Introduction to Lattice Graphs: a Powerful Tool for Displaying Multivariate Relationships

Emma Wang

October 29, 2017

Introduction

The lattice package, developed by Deepayan Sarker, is a powerful tool for displaying multivariate relationships. It is a high-level data visualization system that is sufficient for typical graphing needs, and also flexible enough to handle nonstandard requirements. Even though it is still possible to generate the same graphs with base R graphics, the lattice package makes the whole process significantly easier. Compared to ggplot2, lattice is more advantageous becasue it has a much shorter rendering time (the graphs can be generated quicker), and it has the capacity to generate 3D graphics. This post will demonstrate the usage of the most popular functions in the lattice package, with an emphasis on grouping/factoring techniques and certain visual elements.

Outline

Overview of Functions in Lattice Package

The Trellis Object

Note that many functions, such as `xyplot()` in the lattice package are high level functions. These functions are different from traditional R functions in that they do not perform any plotting themselves. Instead, they return an object of class `"trellis"`, which needs to be printed or plotted to create an actual plot. Due to R's automatic printing rule, it is usually not necessary to write the line that prints the plot. There are several scenarios where automatic printing is suppressed, but they are beyond the scope of this post.

Functions in Lattice and Their Usage

Graphing Function Names	Description	Formula Examples	Type
barchart	bar chart	`x~A` or `A~x`	Univariate
bwplot	box-and-whisker plot	`x~A` or `A~x`	Univariate
cloud	3D scatterplot	`z~x*y|A`	Trivariate
contourplot	3D contour plot	`z~x*y`	Trivariate
densityplot	kernal density plot	`~x|A*B`	Univariate
dotplot	dotplot	`~x|A`	Univariate
histogram	histogram	`~x`	Univariate
levelplot	3D level plot	`z~y*x`	Trivariate
parallel	parallel coordinates plot	data frame	Hypervariate
splom	scatterplot matrix	data frame	Hypervariate
stripplot	strip plots	`A~x` or `x~A`	Univariate
xyplot	scatterplot	`y~x|A`	Bivariate
wireframe	3D wireframe graph	`z~y*x`	Trivariate

In this post, we will focus on exploring `xyplot()`, `cloud()`, `bwplot()`, `stripplot()`, `dotplot()`, `densityplot()`, `histogram()`.

All datasets used in the demonstrations are R's built-in datasets, which include `mtcars`, `iris`, `ChickWeight` and `ToothGrowth`.

Let's begin by installing and loading the package:

Install: `install.packages("lattice")`

```
# Load
library("lattice")
```

xyplot(): Scatter plot

The R function xyplot() is used to produce bivariate scatter plots or time-series plots. In other words, it describes the relationship between two variables with points on a cooridinate system. The simplified format is as follow: `xyplot(y ~ x|A, data)` whre y is the variable on the y-axis, x is the variable on the x-axis, A is an optional conditioning variable (typically a category/factor), and data is the data frame used for plotting.

Data set: mtcars

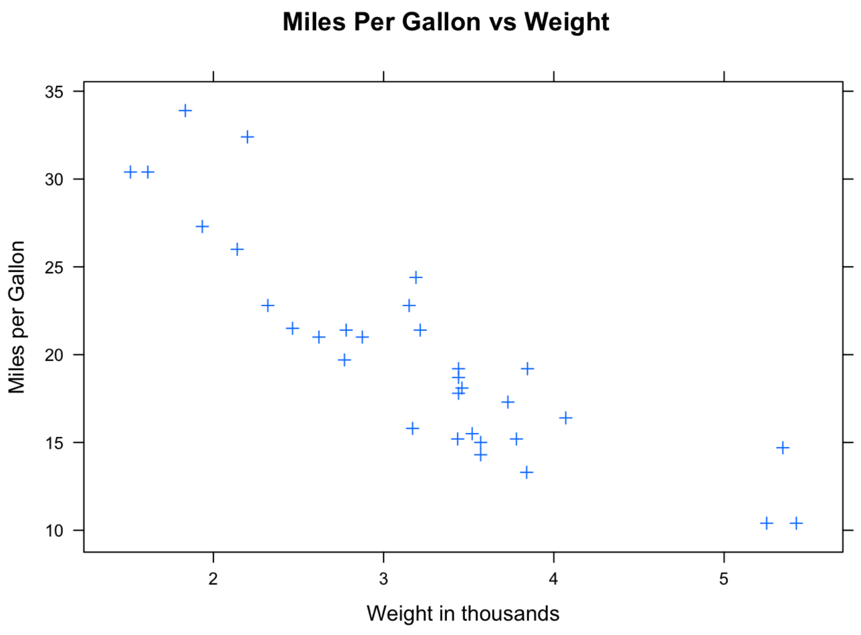
Take a look at the data:

```
mtcars[1:10, ]
```

```
##                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
```

