Graphics in R

October 28, 2017

Introduction

Graphics can be customized in almost any conceivable way in R. You can change colors, create multipane plots, customize axis labels/tick marks, overlay multiple graphs, create custom textboxes, change size/shape of plotting characters and line, and on and on. You can also create a variety of 2-D and 3-D graphs including line graphs, boxplots and 3-D mesh graphs.

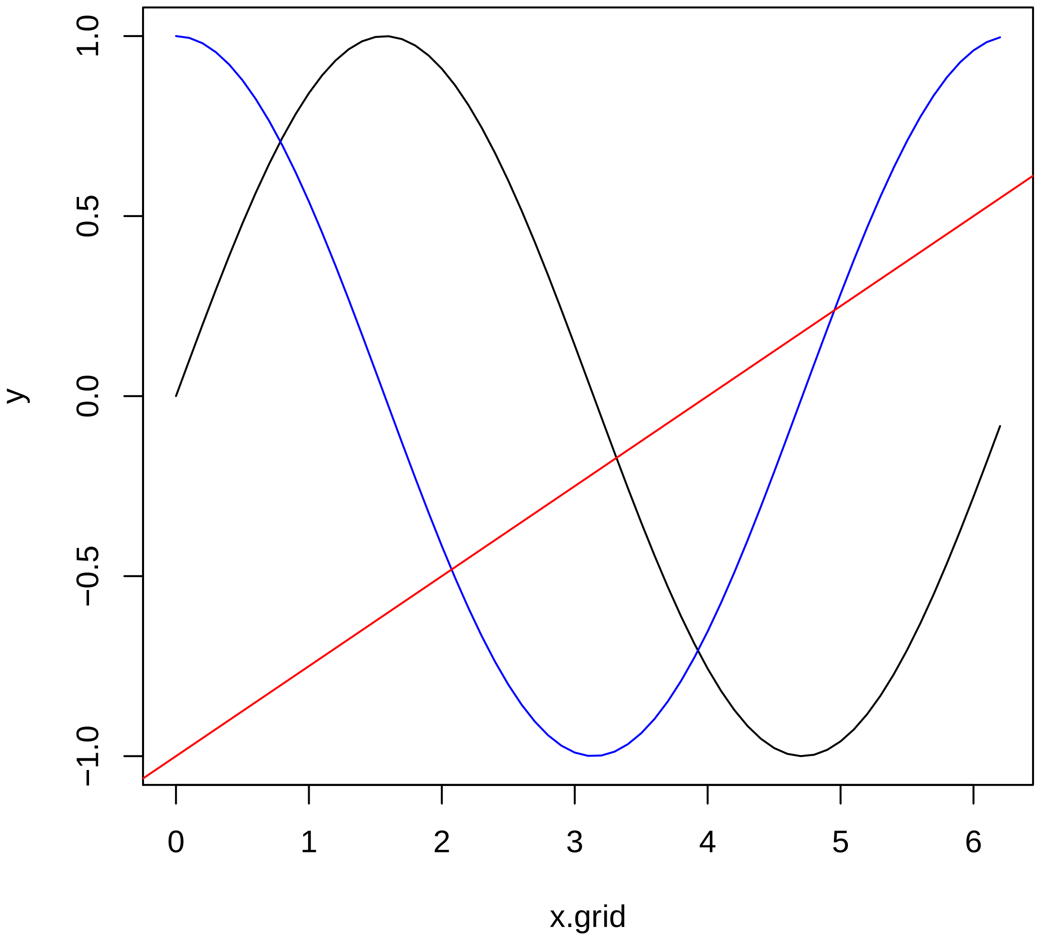
Ch. 12 of the Matloff text covers the basics of R's base or traditional graphics package. I will cover these topics and a bit more about the `lattice` package. If you want to learn more advanced graphics good references include Hadley Wickham's *ggplot2: Elegant Graphics for Data Analysis*, Deepayan Sarkar's *Lattice: Multivariate Data Visualization with R* and a few others in Matloff, p. 261.

You can see a list of all graphical parameters in R if you type `?par`.

Review

Recall we have already used the `plot(x,y,...), points(x,y,...), and abline(a,b,...)`. Let's plot the sine function, cosine function and the line $y = 0.25x - 1$ in the same plot area.

