# Graphics: Base: Barplot

Last week we left off with a discussion on colors and how they can improve a chart or plot.

We found that we could use built-in palettes from which we could pick "pre-made" colors. This makes it easy for those of us who can't really pick matching color schemes so easily.

This week we extend this idea some and learn about "color ramps", annotation, as well as some other chart types.
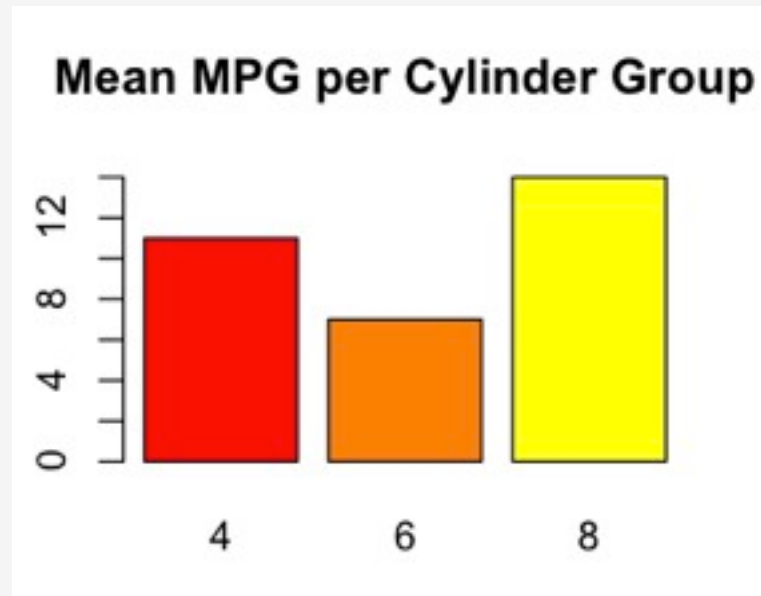
We also explore lattice graphics.

Tuesday, October 29, 13

# Graphics: Base: Barplot

```
myt = tapply(mtcars$mpg, mtcars$cyl, mean)

# tapply produces a table

        4        6        8
26.66364 19.74286 15.10000

barplot(myt, main = "Mean MPG per Cylinder Group",
col=heat.colors(3))
```

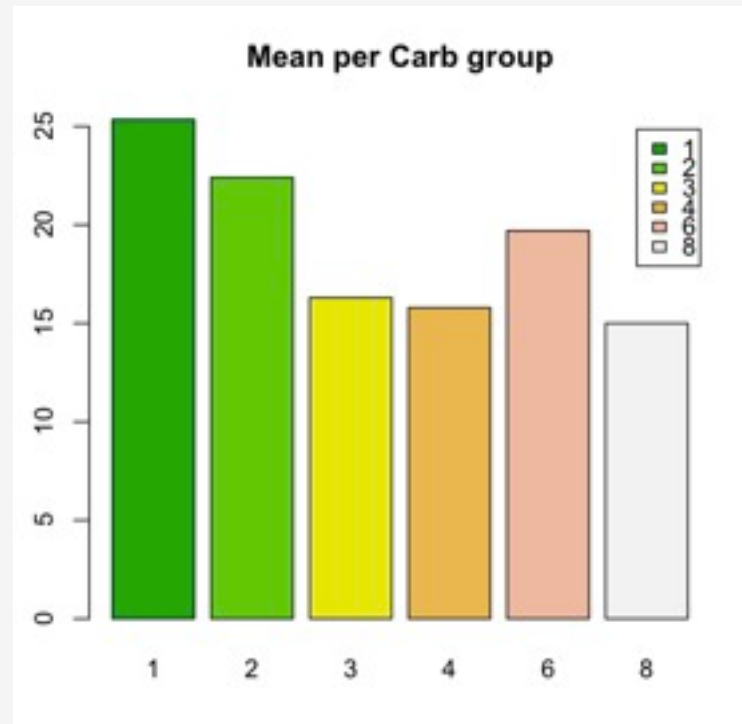Tuesday, October 29, 13

# Graphics: Base: Barplot

```
myt = tapply(mtcars$mpg, mtcars$carb, mean)

        1        2        3        4        6        8
25.34286 22.40000 16.30000 15.79000 19.70000 15.00000

barplot(myt, main = "Mean MPG per Carb group",
             col= terrain.colors(length(myt)),legend=T)
```
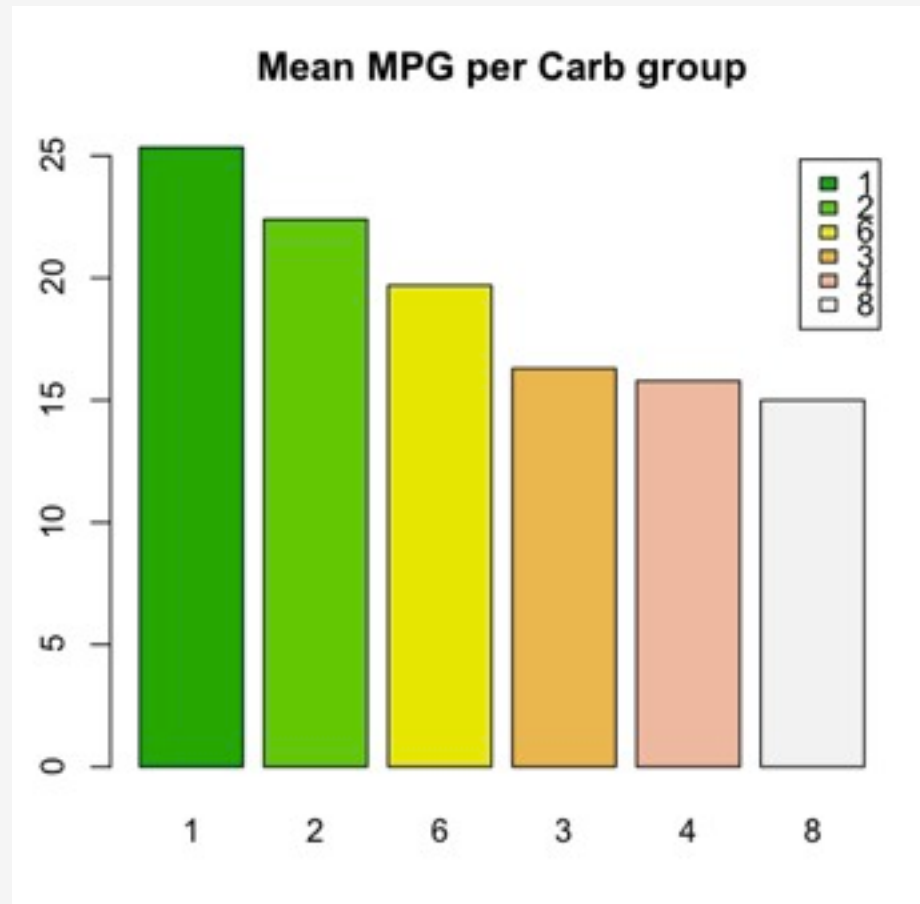
Tuesday, October 29, 13

# Graphics: Base: Barplot

```
barplot(rev(sort(myt)), main = "Mean MPG per Carb group",
              col= terrain.colors(length(myt)),legend=T)
```

Tuesday, October 29, 13

# Graphics: Base: Barplot

```
barplot(rev(sort(myt)), main = "Mean MPG per Carb group",
          col= terrain.colors(length(myt)),legend=T)



sort(myt)
        8        4        3        6        2        1
15.00000 15.79000 16.30000 19.70000 22.40000 25.34286

rev(sort(myt))
        1        2        6        3        4        8
25.34286 22.40000 19.70000 16.30000 15.79000 15.00000


#  I reverse the sort to make it so the legend doesn't overwrite one of the bars. Try
#  plotting it without doing the rev to see what I mean.

barplot(sort(myt), main = "Mean MPG per Carb group",
          col= terrain.colors(length(myt)),legend=T)
```

Tuesday, October 29, 13

# Graphics: Base: Barplot

What about making the colors reflect the mean ? If we have an increasing sequence of bars why not make the color for each group a different shade from a graduated color scale.

To do this we need a "color ramp" command to generate a palette. Its easier to understand this with an example.
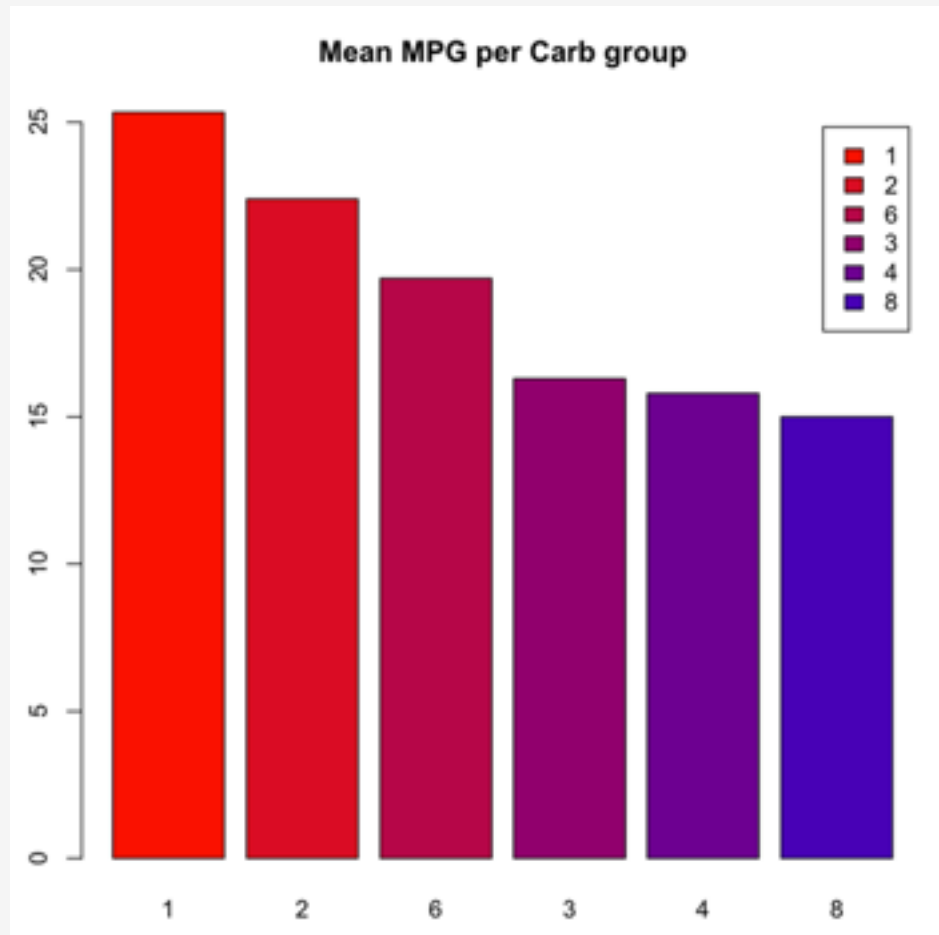
```
mycols = colorRampPalette(c("red","blue"))( 6 )

[1] "#FF0000" "#DA0024" "#B60048" "#91006D" "#6D0091" "#4800B6"
```

This creates a graduated color scheme between red and blue. Let's apply this to our barplot.

Tuesday, October 29, 13

# Graphics: Base: Barplot

```
barplot(rev(sort(myt)),
         main = "Mean MPG per Carb group", col=mycols, legend=T)
```

Tuesday, October 29, 13
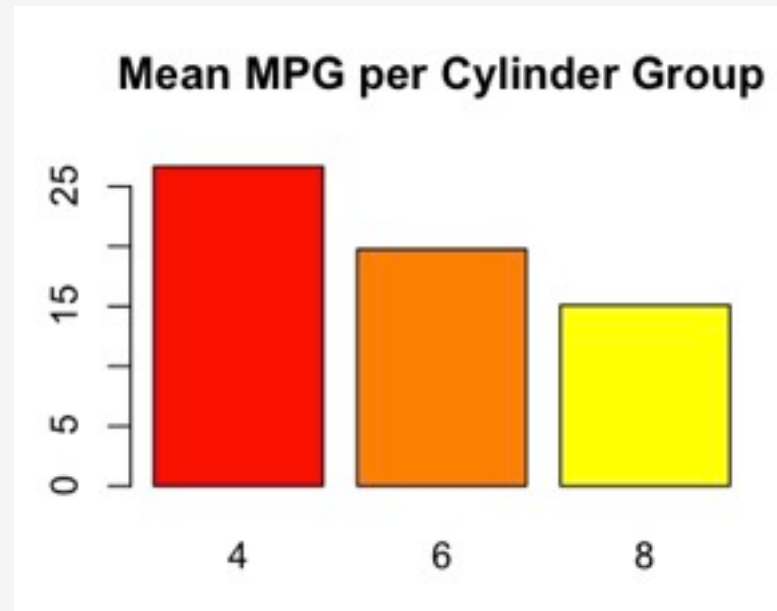
# Graphics: Base: Barplot

```
myt = tapply(mtcars$mpg, mtcars$cyl, mean)

# tapply produces a table

        4        6        8
26.66364 19.74286 15.10000

barplot(rev(sort(myt)), main = "Mean MPG per Cylinder Group",
col=heat.colors(3))
```