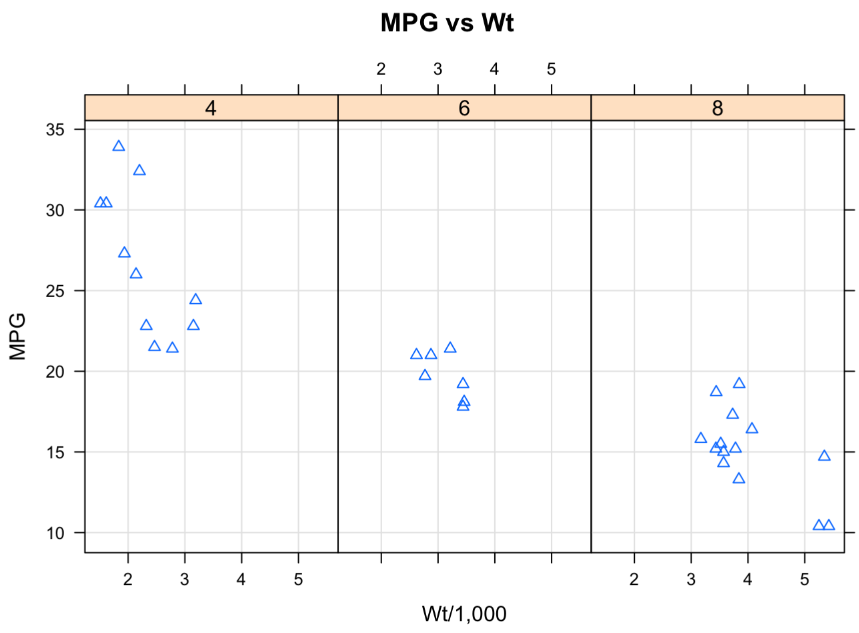
Let’s start with the simplest example:

```
## This is just a scatterplot of mpg vs. weight of the car
xyplot(mpg~wt, data = mtcars,

       ## Similar to base R graphics, you can add graph/axis titles
       main="Miles Per Gallon vs Weight",
       xlab = "Weight in thousands",
       ylab = "Miles per Gallon",

       ##pch controls the shape of the points displayed,
       ##ranges from 1 to 25, each number representing a unique shape
       pch = 3
       )
```



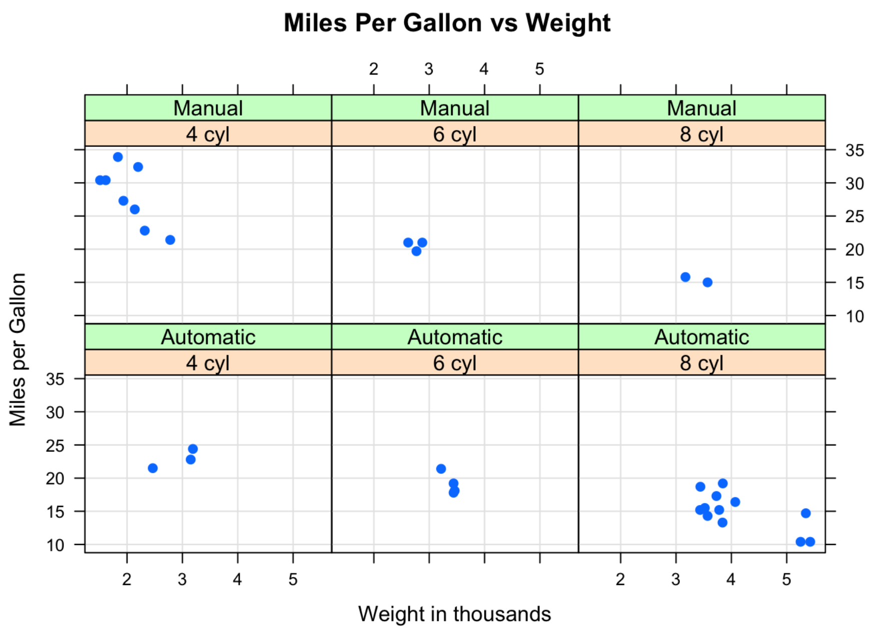
Note how the previous example does not inlude a conditioning variable. Now let’s draw mpg vs. wt for cars with each number of cylinder (conditioning variable). The resulting plot will consist of several panels, the number of which will correspond to the number of unique values assumed by the conditioning variable(s). In the case of mtcars$cyl, there will be three panels since the unique values assumed by cylinder are 4,6,8. However, we really don’t have to know the number of values in advance. We can just plot it and view the defaults.

```
xyplot(mpg~wt | factor(cyl), ##converting the number of cylinders into factors
      ## Selecting a data frame
      data=mtcars,

      ##Giving titles
      main="MPG vs Wt",
      xlab="Wt/1,000",
      ylab="MPG",

      ##Picking a shape
      pch=24,

      ##Here, I am choosing the type as "points" and "grids", meaning points and grid in the background will be di
splayed.
      type=c("p","g"))
```



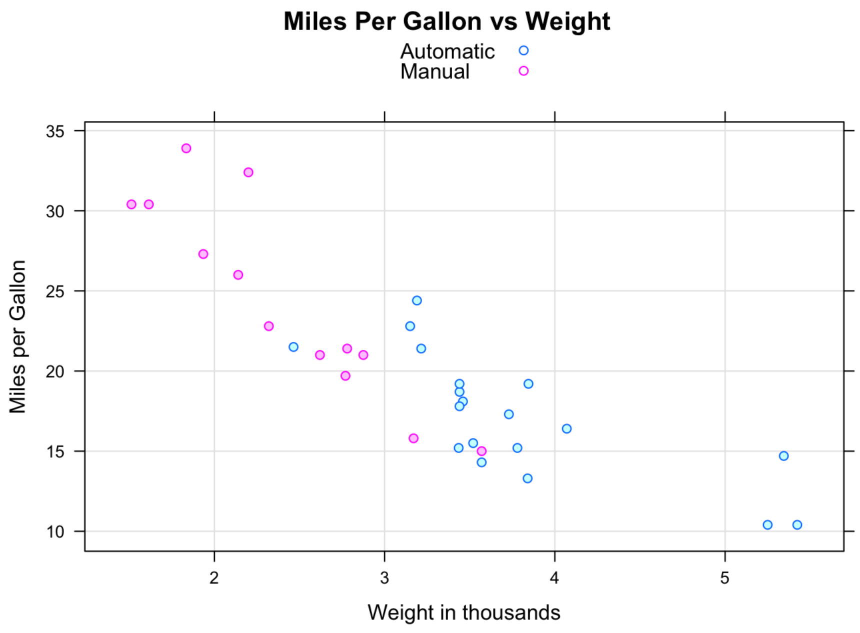
Now let’s look at a more sophisticated example with **multiple labeled** conditioning variables. The graph below shows the relationship between mpg and weight for each combination of number of cylinders and transmissions. It is evident that manual 4 cylinder cars have the highest mileage.

```
xyplot(mpg~wt | factor(cyl, labels = c("4 cyl", "6 cyl", "8 cyl")) +
      ##in the step above I labeled each level (4, 6, 8) of the factor cyl as "4 cyl", "6 cyl", "8 cyl", respectiv
ely.

      ##Labelling the levels of am in a similar fashion, and adding am (the type of transmission) as the second co
nditioning variable
      factor(am,labels = c("Automatic","Manual")),

      data=mtcars,
      main="Miles Per Gallon vs Weight",
       xlab = "Weight in thousands",
       ylab = "Miles per Gallon",

      #Picking a shape and elements to be displayed
      pch=19, type=c("p","g"))
```