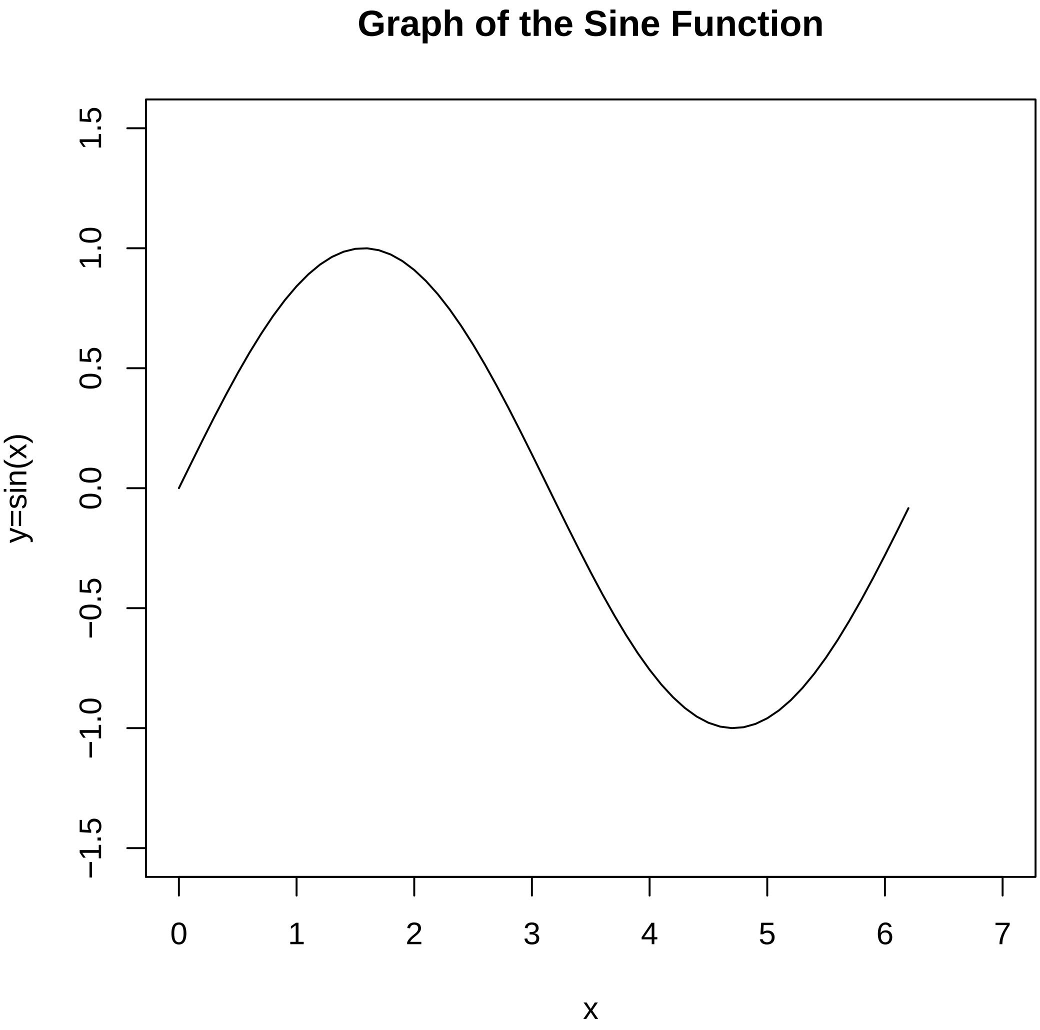
```
> x.grid <- seq(0,2*pi,by=0.1)
> y <- sin(x.grid)
> plot(x.grid,y,type="l")  #graph of sine
> points(x.grid,cos(x.grid), type="l",col="blue")  #graph of cosine
> abline(-1,0.25,col="red")  #y = -1+0.25x
```



Changing the Range of the Axes, Labelling Axes and Adding a Title

Axes ranges, labels and the title are set by arguments inside `plot()`.

```
> x.grid <- seq(0,2*pi,by=0.1)
> y <- sin(x.grid)
> plot(x.grid,y,type="l",xlim=c(0,7),ylim=c(-1.5,1.5),xlab="x",ylab="y=sin(x)",
+     main="Graph of the Sine Function")  #graph of sine
```