Mean MPG per Cylinder Group

Tuesday, October 29, 13

# Graphics: Base: Barplot

```
myt = tapply(mtcars$mpg, mtcars$cyl, mean)

# tapply produces a table

        4         6         8
26.66364 19.74286 15.10000

temp = barplot(myt, main = "Mean MPG per Cylinder Group",
                    col=heat.colors(3))

temp          # These represent the X coordinates for each bar
     [,1]
[1,]  0.7
[2,]  1.9
[3,]  3.1

#      x     y     text

text(0.7, 10, 26.66)     # We put up the means one by one

text(1.9, 10, 19.74)

text(3.1, 10, 15.10)
```
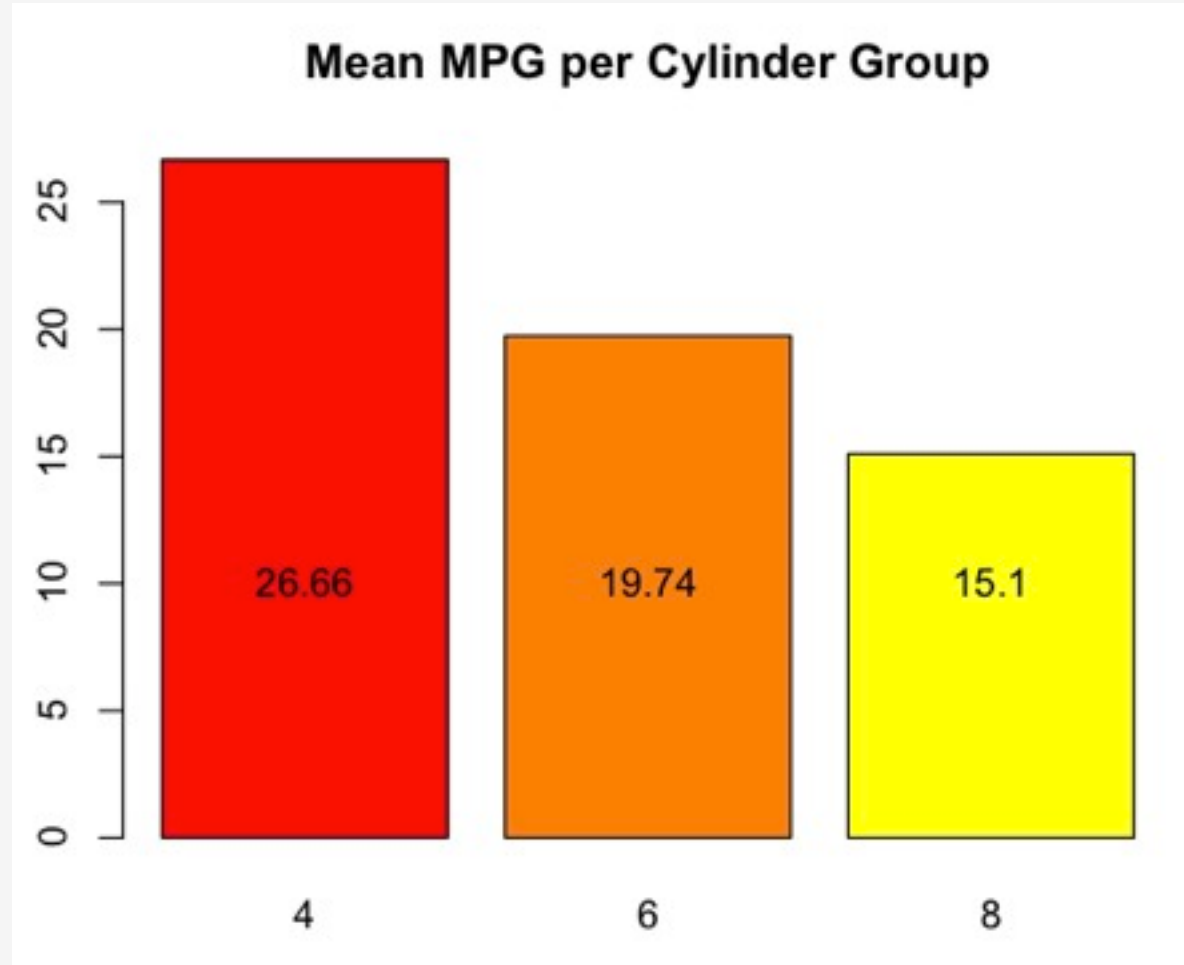
Tuesday, October 29, 13

# Graphics: Base: Barplot



Mean MPG per Cylinder Group

Tuesday, October 29, 13

# Graphics: Base: Barplot

But there is an easier way. The text function arguments can accept vectors. So an easier way to do this:

```
#      x      y     text

text(0.7, 10, 26.66)     # We put up the means one by one

text(1.9, 10, 19.74)

text(3.1, 10, 15.10)

# is:

text(temp,10,round(myt,2))

temp
      [,1]
[1,]  0.7
[2,]  1.9
[3,]  3.1
```

Tuesday, October 29, 13

# Graphics: Base: boxplots

If you are doing a X/Y scatterplot and one of them is a categorical variable then the resulting plot will be a boxplot.
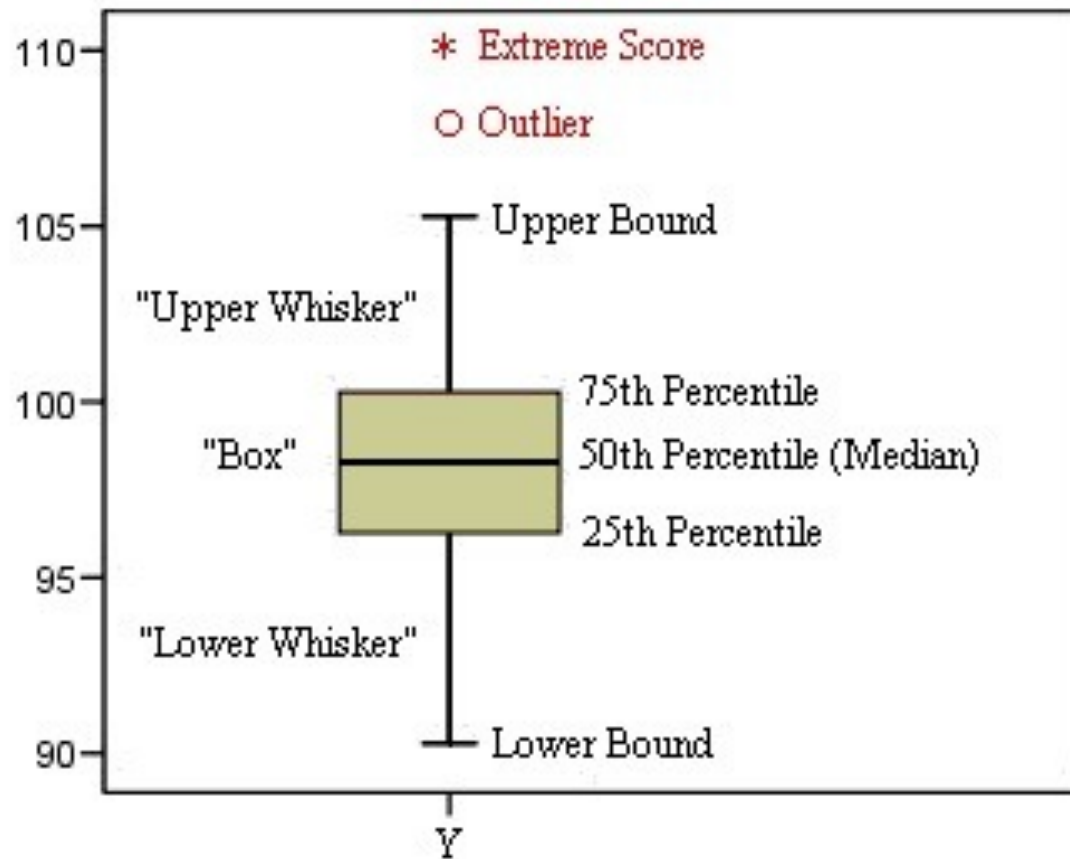
A box plot summarizes a lot of information clearly. It shows the location and spread of data as well as skewness.

The horizontal line shows the median. The bottom and top lines show the 25th and 75th percentiles respectively.

The vertical dashed lines are called "whiskers". The show the maximum of 1) the smaller of the data being plotted or 1.5 times the interquartile range.
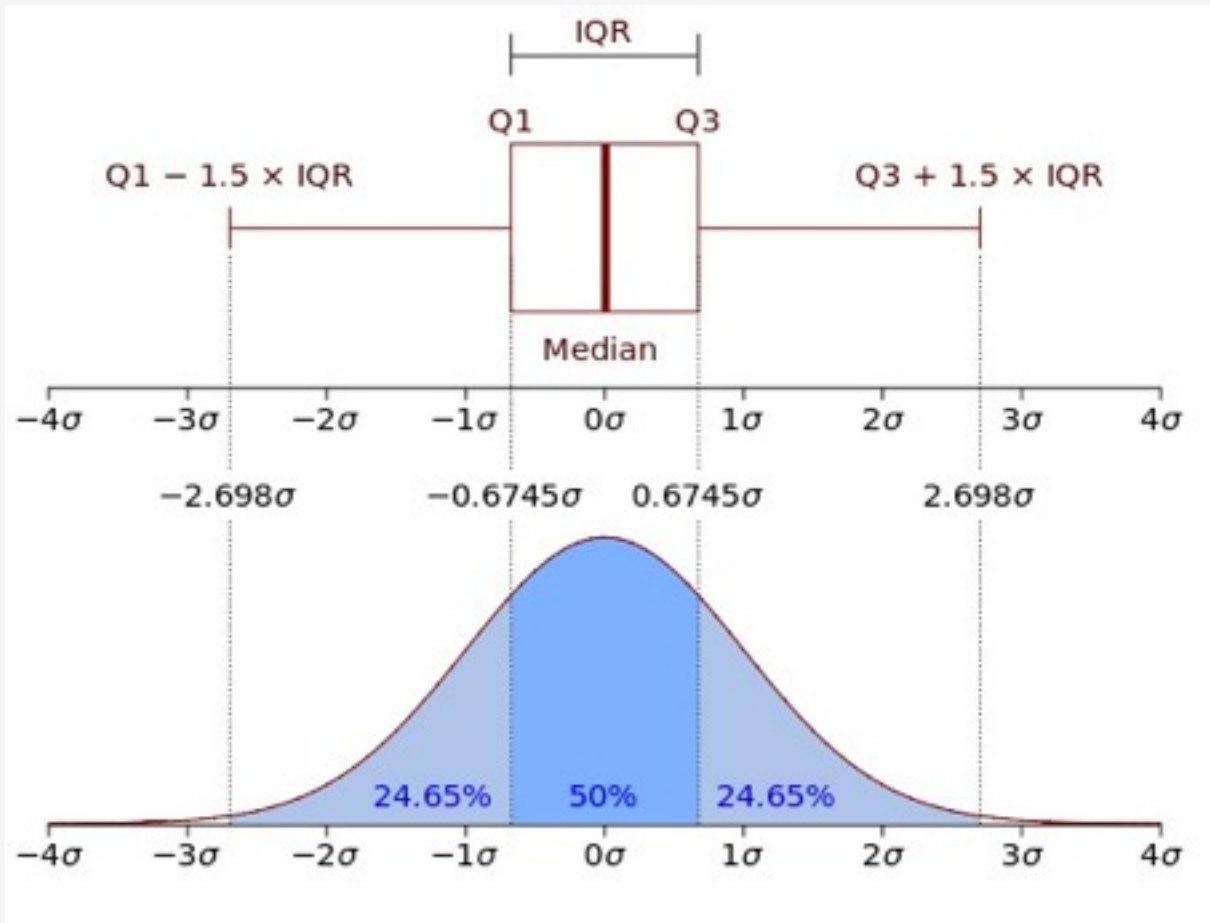
1.5 times the interquartile is roughly two standard deviations. And the IQR is the difference in the response variable between the first and third quartile.