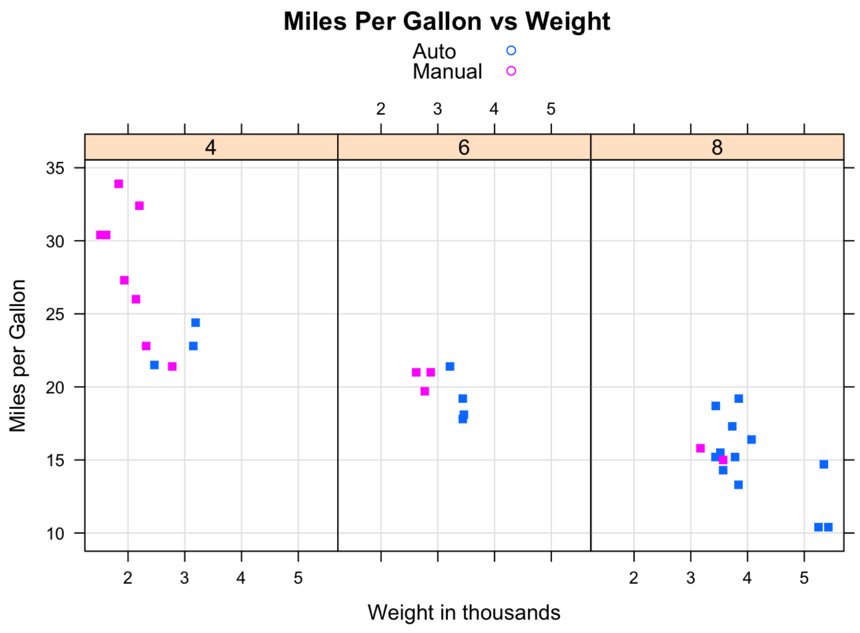
What if I want to see the relationship between weight, mpg, and transmission all in one panel? This is easy to accomplish with lattice by using the “groups” argument. The graph below clearly shows that the lightest manual cars have the highest mileage.

```
xyplot(mpg~wt, data=mtcars,

       ##Coloring points according to the transmission group they occupy
       groups=factor(am,labels=c("Automatic","Manual")),

       #Picking a shape and elements to be displayed
       pch=21, type=c("p","g"),
       main="Miles Per Gallon vs Weight",
       xlab = "Weight in thousands",
       ylab = "Miles per Gallon",

       ##auto.key controls how we format the legends. Here I am simply telling R to display the legend.
       auto.key=TRUE)
```



Combining all we have learned….I will plot MPG vs Wt as conditioned by Cylinders while grouping by Transmission.

```
xyplot(mpg ~ wt | factor(cyl), data=mtcars,
      pch=15,

      ##The layout factor controls how the panels are displayed. Here I am saying that there should be 1 row and 3 columns of panels
      layout=c(3,1),

      main="Miles Per Gallon vs Weight",
       xlab = "Weight in thousands",
       ylab = "Miles per Gallon",
      groups = factor(am,labels=c("Auto","Manual")),
      type=c("p","g"),
      auto.key = TRUE)
```



Other examples with data set: iris

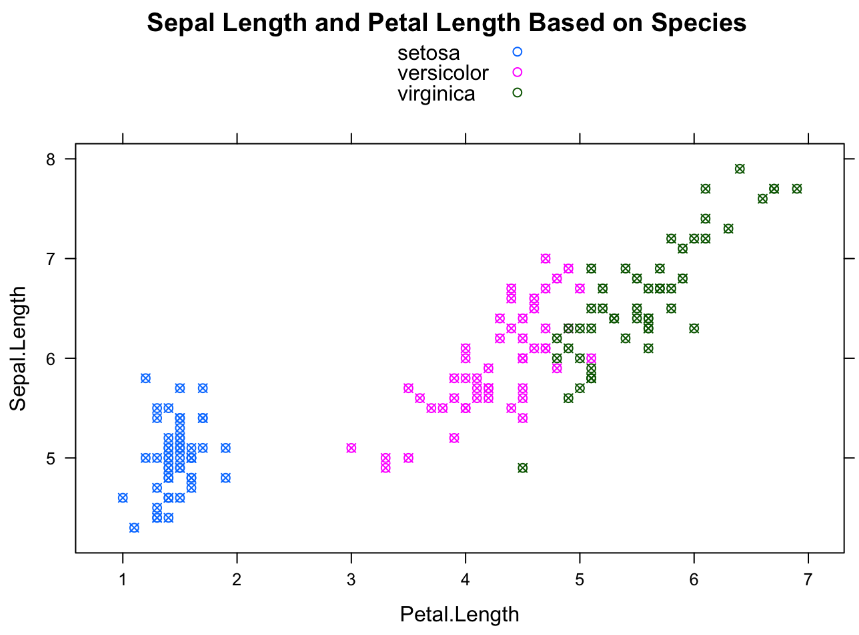
Take a look at the data:

```
iris[1:10, ]
```

```
##    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
## 4           4.6         3.1          1.5         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 7           4.6         3.4          1.4         0.3  setosa
## 8           5.0         3.4          1.5         0.2  setosa
## 9           4.4         2.9          1.4         0.2  setosa
## 10          4.9         3.1          1.5         0.1  setosa
```

