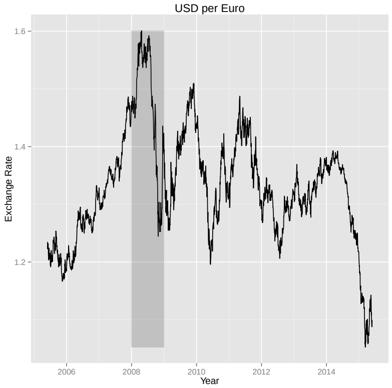
Annotating Figures

Annotations can be through of as adding additional layers of data to a plot.

```
g <- ggplot(fx_rate, aes(x=date, y=usd_euro)) + geom_line() +
            labs(x='Year', y='Exchange Rate') +
             ggtitle('USD per Euro')

g <- g + geom_segment(x=as.numeric(as.Date('2008-01-01')),
                      xend=as.numeric(as.Date('2008-01-01')),
                      y=yrng[1], yend=yrng[2])

g <- g + annotate("rect", xmin=as.Date('2008-01-01'),
                          xmax=as.Date('2009-01-01'),
                          ymin=yrng[1], ymax=yrng[2],
                          alpha=0.2)
```

Example: Annotating Figures





Scales

Map data from the data domain to the aesthetic range. Default scales are generated automatically but can be customized.

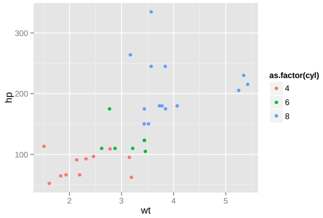
```
scale_x_continuous(breaks, labels, limits)
scale_x_discrete(breaks, labels, limits)
scale_x_date(breaks, labels, limits)
```

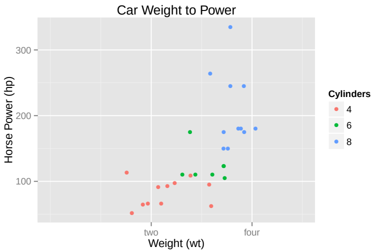
Labels

```
+ labs(x = "", y="", colour="")
+ ggtitle("")
```

Example: Scales

```
g <- ggplot(data=mtcars, aes(x=wt, y=hp, colour=as.factor(cyl))) +
  geom_point() +
  labs(x="Weight (wt)", y="Horse Power (hp)", colour="Cylinders") +
  ggtitle("Car Weight to Power") +
  scale_x_continuous(breaks=c(2,4), labels=c('two','four'),
                              limits=c(0,5))
```





Themes

Address the look and feel of the visualization without modifying the underlying data display.

Built-in Themes

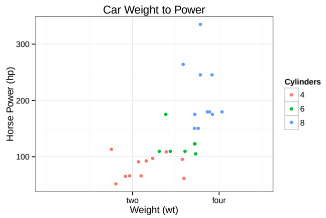
- theme_gray()
- theme_bw()

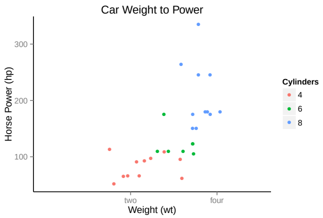
Finer Detail

```
+ theme([theme element] = [theme value])
```

Example: Themes

```
# Left Graph
g <- g + theme_bw()

# Right Graph
g <- g + theme(axis.line = element_line(colour="black"),
               panel.grid.major = element_blank(),
               panel.grid.minor = element_blank(),
               panel.border = element_blank(),
               panel.background= element_blank())
```


Car Weight to Power
Horse Power (hp)
two
four
Weight (wt)
Cylinders


Car Weight to Power
Horse Power (hp)
two
four
Weight (wt)
Cylinders