Tuesday, October 29, 13

# Graphics: Base: boxplots



http://psystats.wikispaces.com/Boxplots

Tuesday, October 29, 13

# Graphics: Base: boxplots



http://en.wikipedia.org/w/index.php?title=File:Boxplot_vs_PDF.svg

Tuesday, October 29, 13

# Graphics: Base: boxplots

Note that any boxplot output will also match the output from the "fivenum" command:

```
fivenum(mtcars$mpg)
[1] 10.40 15.35 19.20 22.80 33.90

boxplot(mtcars$mpg, main="MPG Boxplot",xlab="MPG",col="aquamarine")
```

Tuesday, October 29, 13
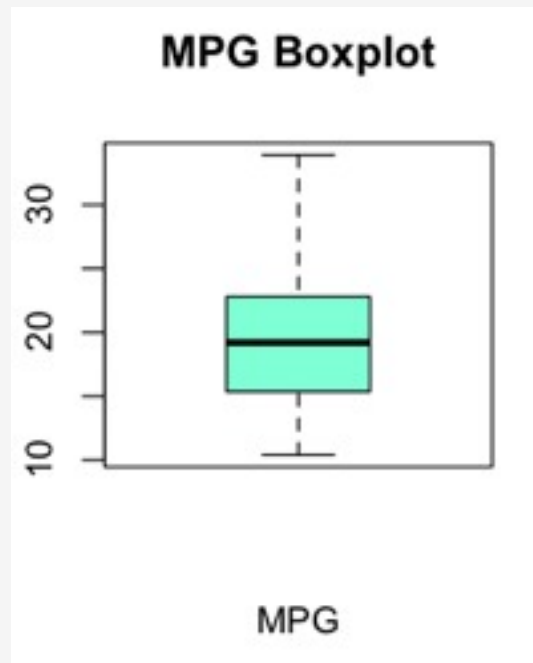
# Graphics: Base: boxplots

We can get more than one boxplot within a plot window. If you are doing a X/Y scatterplot and one of them is a categorical variable then the resulting plot will be a boxplot. Let's look at the mtcars dataset now.

```
boxplot(mpg~am, data = mtcars, main="MPG / Transmission",
                col=heat.colors(2), names=c("auto","manual"))

legend(0.5,33,c("Auto","Manual"),fill=heat.colors(2))
```
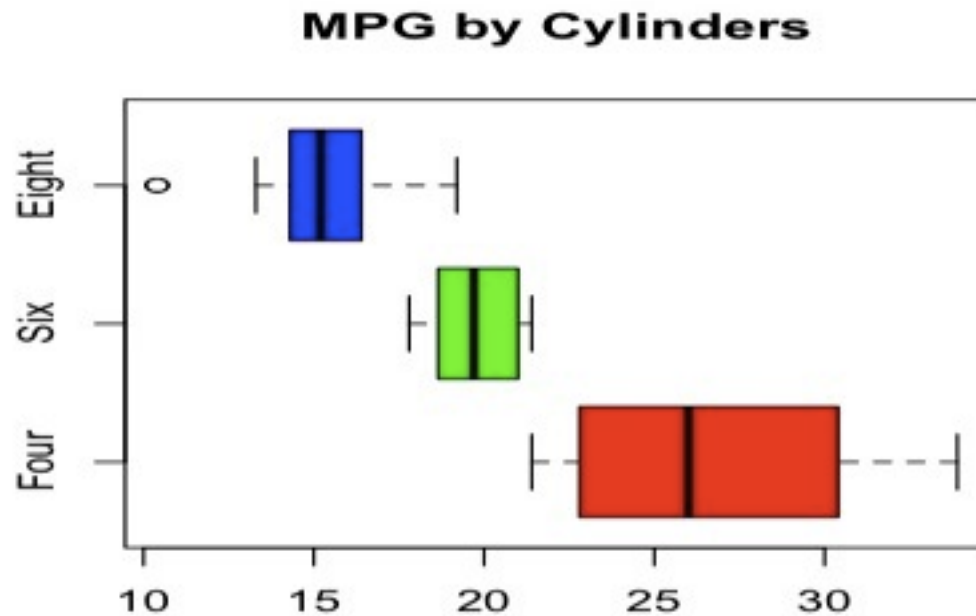
Tuesday, October 29, 13

# Graphics: Base: boxplots



**MPG / Transmission**

Tuesday, October 29, 13
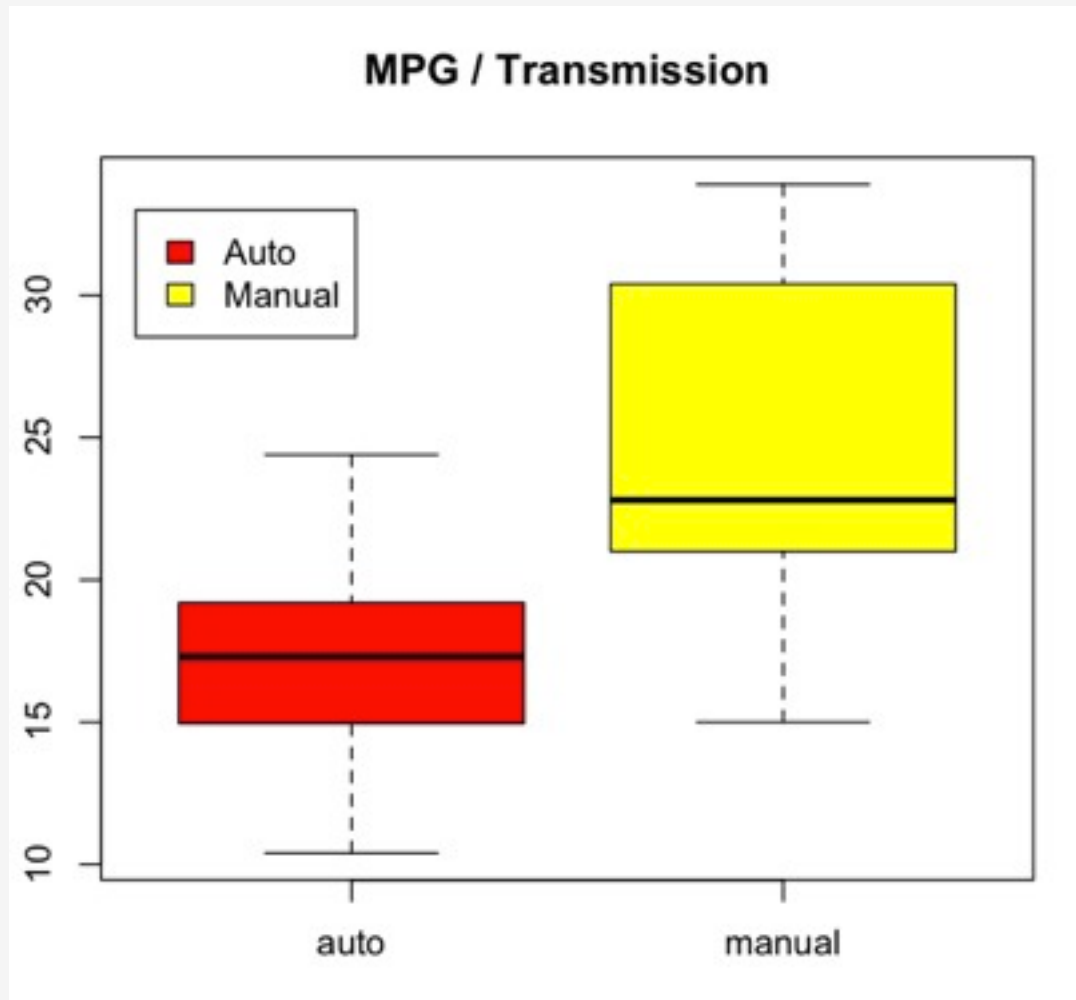
# Graphics: Base: boxplots

```
boxplot(mpg~cyl,data=mtcars,main="MPG by Cylinders",
        col=rainbow(3), horizontal=TRUE,
        names=c("Four","Six","Eight"))
```



**MPG by Cylinders**

Tuesday, October 29, 13

# Graphics: Base: boxplots

Boxplots are generally good at showing distribution of data around the median but not so good at showing the significance of differences between medians. Tukey introduced the idea of "notched" plots to address this problem.

Boxes for which the notches do not overlap are "likely" to have significantly different medians in terms of testing.

The width of the notches is proportional to the interquartile range of the sample and inversely proportional to the square root of the size of the sample.

`http://analyse-it.com/docs/220/standard/summary_paired.htm`

Tuesday, October 29, 13

# Graphics: Base: boxplots

- **Notched** box plots show a basic box plot as above, with the addition of a notched (pinched or indented) section for the confidence interval around the median (see below).
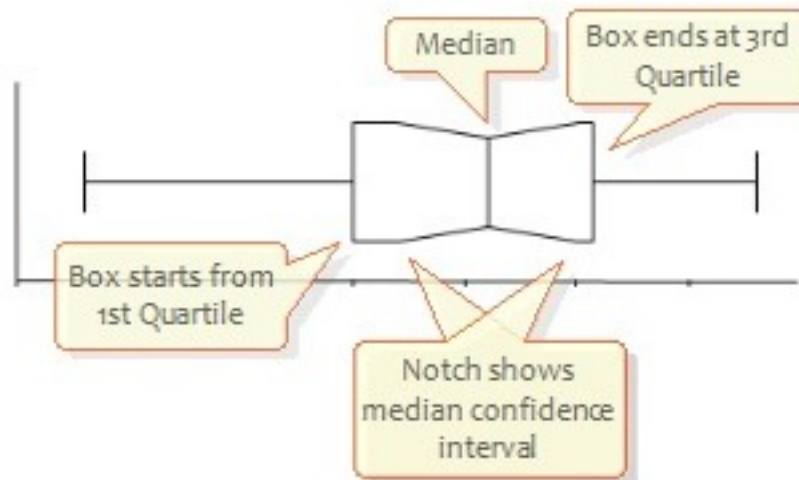


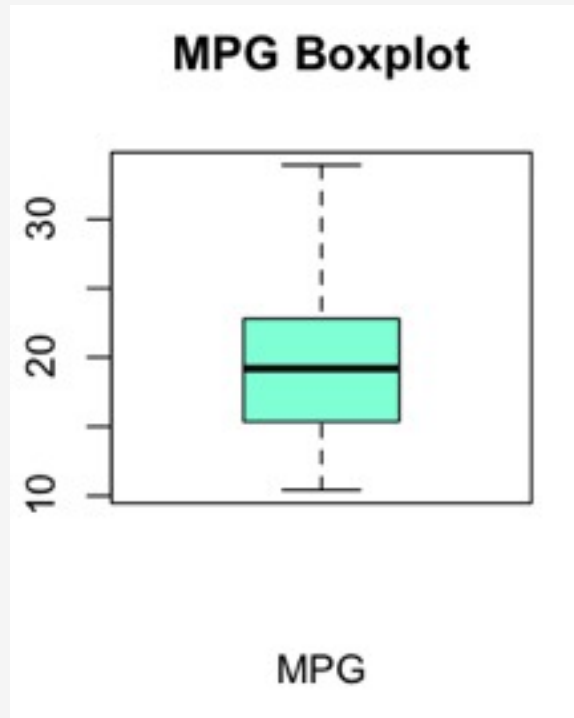http://analyse-it.com/docs/220/standard/summary_paired.htm

Tuesday, October 29, 13

# Graphics: Base: boxplots

```
boxplot(mpg~cyl, main="MPG vs. Cylinder", data=mtcars,
        notch=TRUE, col=terrain.colors(3))
```

Tuesday, October 29, 13
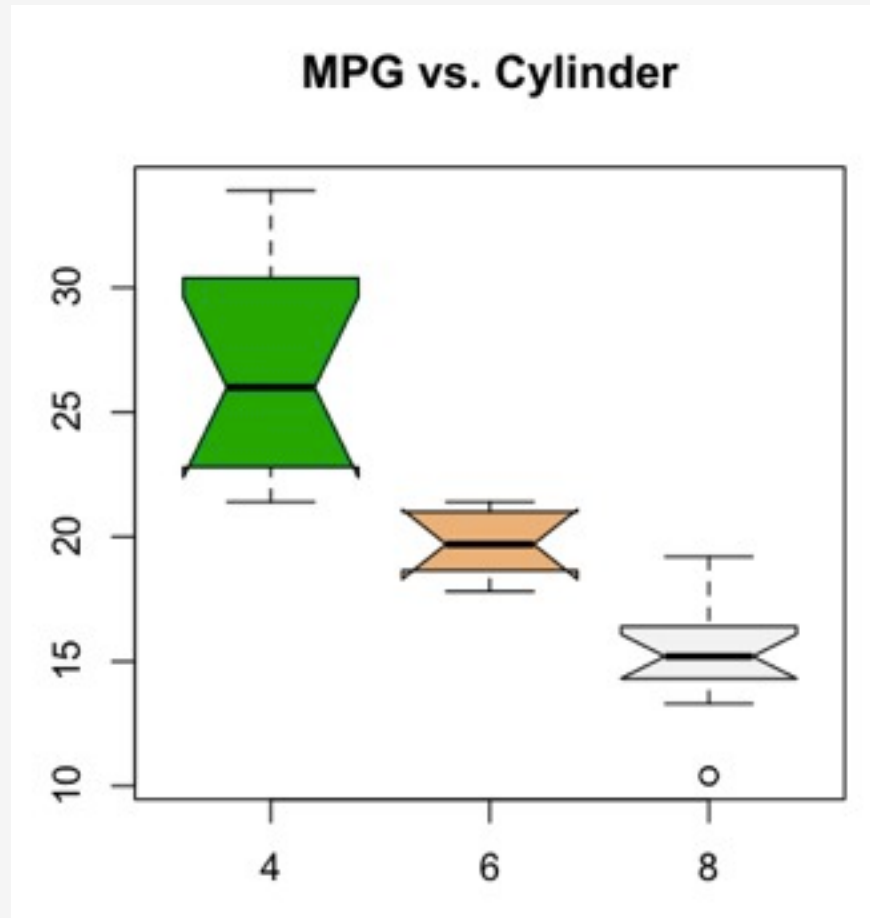
# Graphics: Base: boxplots

```
boxplot(mtcars$mpg, main="MPG Boxplot",xlab="MPG",col="aquamarine")
```