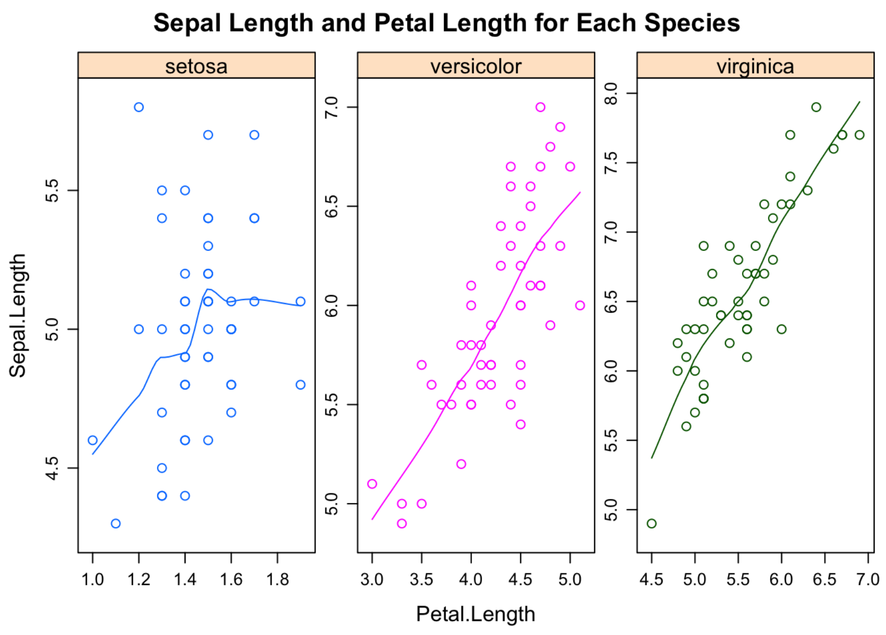
The graph below displays the relationship between Sepal.Length and Petal.Length for each species through different colors on the same panel. This shows that Virginica on average has the biggest sepal length and petal length among the three species, while Setosa on average has the smallest sepal length and petal length among the three.

```
##Coloring by groups of species
xyplot(Sepal.Length ~ Petal.Length,
       group = Species,
       data = iris,
       pch = 13,
       main = "Sepal Length and Petal Length Based on Species",
       auto.key = TRUE)
```



The graph below displays the relationship between Sepal.Length and Petal.Length for each species in different panels. In addition to the insights from the previous graph, this also shows that the correlation between sepal length and petal length for Setosa is the smallest among the three species.

```
xyplot(Sepal.Length ~ Petal.Length | Species,
       group = Species,
       data = iris,
       main = "Sepal Length and Petal Length for Each Species",
       ##Displaying the points and smoothing line without the grid
       type = c("p", "smooth"),

       ##This ensures that each panel customize the scales as appropriate. Note how the scales of each panel is di
fferent.
       scales = "free")
```



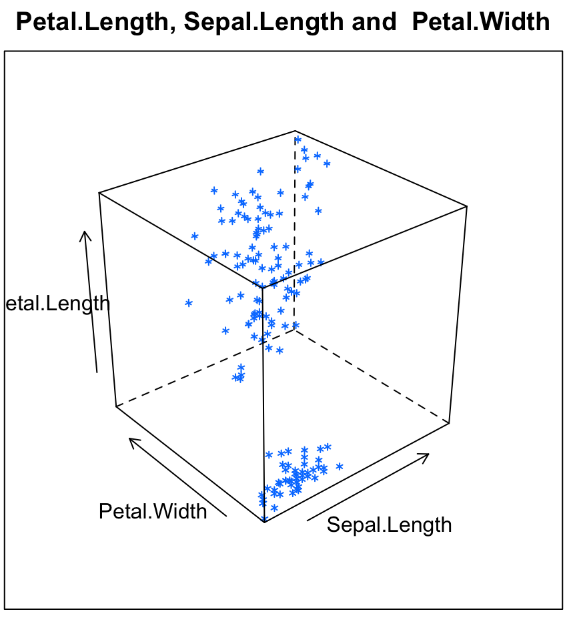
Cloud(): 3D Scatter Plot

The 3D scatterplot describes the relationship between 3 variables, so it's designed for trivariate relationships. Many logic and lines of reasoning from xyplot() carry over easily.

Data set: iris

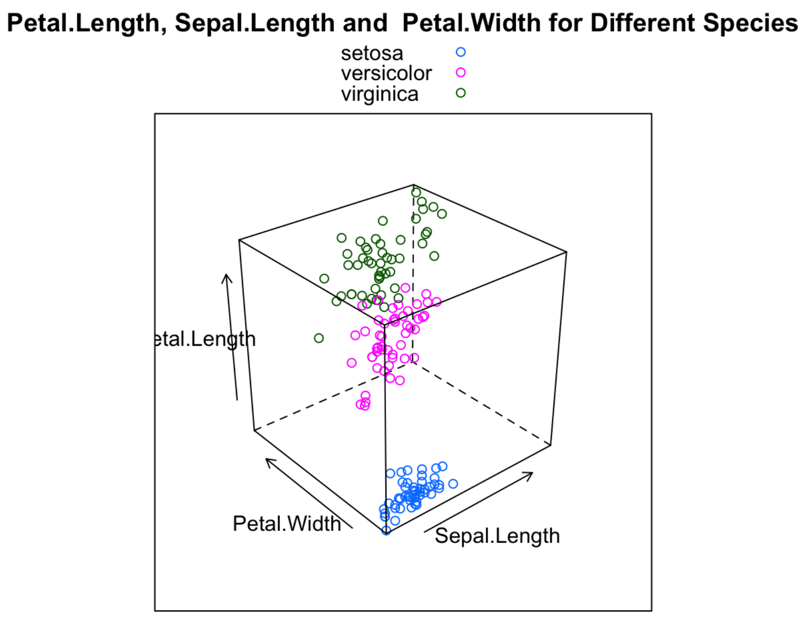
Let's start with a simple 3D scatter plot that describes the relationship between Sepal.Length, Sepal.width, and Petal.length. Keep in mind that the general formula is z ~ x*y.

```
# Basic 3D scatter plot
cloud(Petal.Length ~ Sepal.Length * Petal.Width,
       data = iris,
      main = "Petal.Length, Sepal.Length and  Petal.Width")
```