Boxplots and Fancier Scatter Plots

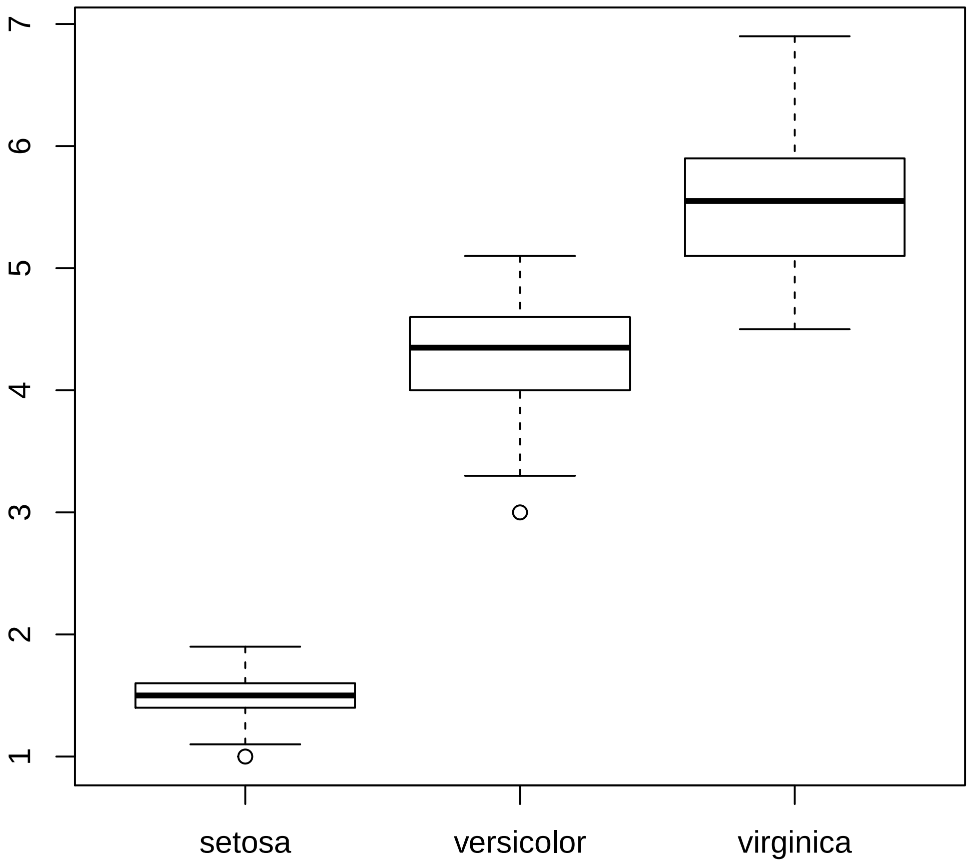
Let's work with the built-in iris dataset. We'll construct a boxplot to compare Petal Length across different species. *Boxplots are excellent for bivariate data where one variable is categorical and the other is numeric.*