Tuesday, October 29, 13

# Graphics: Base: boxplots

```
hold = boxplot(mtcars$mpg, main="MPG Boxplot",
                xlab="MPG",col="aquamarine")
$stats
        [,1]      # This contains the fivenum summary info
[1,] 10.40
[2,] 15.35
[3,] 19.20
[4,] 22.80
[5,] 33.90

$n                # This contains the number of records in the boxplot
[1] 32

$conf
         [,1]
[1,] 17.11916
[2,] 21.28084

$out
numeric(0)

$group
numeric(0)

$names
[1] ""
```
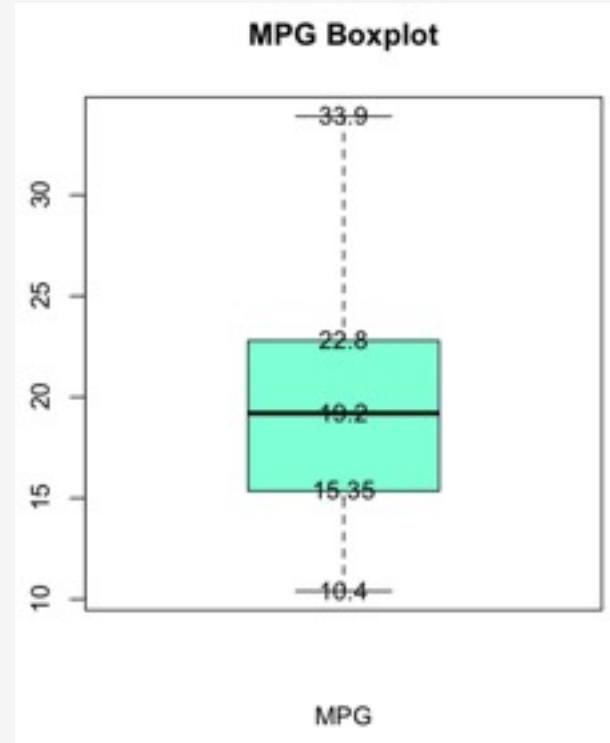
Tuesday, October 29, 13

# Graphics: Base: boxplots

There is only one boxplot so the X coordinate is "1".

```
text(1,10.4,10.4)
text(1,15.35,15.35)
text(1,19.20,19.20)
text(1,22.8,22.8)
text(1,33.9,33.9)
```



MPG Boxplot

We could do this more simply by recognizing that the text function can handle vectors as well as single values (as in the previous example).

```
text(1,hold$stats[,1],hold$stats[,1])
```

Tuesday, October 29, 13

# Graphics: Base: boxplots

As with the barchart command, the boxplot command returns some interesting information such as the fivenum value statistics and some other stuff:

```
tmpvar = boxplot(mpg~cyl,data=mtcars)

$stats
      [,1]  [,2] [,3]
[1,] 21.4 17.80 13.3
[2,] 22.8 18.65 14.3
[3,] 26.0 19.70 15.2      # This row represents the median across the cylinder groups
[4,] 30.4 21.00 16.4
[5,] 33.9 21.40 19.2

$n                       # This row represents the number of observations in each cyl group
[1] 11  7 14

$conf
         [,1]     [,2]     [,3]
[1,] 22.37945 18.29662 14.31323
[2,] 29.62055 21.10338 16.08677

$out
[1] 10.4 10.4

$group
[1] 3 3

$names
[1] "4" "6" "8"
```

Tuesday, October 29, 13

# Graphics: Base: boxplots

As with the barchart command, the boxplot command returns some interesting information such as the fivenum value statistics and some other stuff:

```
tmpvar = boxplot(mpg~cyl,data=mtcars)

tmpvar$stats
      [,1]  [,2] [,3]
[1,] 21.4 17.80 13.3
[2,] 22.8 18.65 14.3
[3,] 26.0 19.70 15.2
[4,] 30.4 21.00 16.4
[5,] 33.9 21.40 19.2

tmpvar$stats[3,]
[1] 26.0 19.7 15.2

mtext(at=1:3, text=tmpvar$stats[3,])

# same as doing the following:

mtext(at=1, text=tmpvar$stats[3,1], side=3) # top margin
mtext(at=2, text=tmpvar$stats[3,2], side=3) # top margin
mtext(at=3, text=tmpvar$stats[3,3], side=3) # top margin
```
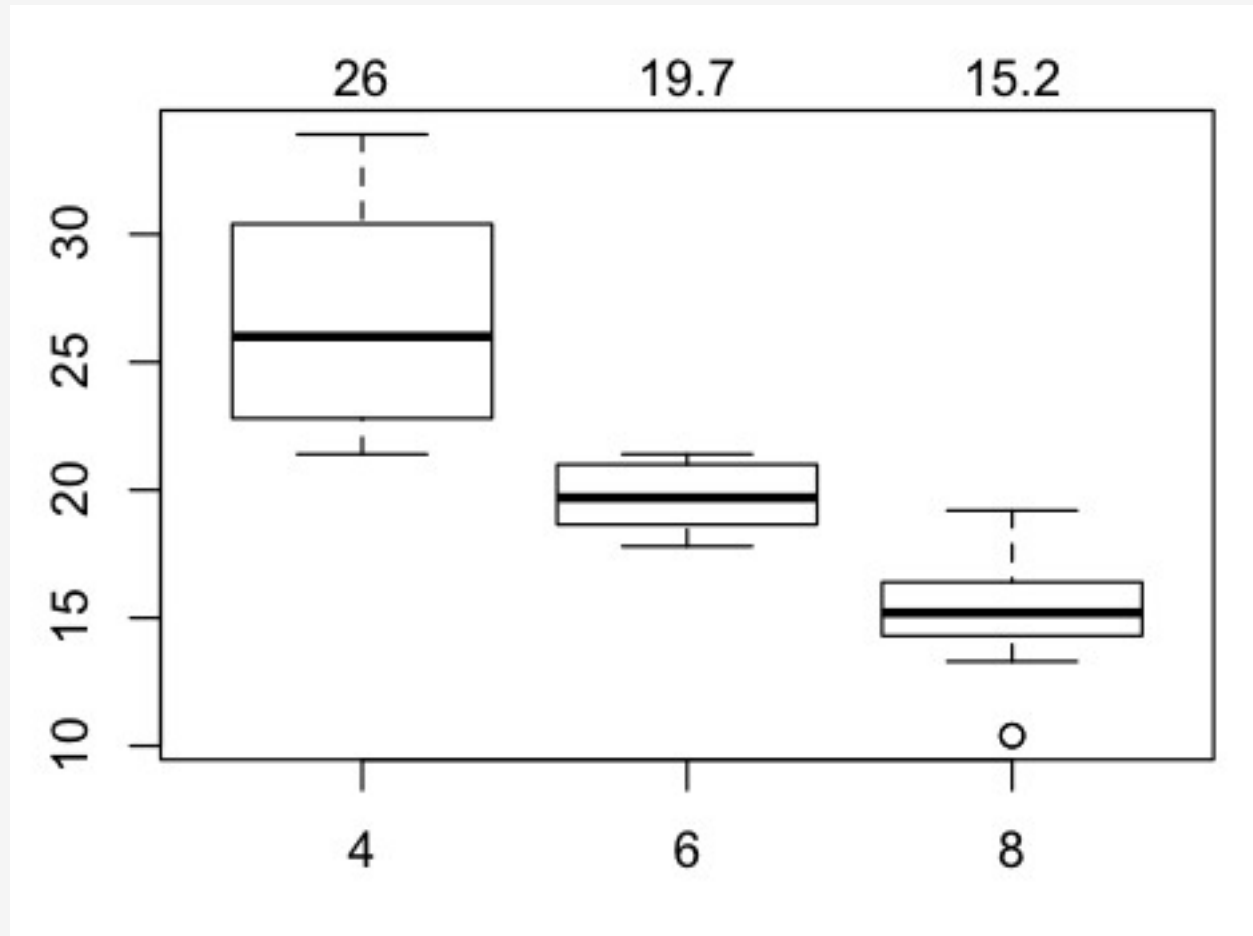
BIOS 560R - Graphics

Tuesday, October 29, 13

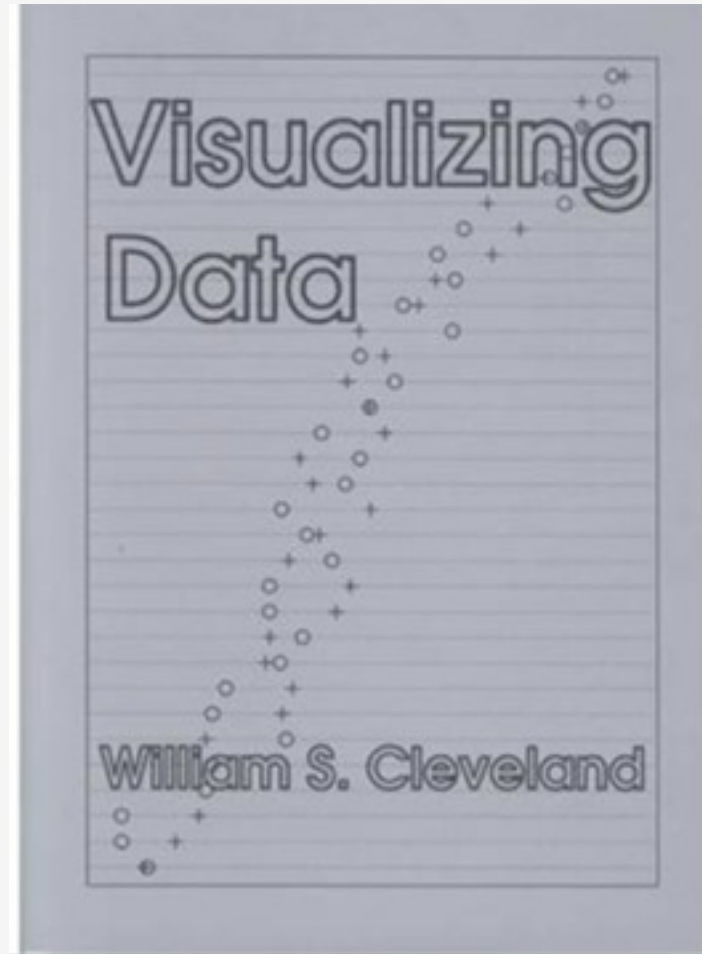# Graphics: Base: boxplots

Tuesday, October 29, 13

# Graphics

## LATTICE Graphics

Note that much of the material in this section is attributable to the Lattice intro available at:

http://lattice.r-forge.r-project.org/Vignettes/src/lattice-intro/lattice-intro.pdf

Tuesday, October 29, 13

# Graphics: lattice

Tuesday, October 29, 13

# Graphics: lattice

Why "lattice" ?

Tuesday, October 29, 13
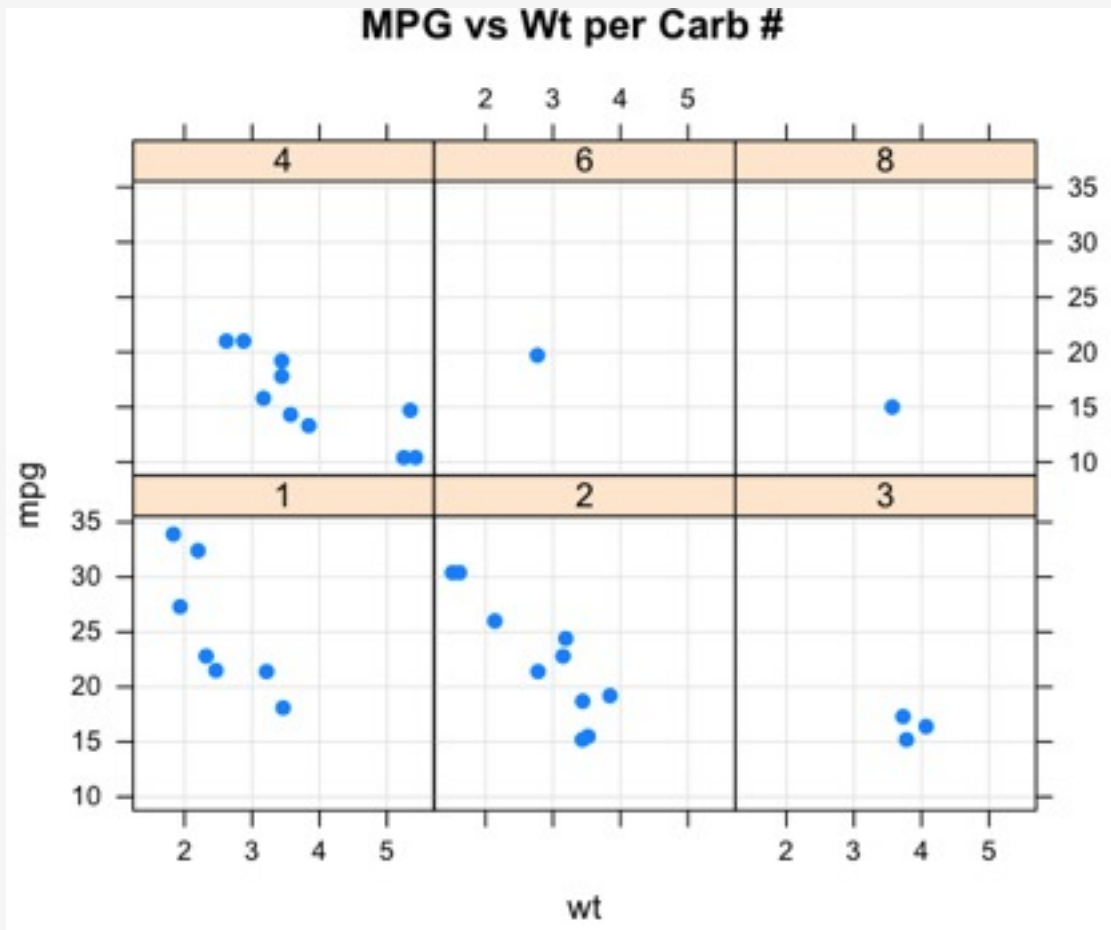
# Graphics: lattice

```
xyplot(mpg~wt|factor(carb), data=mtcars, pch=19,
        main="MPG vs Wt per Carb #", type=c("p","g"))
```



MPG vs Wt per Carb #

Tuesday, October 29, 13

# Graphics: lattice

## Design goals

Visualization is an art, but it can benefit greatly from a systematic, scientific approach. In particular, ? has shown that it is possible to come up with general rules that can be applied to design more effective graphs.