If we want more revealing infomation about how these three variables differ for different species, we can use the `group` argument.

```
## 3D Scatter plot grouped by species
cloud(Petal.Length ~ Sepal.Length * Petal.Width,
      group = Species,
      data = iris,
      auto.key=TRUE,
      main = "Petal.Length, Sepal.Length and  Petal.Width for Different Species")
```



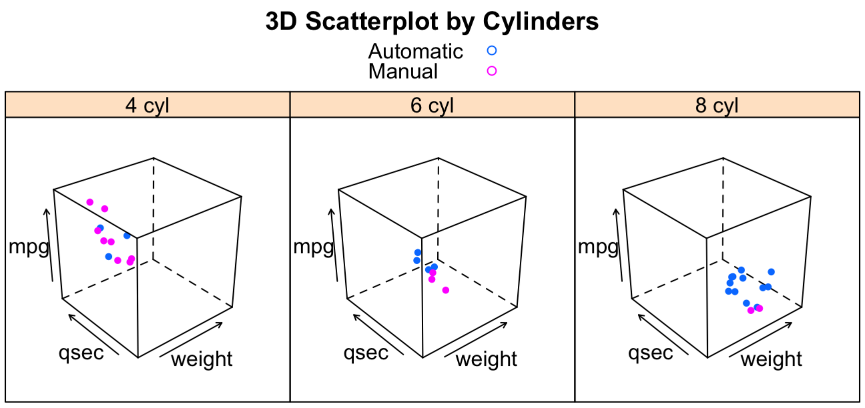
As above, it is evident that species of

Virginica typically has the largest Petal.Length, Petal.Width, and Sepal.Length.

Data set: mtcars

Finally, we can now look at how we could add a conditioning variable, and combining everything we have learned before. The graph below describes the relationship between mpg, weight, and 1/4 mile time for cars of each number of cylinder. It is shown that cars with fewer cylinders typically have lower weights, higher mileage, and lower 1/4 mile time. This makes sense because less powerful cars save fuel, weight less, but accelerate slower. On top of this, manual cars have higher mileage on average.

```
cloud(mpg~wt*qsec|factor(cyl, labels = c("4 cyl", "6 cyl", "8 cyl")),
      ## Splitted into different panels based on the number of cylinders.

      data = mtcars,

      ## Coloring according to transmission
      groups = factor(am, label = c("Automatic", "Manual")),

      ##Selecting shape of points
     pch = 20,

     ## Giving titles
      main="3D Scatterplot by Cylinders",
     xlab = "weight",
     ylab = "qsec",
     zlab = "mpg",

     ##Choosing a layout of 1 row and 3 columns
     layout = c(3,1),

     ##Asking R to display the lengends
     auto.key = TRUE)
```



Box-and-whisker plot, Dot plot, Strip plot

Box plot, dot plot, and strip plot all describe univariate relationships. The underlying logic for them is pretty similar, with one variable `x` and one conditioning variable `A` (a factor). These plots are best describing the distribution of `x` over different levels of `A`.

Data set: Tooth Growth

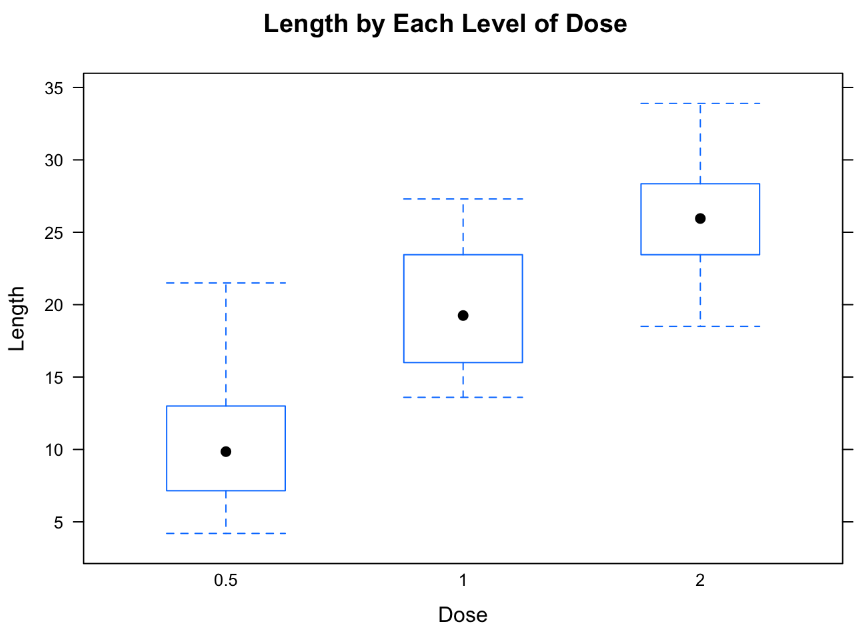
Take a look at the data:

```
head(ToothGrowth)
```

```
##    len supp dose
## 1  4.2   VC  0.5
## 2 11.5   VC  0.5
## 3  7.3   VC  0.5
## 4  5.8   VC  0.5
## 5  6.4   VC  0.5
## 6 10.0   VC  0.5
```