**********Review boxplot on the board*******

The syntax is: `boxplot(formula, data = NULL, ..., subset, na.action = NULL)`

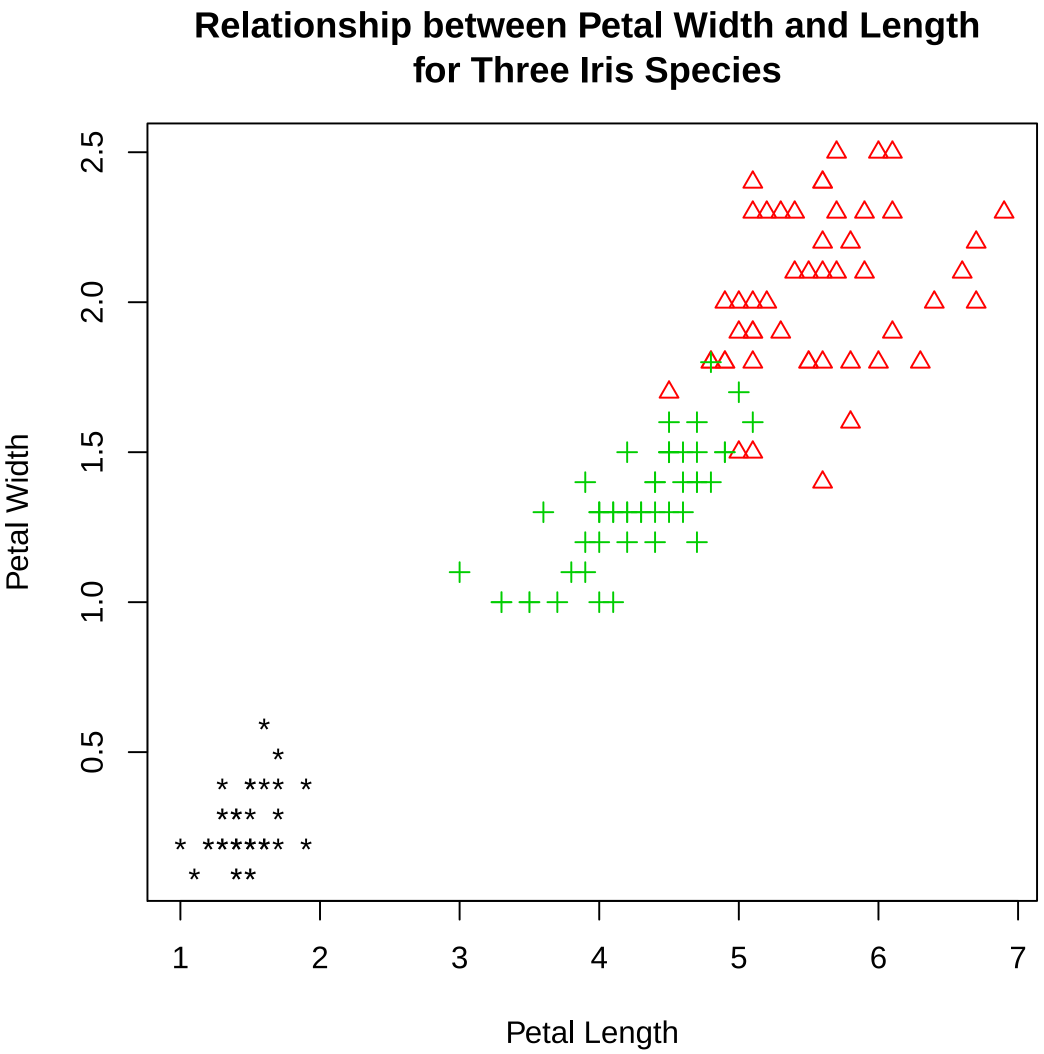
'formula' is something like $y \sim grp$ where y is a numeric vector of data to be split into groups according to the factor grp. If no grp is specified a single boxplot is constructed.

```
> head(iris,n=3)

  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa

> boxplot(Petal.Length ~ Species,data=iris)
```



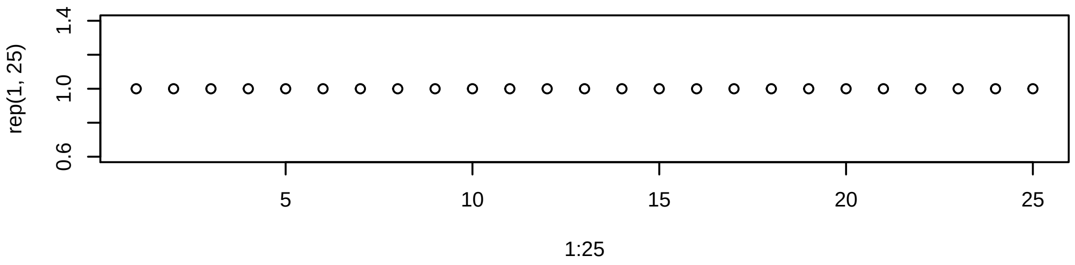
Now, we construct a trivariate plot. We will have an easier way to do this when we learn the ggplot2 package.