One of the primary goals of Trellis graphics is to provide tools that make it easy to apply these rules, so that the burden of compliance is shifted from the user to the software to the extent possible. Some obvious examples of such rules are:

- Use as much of the available space as possible

- Force direct comparsion by superposition (grouping) when possible

- Encourage comparison when juxtaposing (conditioning): use common axes, add common reference objects such as grids.

Tuesday, October 29, 13

# Graphics: lattice

The following display types are available in lattice.

| Function | Default Display |
|---|---|
| histogram() | Histogram |
| densityplot() | Kernel Density Plot |
| qqmath() | Theoretical Quantile Plot |
| qq() | Two-sample Quantile Plot |
| stripplot() | Stripchart (Comparative 1-D Scatter Plots) |
| bwplot() | Comparative Box-and-Whisker Plots |
| dotplot() | Cleveland Dot Plot |
| barchart() | Bar Plot |
| xyplot() | Scatter Plot |
| splom() | Scatter-Plot Matrix |
| contourplot() | Contour Plot of Surfaces |
| levelplot() | False Color Level Plot of Surfaces |
| wireframe() | Three-dimensional Perspective Plot of Surfaces |
| cloud() | Three-dimensional Scatter Plot |
| parallel() | Parallel Coordinates Plot |

Tuesday, October 29, 13

# Graphics: lattice

| Lattice Function | Description | Traditional Analogue |
|---|---|---|
| barchart() | Barcharts | barplot() |
| bwplot() | Boxplots<br>Box-and-whisker plots | boxplot() |
| densityplot() | Conditional kernel density plots<br>Smoothed density estimate | *none* |
| dotplot() | Dotplots<br>Continuous versus categorical | dotchart() |
| histogram() | Histograms | hist() |
| qqmath() | Quantile–quantile plots<br>Data set versus theoretical distribution | qqnorm() |
| stripplot() | Stripplots<br>One-dimensional scatterplot | stripchart() |
| qq() | Quantile–quantile plots<br>Data set versus data set | qqplot() |
| xyplot() | Scatterplots | plot() |
| levelplot() | Level plots | image() |
| contourplot() | Contour plots | contour() |
| cloud() | 3-dimensional scatterplot | *none* |
| wireframe() | 3-dimensional surfaces | persp() |
| splom() | Scatterplot matrices | pairs() |
| parallel() | Parallel coordinate plots | *none* |

Tuesday, October 29, 13

# Graphics: lattice

Even though lattice graphics is part of R you have to first load the lattice library before using any of the commands:

library(lattice)

Tuesday, October 29, 13

# Graphics: lattice

Trellis displays are defined by the type of graphic and the role different variables play in it. Each display type is associated with a corresponding high-level function (histogram, densityplot, etc.). Possible roles depend on the type of display, but typical ones are:

primary variables: those that define the primary display (e.g., usually some continuous variables)

conditioning variables: divides data into subgroups, each of which are presented in a different panel (e.g., a category or factor).

grouping variables: subgroups are contrasted within panels by superposing the corresponding displays (e.g., gender in the last example).

```
http://lattice.r-forge.r-project.org/Vignettes/src/lattice-intro/lattice-intro.pdf
```

Tuesday, October 29, 13

# Graphics: lattice

Factor review:

```
factor                    package:base                    R Documentation

Factors

Description:

    The function 'factor' is used to encode a vector as a factor (the
    terms 'category' and 'enumerated type' are also used for factors).
    If argument 'ordered' is 'TRUE', the factor levels are assumed to
    be ordered.  For compatibility with S there is also a function
    'ordered'.

    'is.factor', 'is.ordered', 'as.factor' and 'as.ordered' are the
    membership and coercion functions for these classes.

Usage:

    factor(x = character(), levels, labels = levels,
           exclude = NA, ordered = is.ordered(x), nmax = NA)
```

Tuesday, October 29, 13

# Graphics: lattice

```
Factor review:

mtcars$am = factor(mtcars$am, labels=c("auto","manual"))

mtcars$am
 [1] manual manual manual auto    auto    auto    auto    auto    auto    auto
[11] auto    auto    auto    auto    auto    auto    auto    manual manual manual
[21] auto    auto    auto    auto    auto    manual manual manual manual manual
[31] manual manual
Levels: auto manual

levels(mtcars$am)
[1] "auto"   "manual"
```

Tuesday, October 29, 13

# Graphics: lattice

Lattice/Trellis graphics display a variable,or a relationship between variables, conditioned on one or more other variables. The typical format is:

*graph_type(formula,data=some.data)  where formula specifies the variable(s) to display and any conditioning variables.*

*~x|A means display numeric variable x for each level of factor A on separate panels.*
```
bwplot(~mpg | factor(am), data=mtcars)
```

*x ~ A means display numeric variable x for each level of A on One panel.*
```
bwplot(mpg ~ factor(am), data=mtcars)
```

*~x means display numeric variable x alone.*
```
hist(~mpg, data=mtcars)
```

Tuesday, October 29, 13

# Graphics: lattice

| graph_type | description | formula examples |
| --- | --- | --- |
| barchart | bar chart | x~A or A~x |
| bwplot | boxplot | x~A or A~x |
| cloud | 3D scatterplot | z~x*y\|A |
| contourplot | 3D contour plot | z~x*y |
| densityplot | kernal density plot | ~x\|A*B |
| dotplot | dotplot | ~x\|A |
| histogram | histogram | ~x |
| levelplot | 3D level plot | z~y*x |
| parallel | parallel coordinates plot | dataframe |
| splom | scatterplot matrix | dataframe |
| stripplot | strip plots | A~x or x~A |
| xyplot | scatterplot | y~x\|A |

Tuesday, October 29, 13

# Graphics: lattice: xyplot

The typical Basic form when calling a lattice function such as xyplot looks like:

```
vertical.axis.variable ~ horizontal.axis.variable

xyplot(mpg ~ wt, data = mtcars, pch = 19, main = "MPG vs. Weight")
```

Tuesday, October 29, 13

# Graphics: lattice: xyplot
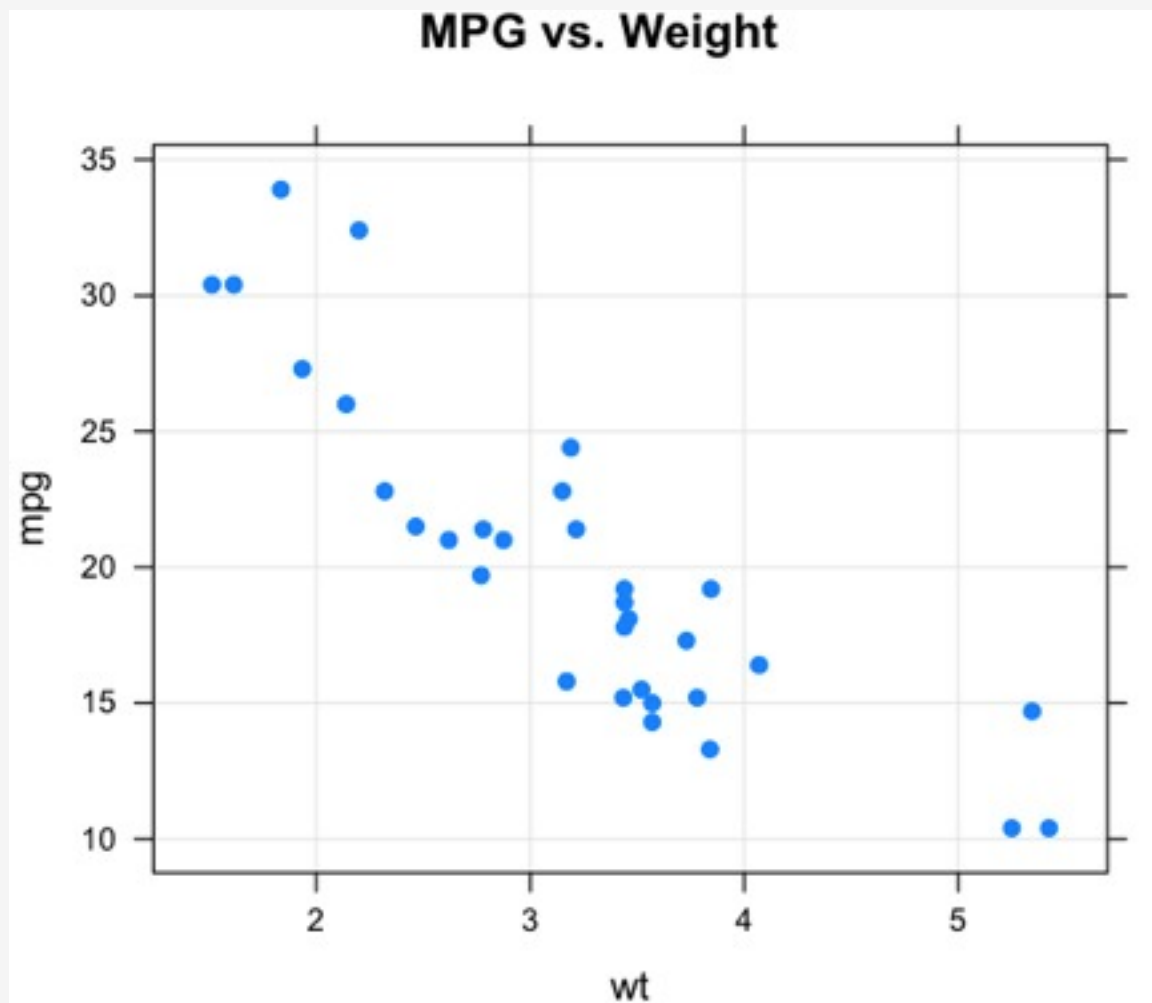


**MPG vs. Weight**

Tuesday, October 29, 13
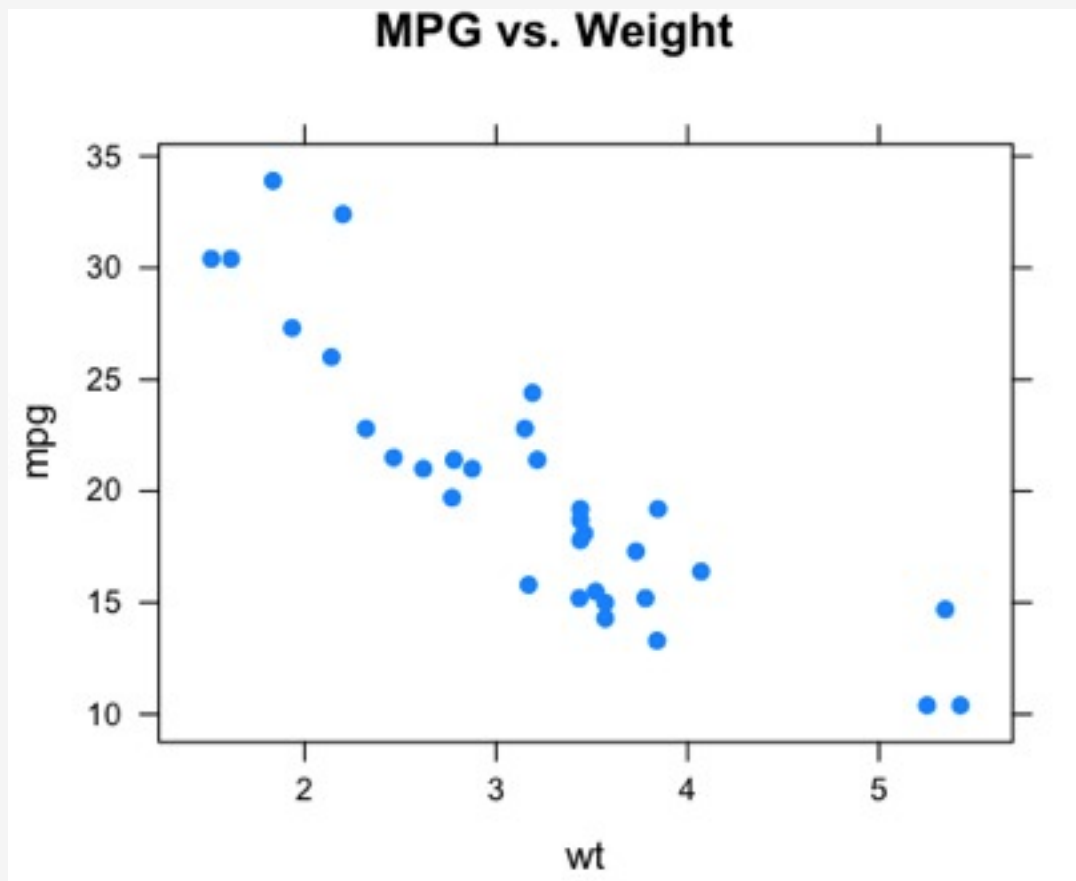
# Graphics: lattice: xyplot