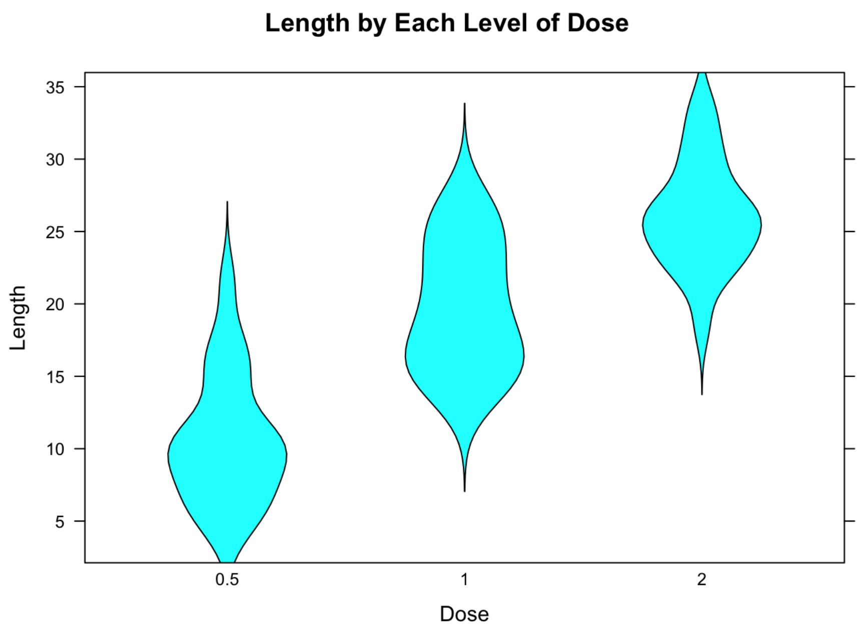
The graph below shows how the distribution of length vary over each level of dose. As shown, there is a positive correlation between the median of the length and the dose level, while the spread of length remains similar for all does levels.

```
## Converting dose to factors so that I could find the spread of Length over each level of dose.
ToothGrowth$dose = as.factor(ToothGrowth$dose)

# Basic box plot
bwplot(len ~ dose,  data = ToothGrowth,
       main = "Length by Each Level of Dose",
       xlab = "Dose", ylab = "Length")
```



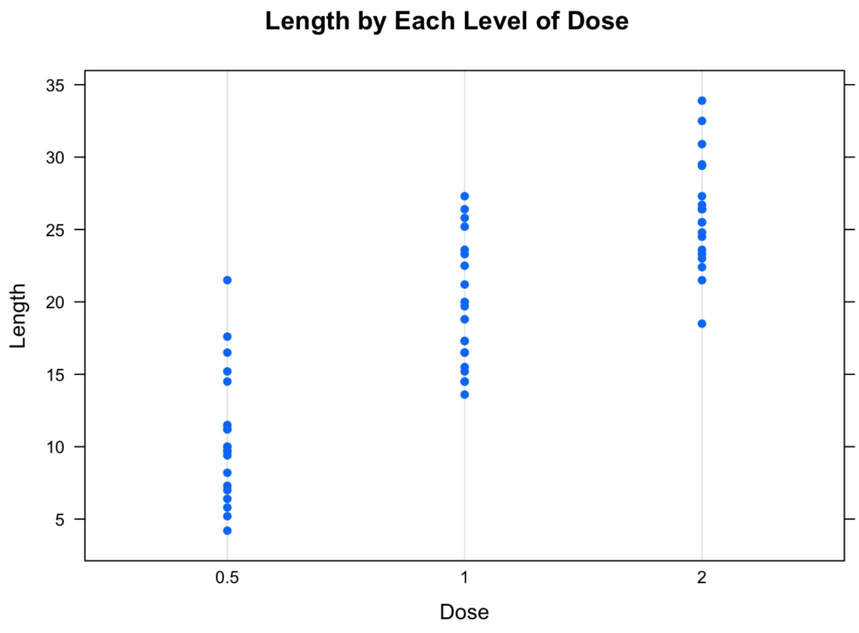
For the same box plot as above, I can change the method of visualization by varying the `panel` factor. You can interpret the violin plot as each violin is the mirrored image of the density plot of length over each level of dose.

```
# Violin plot using panel = panel.violin
bwplot(len ~ dose,  data = ToothGrowth,
       panel = panel.violin,
       main = "Length by Each Level of Dose",
       xlab = "Dose", ylab = "Length")
```



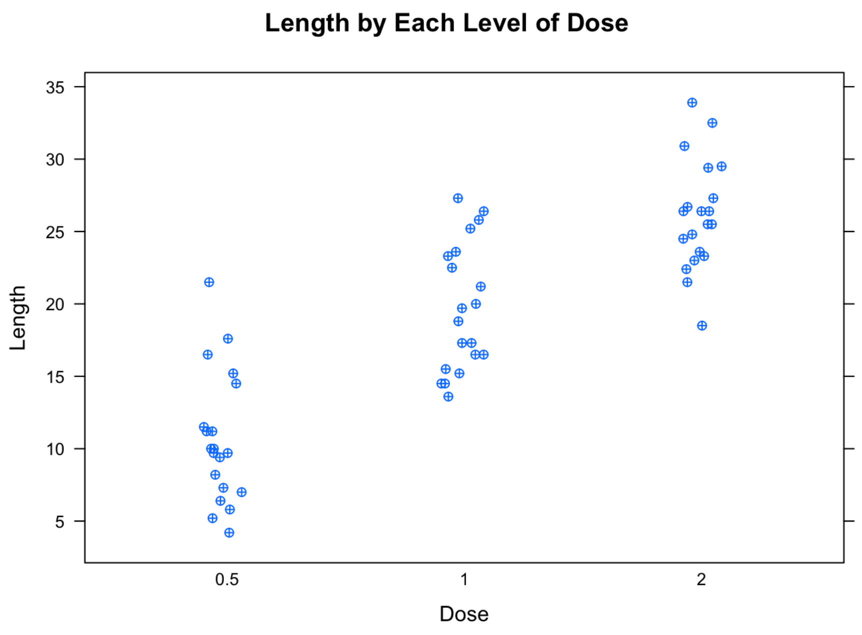
Again, the graph below is a dot plot that shows the distribution of length for each level of dose. This time the spread is represented through a series of dots on a line.

```
# Basic dot plot
dotplot(len ~ dose,  data = ToothGrowth,
        main = "Length by Each Level of Dose",
        xlab = "Dose", ylab = "Length")
```