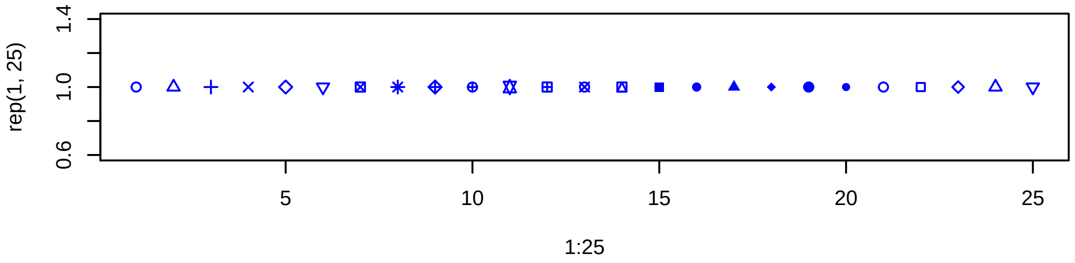
```
> attach(iris)   #column names will be assumed to be from iris
> x.limits = c(min(Petal.Length),max(Petal.Length))
> y.limits = c(min(Petal.Width),max(Petal.Width))
> iris.setosa = subset(iris,Species=="setosa")
> plot(iris.setosa$Petal.Length,iris.setosa$Petal.Width, xlim=x.limits,
+      ylim=y.limits,col=1,pch="*", xlab="Petal Length",ylab="Petal Width",
+      main="Relationship between Petal Width and Length \n for Three Iris Species")
> iris.virginica =subset(iris,Species=="virginica")
> points(iris.virginica$Petal.Length,iris.virginica$Petal.Width,col=2,pch=2)
> iris.versicolor =subset(iris,Species=="versicolor")
> points(iris.versicolor$Petal.Length,iris.versicolor$Petal.Width,col=3,pch=3)
```

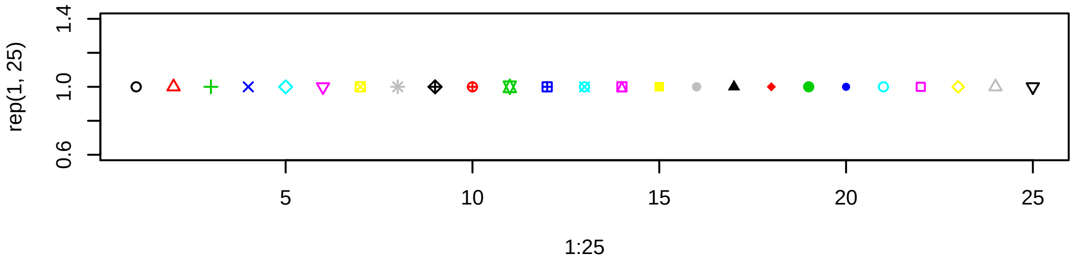


You can select from a variety of plotting characters using the argument, `pch`. Look under ?par for pch choices. You are then directed to help file for points.

```
> par(mfrow=c(3,1))  #multiframe plot with 3 rows and 1 column
> plot(1:25,rep(1,25),pch=1)
> plot(1:25,rep(1,25),pch=1:25,col=4)  #plot first 25 pch with color 4
> plot(1:25,rep(1,25),pch=1:25,col=1:25)   #ith point has pch=i, col=i
> #For example, the 3rd point has pch=3 and col=3
> par(mfrow=c(1,1))  #return to one plot pane
```







What is col=4? pch=24?