* Check out the "type" argument that let's us specify the type of graph we want (e.g. points, lines, grid)

```
xyplot(mpg ~ wt, data = mtcars, pch = 19,
       main = "MPG vs. Weight", type=c("p","g"))
```

Tuesday, October 29, 13

# Graphics: lattice: xyplot



MPG vs. Weight

Tuesday, October 29, 13

# Graphics: lattice: Conditioning

* So check out this formula. "mpg" is the y variable and "wt" is the x variable.
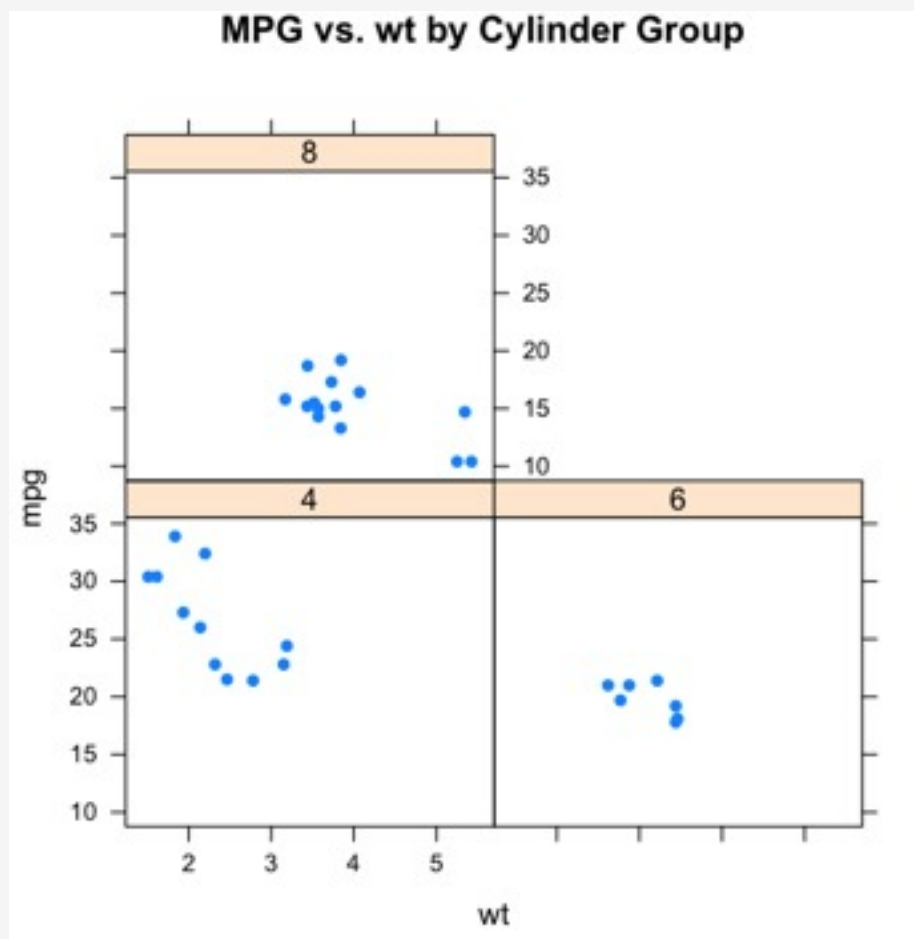
```
xyplot(mpg ~ wt | factor(cyl), data = mtcars, pch=16,
       main = "MPG vs. wt by Cylinder Group"))
```

* "cylinder" is a conditioning variable (preceeded by the | character). The conditioning variable divides the plot into separate panels

* Conditioning variables are usually categorical.

* Note that the "data" argument tells the function what data frame to use.

Tuesday, October 29, 13

# Graphics: lattice: Conditioning



MPG vs. wt by Cylinder Group

Tuesday, October 29, 13

# Graphics: lattice: Conditioning

* So check out this formula. "mpg" is the y variable and "wt" is the x variable.

```
xyplot(mpg ~ wt | factor(cyl), data = mtcars, pch=16, type=c("p","g"),
       main = "MPG vs. wt by Cylinder Group", layout=c(1,3))
```

* "cylinder" is a conditioning variable (preceeded by the | character). The conditioning variable divides the plot into separate panels

* Conditioning variables are usually categorical.

* Note that the "data" argument tells the function what data frame to use.

Tuesday, October 29, 13

# Graphics: lattice: Conditioning

In the Trellis terminology, this plot consists of three panels. Each panel in this case contains a scatterplot and above each panel there is a strip that presents the level of the conditioning variable.
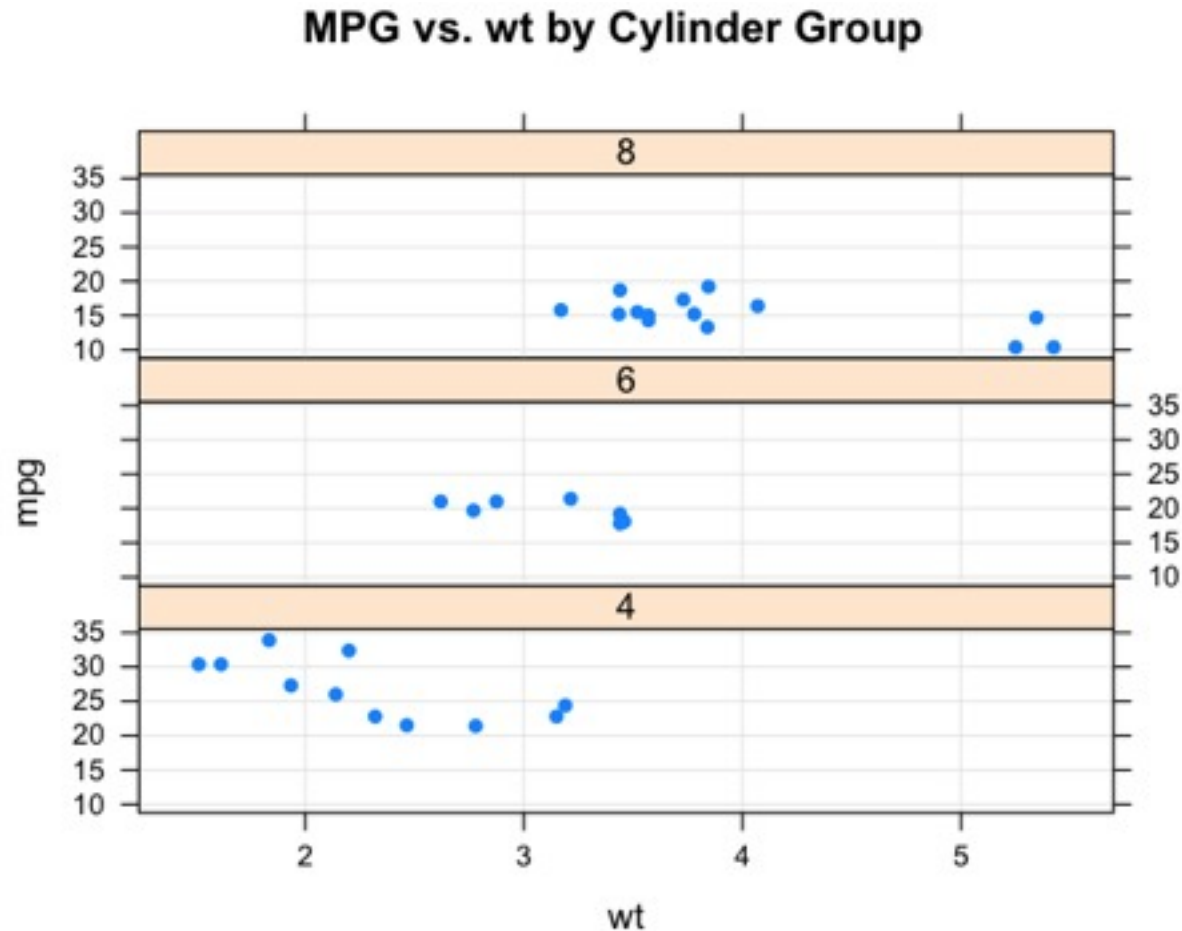


**MPG vs. wt by Cylinder Group**

Tuesday, October 29, 13

# Graphics: lattice: Conditioning

Remember our early X/Y plot with Base graphics where we wanted to plot MPG vs Weight for each cylinder category ? To get a side-by-side panel plot we did something like this:

```
par(mfrow=c(1,3))

fourcyl   =   mtcars[mtcars$cyl == 4,]
sixcyl    =   mtcars[mtcars$cyl == 6,]
eightcyl  = mtcars[mtcars$cyl == 8,]

plot(fourcyl$wt, fourcyl$mpg, main = "MPG vs Wt for 4 Cyl", ylim=c(0,40))
plot(sixcyl$wt, sixcyl$mpg, main = "MPG vs Wt for 6 Cyl", ylim=c(0,40))
plot(eightcyl$wt, eightcyl$mpg, main = "MPG vs Wt for 8 Cyl",
ylim=c(0,40))
```

Tuesday, October 29, 13

# Graphics: lattice: Conditioning

Remember our early X/Y plot with Base graphics where we wanted to plot MPG vs Weight for each cylinder category ? To get a side-by-side panel plot we did something like this:

Tuesday, October 29, 13
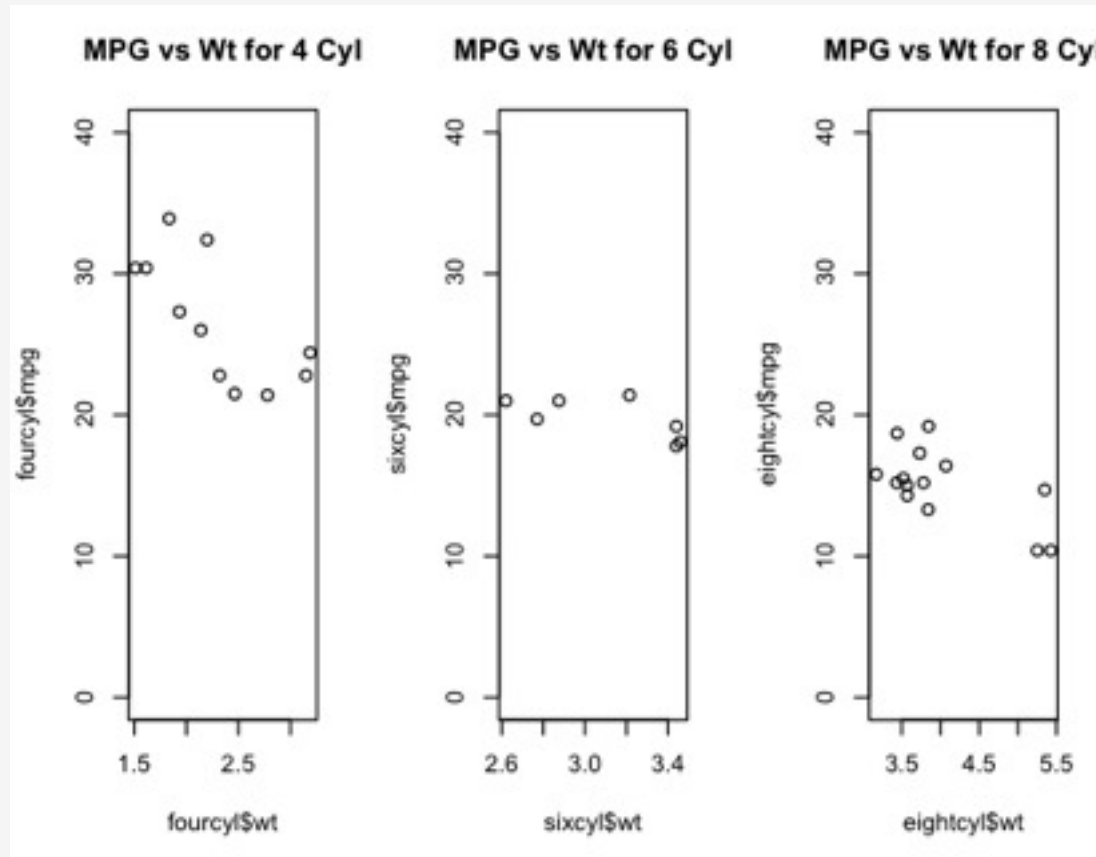
# Graphics: lattice: Conditioning

With lattice plots its much easier:

```
library(lattice)
xyplot(mpg~wt|factor(cyl),data = mtcars,layout=c(3,1))
```