The graph below is known as a strip plot of a 1-D scatter plot. It is used to describe the same relationship as the graphs above. Note the use of jittering and partial transparency to alleviate potential overplotting.

```
# Basic strip plot
stripplot(len ~ dose,  data = ToothGrowth,

          ##Allowing for jittering and partial-transparency
          jitter.data = TRUE, pch = 10,

          main = "Length by Each Level of Dose",
          xlab = "Dose", ylab = "Length")
```



Data set: ChickWeight

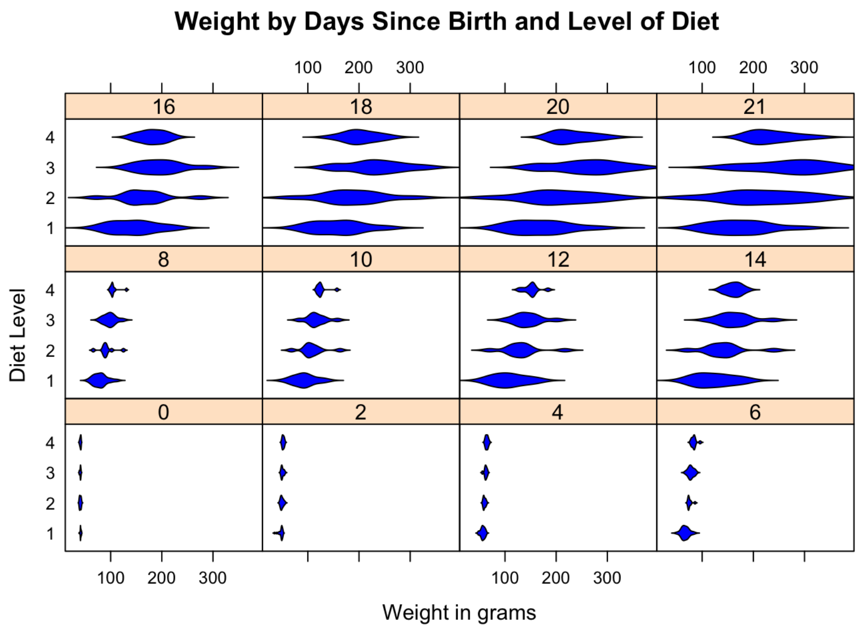
Take a look at the data:

```
head(ChickWeight)
```

```
##   weight Time Chick Diet
## 1     42    0     1    1
## 2     51    2     1    1
## 3     59    4     1    1
## 4     64    6     1    1
## 5     76    8     1    1
## 6     93   10     1    1
```

Now we have learned the basics of each of the three plots and how they are similar. Let’s add another conditioning variable to combine what we learned earlier. The boxplot below describes the distribution of chicken’ weight for each different combination of number of days since birth and diet level. Different from before is that by adding `"|"`, the box plots are split into different panels according to the days since birth instead of being drawn on the same panel with different levels. It shows that the median and spread of the weight both increase as the days since birth increases; the median of the weight increases but the spread decreases as the diet level increases.

```
bwplot(Diet~weight|factor(Time),
       data=ChickWeight,

       ##Diaplying this as a violin plot
       panel = panel.violin,

       col="blue",
       main="Weight by Days Since Birth and Level of Diet",
       xlab="Weight in grams",
       ylab = "Diet Level")
```



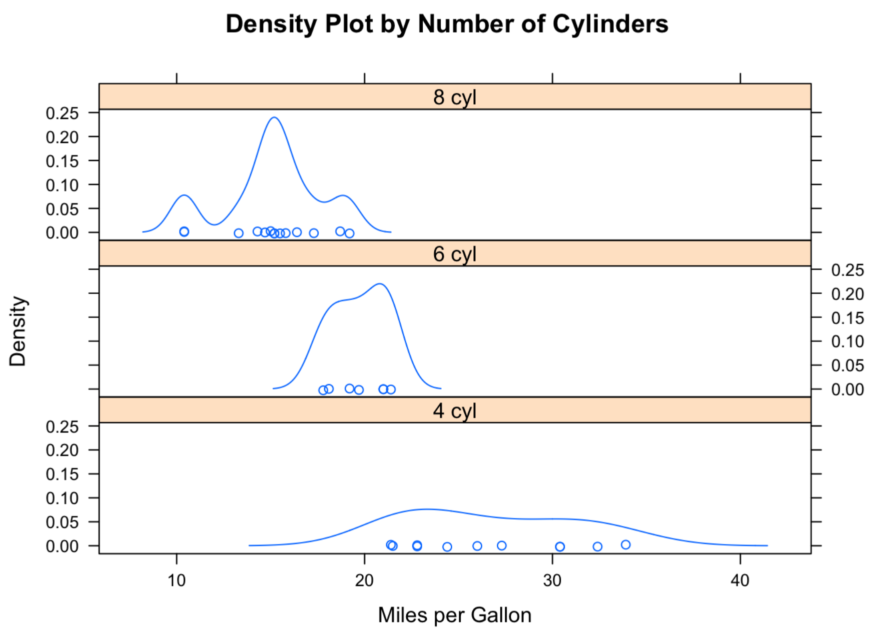
Density Plot and Histogram

Density plots and histogram are both used to describe univariate relationships. The general formula for density plot is `~x|A*B`. And the general formula for histogram is `~x`. Given everything we have done so far, this section should be fairly easy to understand.

Density Plot with Conditioning

The graph below describes the density of MPG for cars with each number of cylinders. As you can see, the center of mpg seems to decrease as the number of cylinders increase because more powerful cars are less fuel efficient.