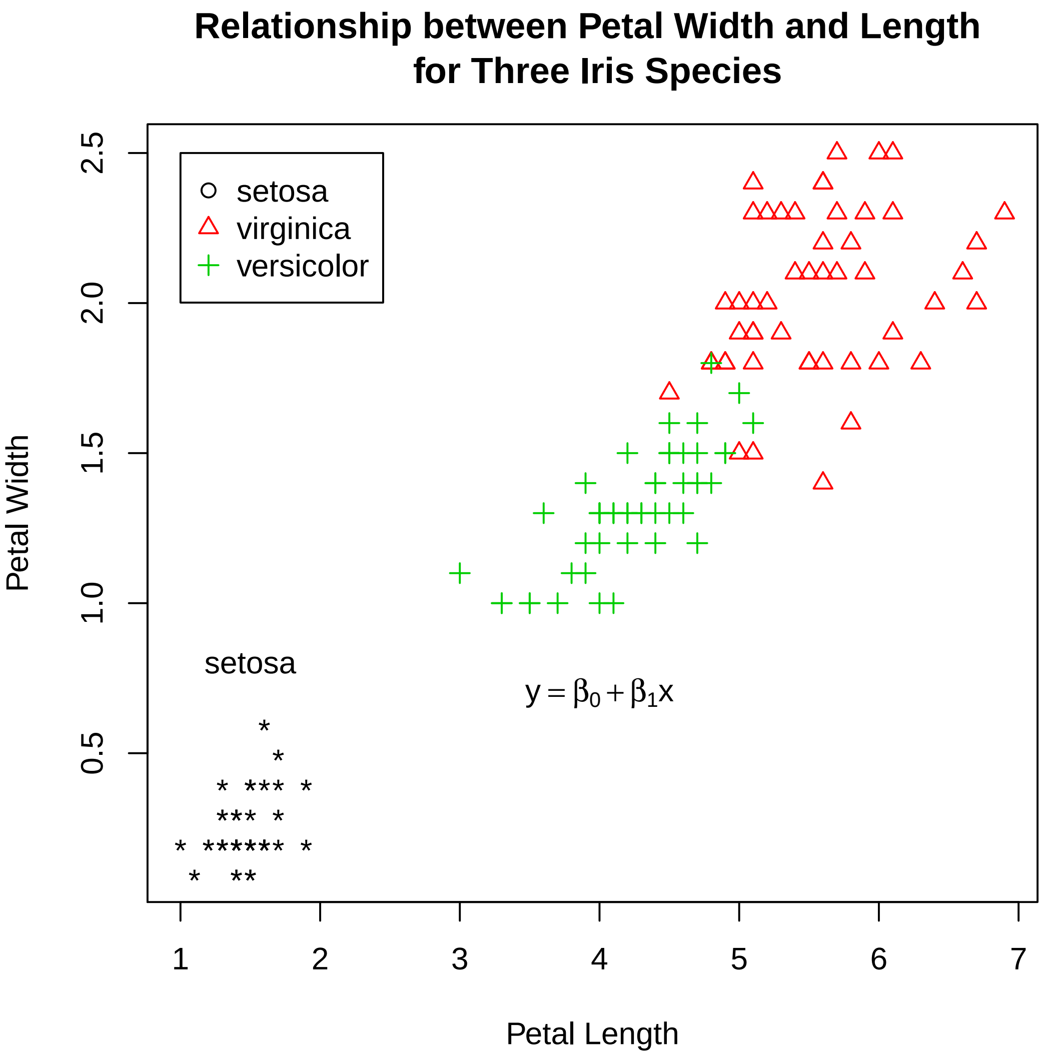
Legends and Text Annotations

Legend is a function to be used after `plot()` has already been invoked. Its sytax is:
`legend(x, y = NULL, legend, fill = NULL, col = par("col"),...,pch)`

The x and y arguments give the location of the upper left corner of the legend and may be input using the `locator(1)` function. The `legend` argument is used to give the descriptions which correspond to the characters given by the pch argument. The legend, pch and col arguments must use the same ordering.

I'll also add a text annotation to identify the setosa data.

```
> attach(iris)   #column names will be assumed to be from iris
> x.limits = c(min(Petal.Length),max(Petal.Length))
> y.limits = c(min(Petal.Width),max(Petal.Width))
> iris.setosa = subset(iris,Species=="setosa")
> plot(iris.setosa$Petal.Length,iris.setosa$Petal.Width, xlim=x.limits,
+      ylim=y.limits,col=1,pch="*", xlab="Petal Length",ylab="Petal Width",
+      main="Relationship between Petal Width and Length \n for Three Iris Species")
> iris.virginica =subset(iris,Species=="virginica")
> points(iris.virginica$Petal.Length,iris.virginica$Petal.Width,col=2,pch=2)
> iris.versicolor =subset(iris,Species=="versicolor")
> points(iris.versicolor$Petal.Length,iris.versicolor$Petal.Width,col=3,pch=3)
> legend(x=1,y=2.5,legend=c("setosa","virginica",'versicolor'),pch=1:3,col=1:3)
> #legend(locator(1),legend=c("setosa","virginica",'versicolor'),pch=1:3,col=1:3)
> #
> text(x=1.5,y=.8,labels="setosa")  #trial and error to get it right
> text(x=4,y=0.7,labels=expression(y==beta[0]+beta[1]*x))
```



Note the last text annotation included mathematical symbols. The syntax for adding a variety of math notation to your plot or legend using the `expression()` function can be found here: `http://vis.supstat.com/2013/04/mathematical-annotation-in-r/`.