MPG vs. wt by Cylinder Group

Tuesday, October 29, 13
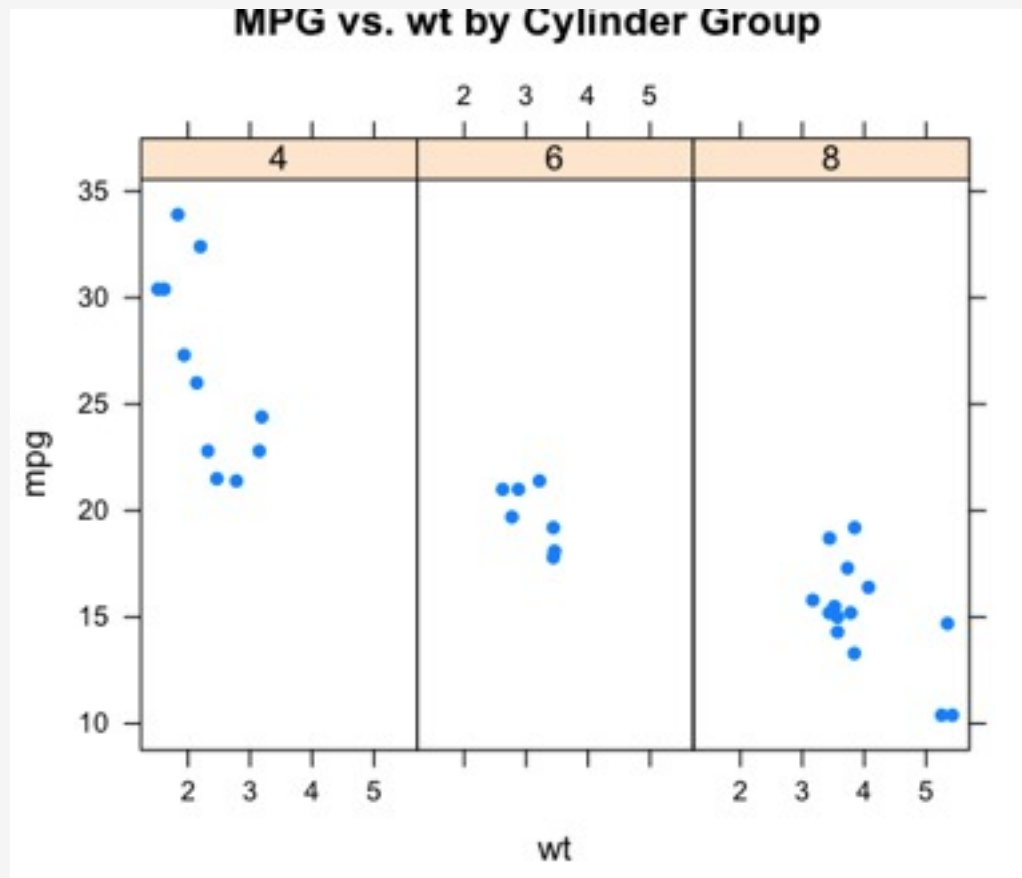
# Graphics: lattice: Conditioning

This works independently of the chart type:

```
histogram(~mpg, data = mtcars, main = "MPG for all groups")
```



MPG for all groups

Tuesday, October 29, 13
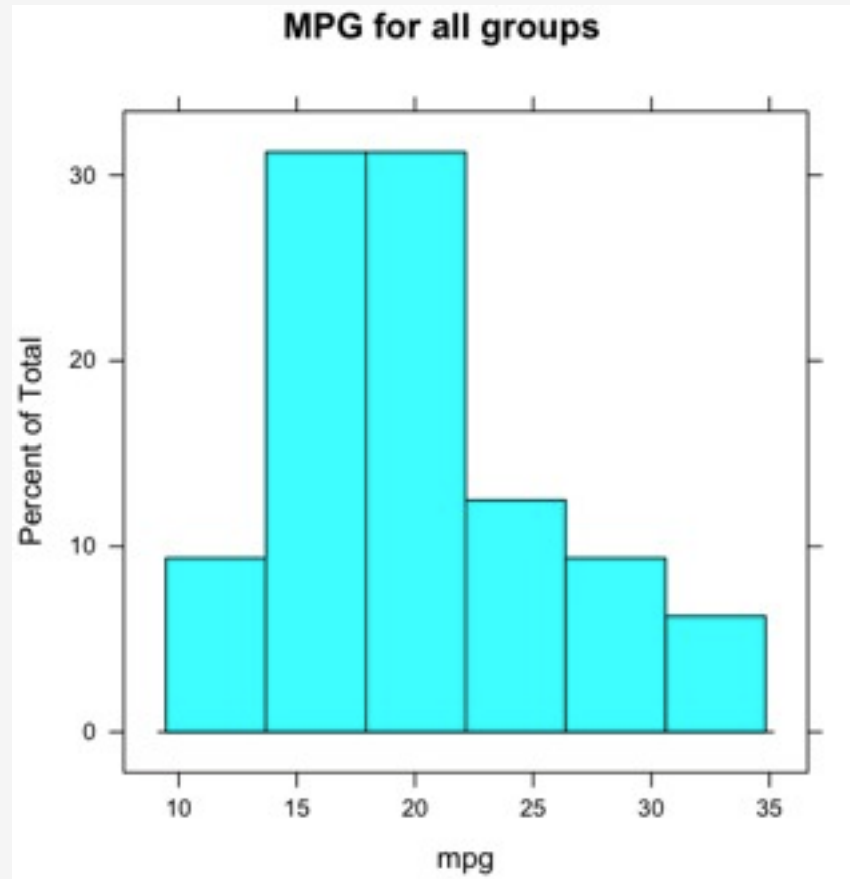
# Graphics: lattice: Conditioning

This works independently of the chart type:

```
histogram(~mpg | factor(cyl), data = mtcars, layout = c(3,1))
```

Tuesday, October 29, 13
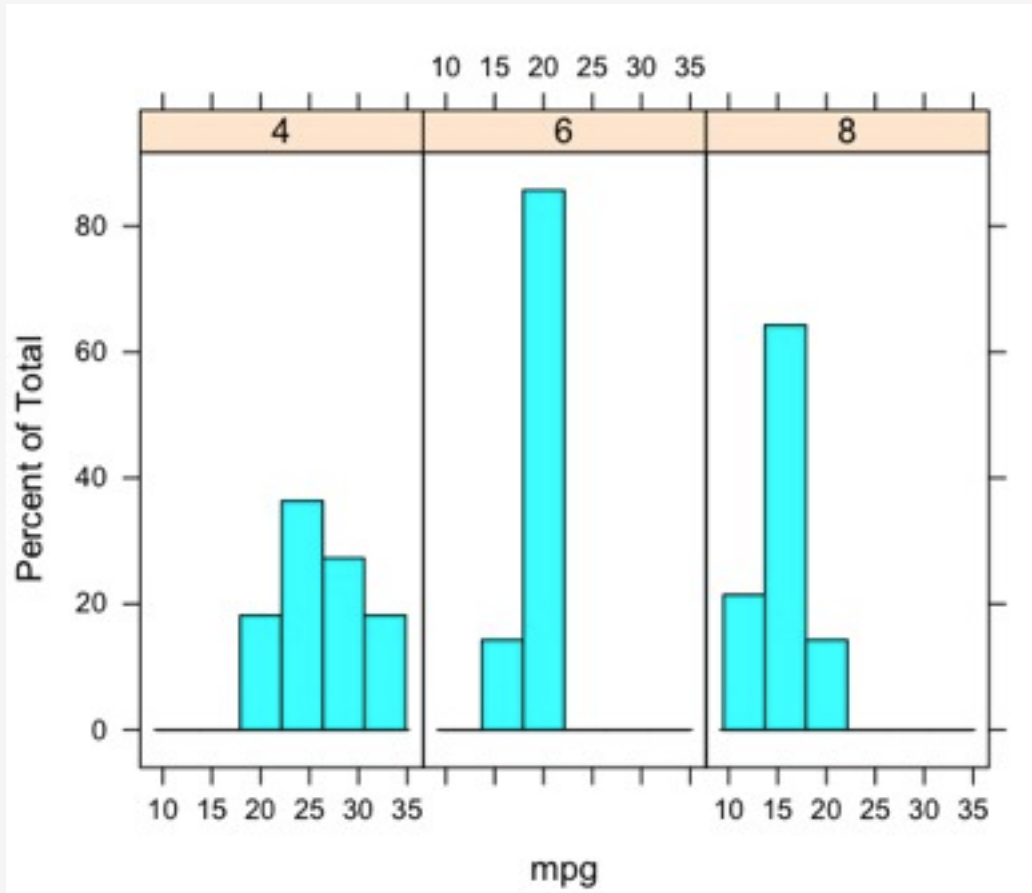
# Graphics: lattice: groups

Remember our early X/Y plot with Base graphics where we wanted to plot MPG vs Weight for each transmission type and have it represented by a different color or plot character ? The most basic solution looked like this (although we did improve it somewhat).

```
plot(mtcars$wt, mtcars$mpg, main="MPG vs. Weight", type="n",
    ylim=c(0,40))

auto = mtcars[mtcars$am == 0,]

manu = mtcars[mtcars$am == 1,]

points(auto$wt, auto$mpg, col="blue")

points(manu$wt, manu$mpg, col="red")

legend("topright",c("auto","manu"),fill=c("blue","red"))
```
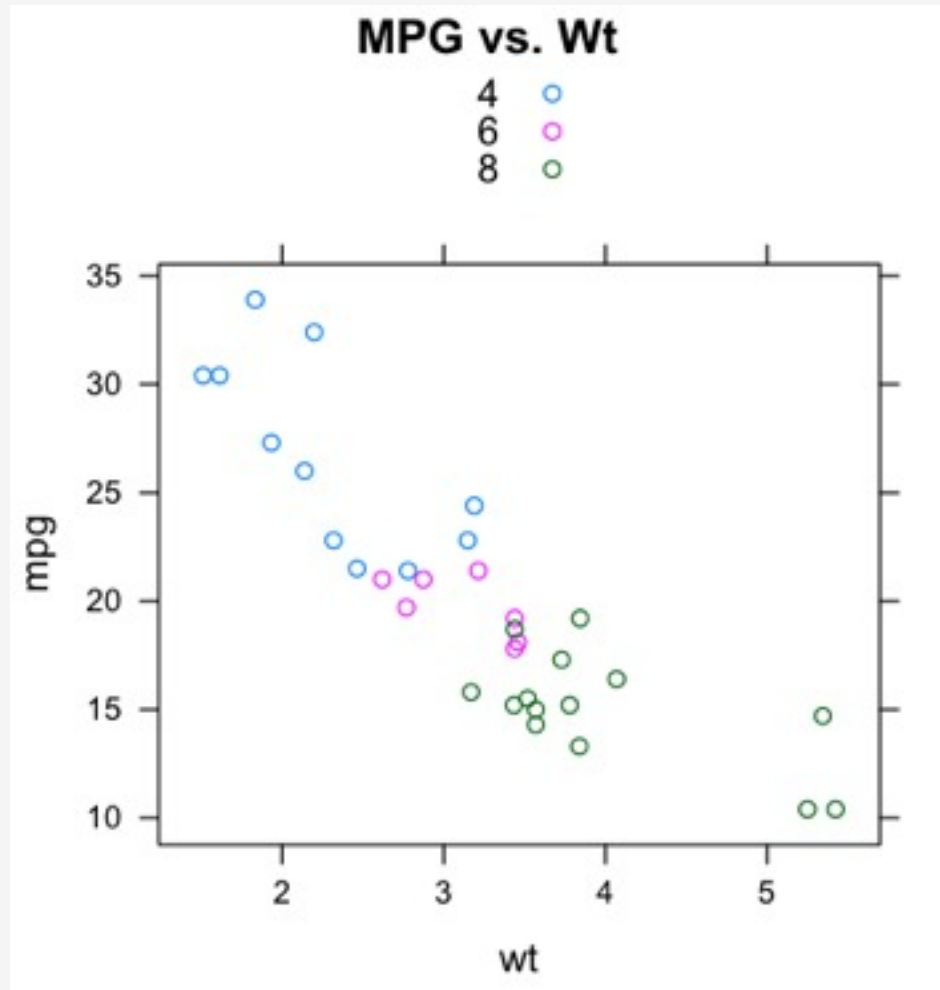
Tuesday, October 29, 13