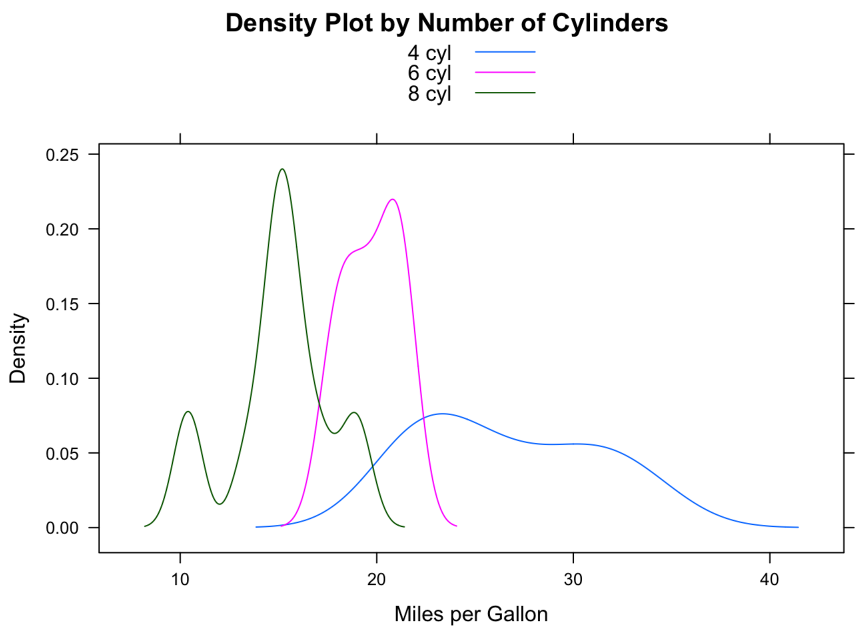
```
densityplot(~mpg|factor(cyl, labels = c("4 cyl", "6 cyl", "8 cyl" )),
    data = mtcars,
    main="Density Plot by Number of Cylinders",
   xlab="Miles per Gallon",
   layout=c(1,3))
```



Density plot with Multiple Groups

The graph below describes, in one panel, how the density of mpg varies with each level of cylinders. The center of the mpg shifts right as the numebr of cylinders decrease because less powerful cars are more fuel efficient.

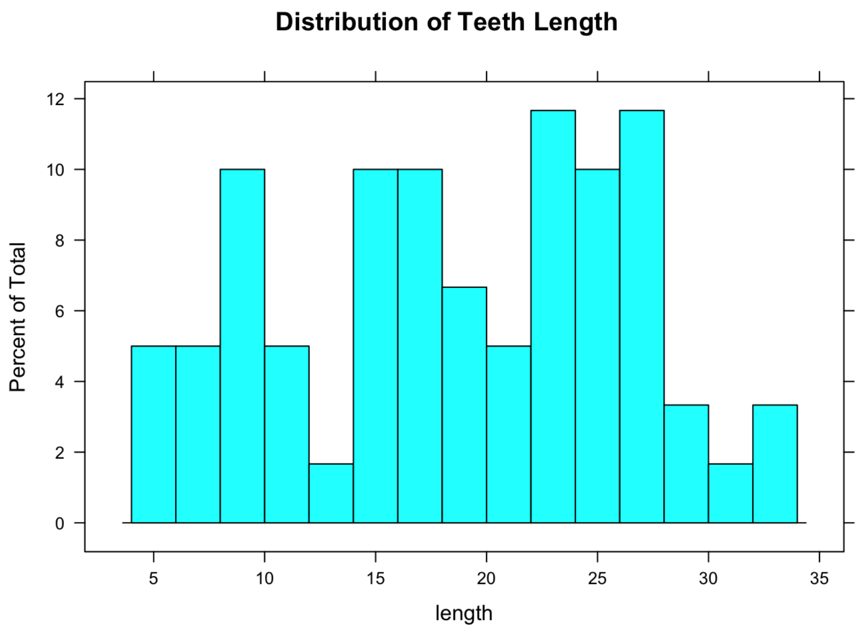
```
densityplot(~mpg,
            groups = factor(cyl,labels = c("4 cyl", "6 cyl", "8 cyl" )),
    data = mtcars,
    main="Density Plot by Number of Cylinders",
   xlab="Miles per Gallon",

   ## Plot density curves without the points underneath.
  plot.points = FALSE,

  ## Diplaying legends
   auto.key = TRUE)
```



Histogram of Tooth length

This is probably the easiest graphs of everything we have learned so far. The graph below shows that teeth lengths between 22 and 27 have the most frequent occurences.

```
histogram(~ len, data = ToothGrowth,
          main = "Distribution of Teeth Length",
          xlab = "length",

          ## this argument controls the bin size. The greater the number of breaks, the smaller the bin size.
          breaks = 20)
```



Summary Message

We have leanred a lot in this post. Lattice is a very useful tool for displaying multivariate relationships, with a short rendering time and ample flexibility. There are many functions, but as a general rule, `"|"` will divide the graph into multiple panels based on a conditional variable, while `"~"` and `"*"` add variables (x, y, z); Important arguments that impact the visual representations are `pch`, `auto.key` and `layout`.

Thank you for reading!

References

1. https://www.r-bloggers.com/conditioning-and-grouping-with-lattice-graphics/
2. https://www.statmethods.net/advgraphs/trellis.html
3. http://polisci.msu.edu/jacoby/icpsr/graphics/lattice/Lattice,%20ICPSR%202016%20Outline,%20Ver%201.pdf
4. https://cran.r-project.org/web/packages/lattice/lattice.pdf
5. http://www.sthda.com/english/wiki/lattice-graphs
6. https://stat.ethz.ch/R-manual/R-devel/library/lattice/html/Lattice.html
7. https://www.stat.ubc.ca/~jenny/STAT545A/block09_xyplotLattice.html