# Graphics: lattice: groups



**MPG vs. Weight**

Tuesday, October 29, 13
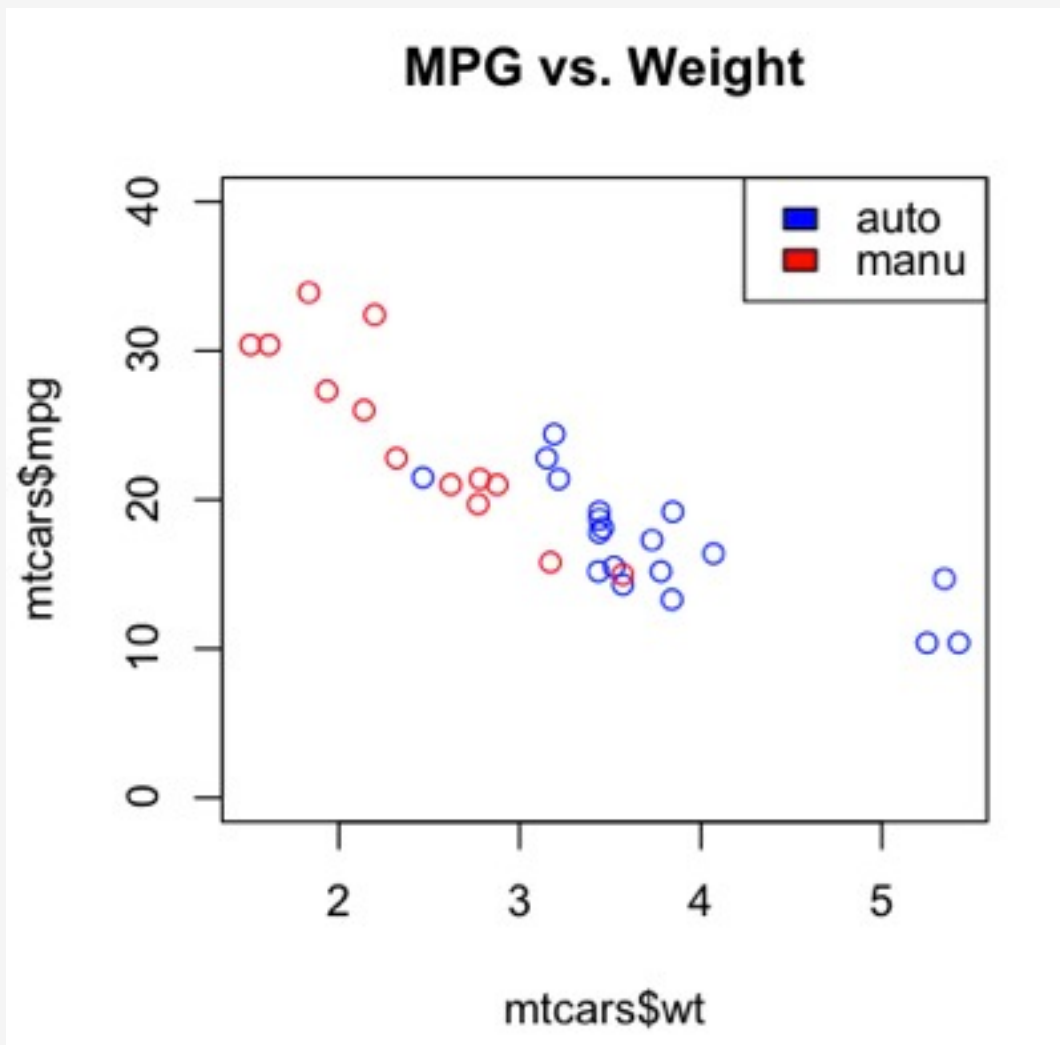
# Graphics: lattice: groups

```
xyplot(mpg~wt, data=mtcars, groups=factor(am), type=c("p","g"),
        main="MPG vs. Wt", auto.key=TRUE)
```
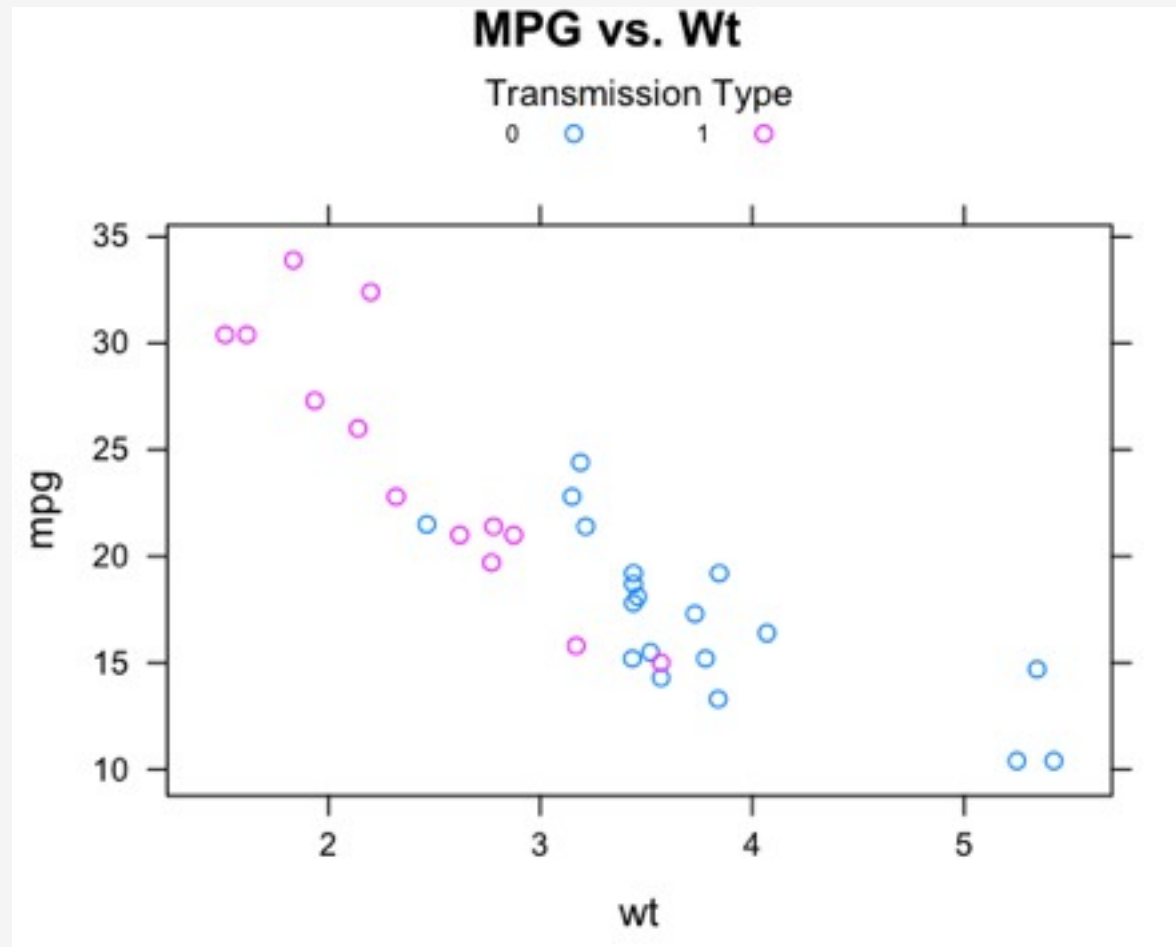
Tuesday, October 29, 13
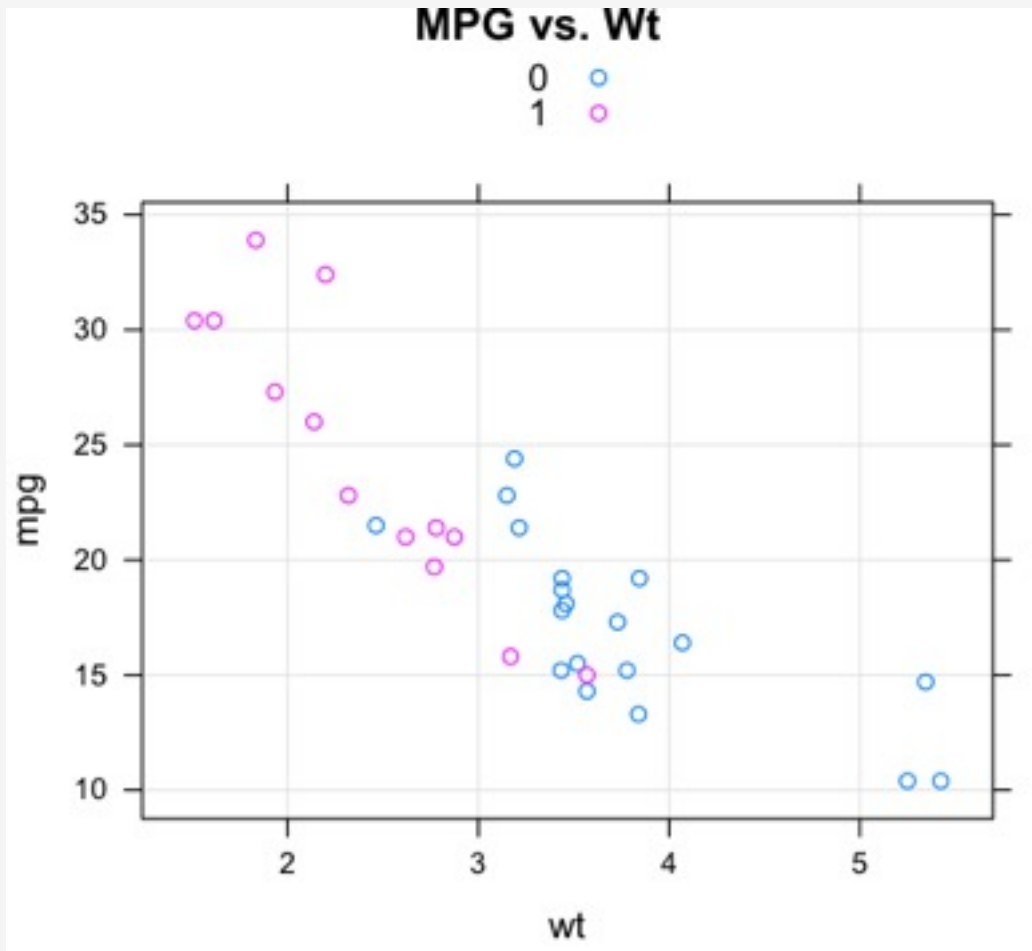
# Graphics: lattice: groups

```
xyplot(mpg~wt, data=mtcars, groups=cyl, main="MPG vs. Wt", auto.key=TRUE)
```
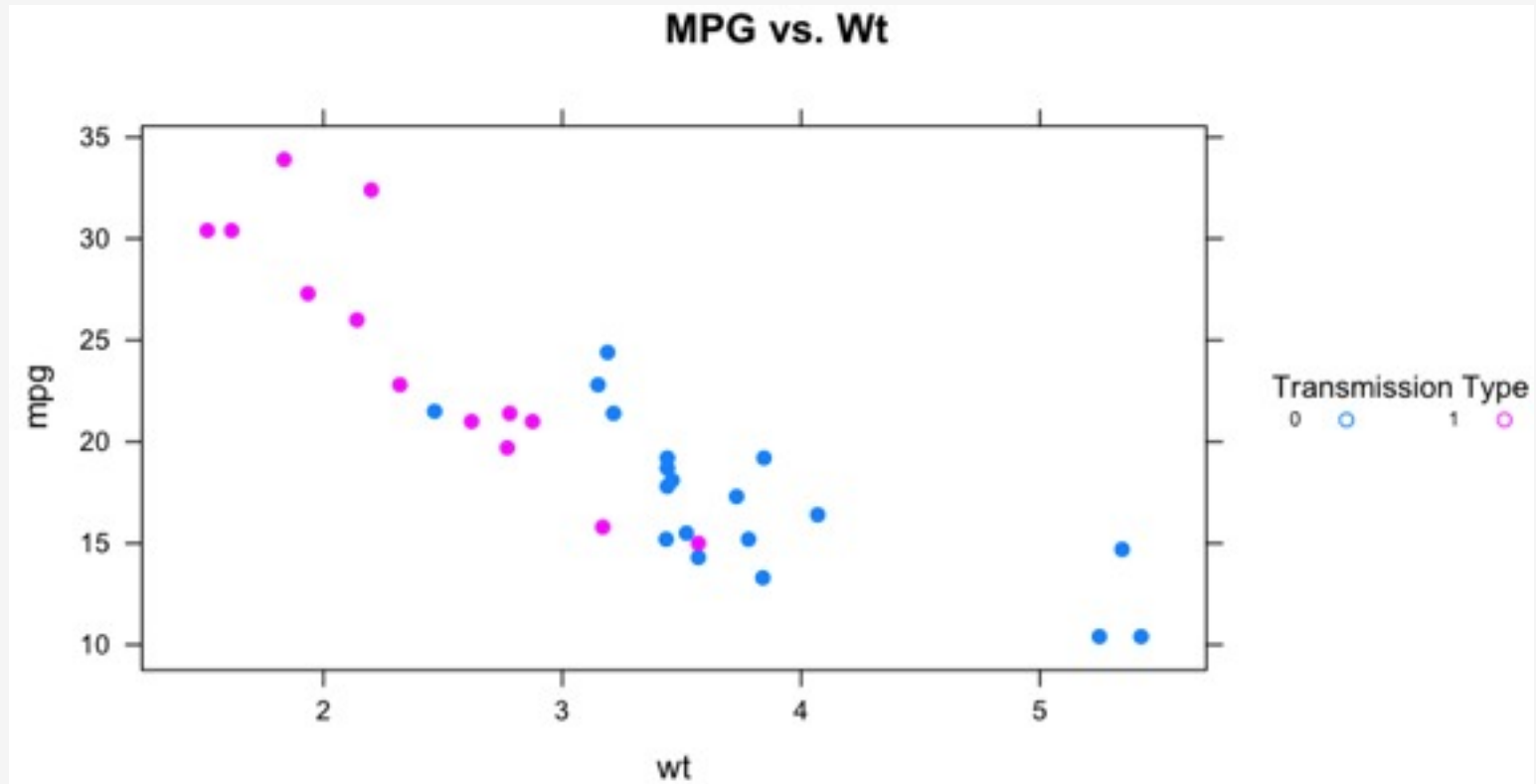
Tuesday, October 29, 13

# Graphics: lattice: legends

```
xyplot(mpg~wt, data=mtcars, groups=factor(am),
       main="MPG vs. Wt",
       auto.key=list(title="Transmission Type",cex=0.6,columns=2))
```
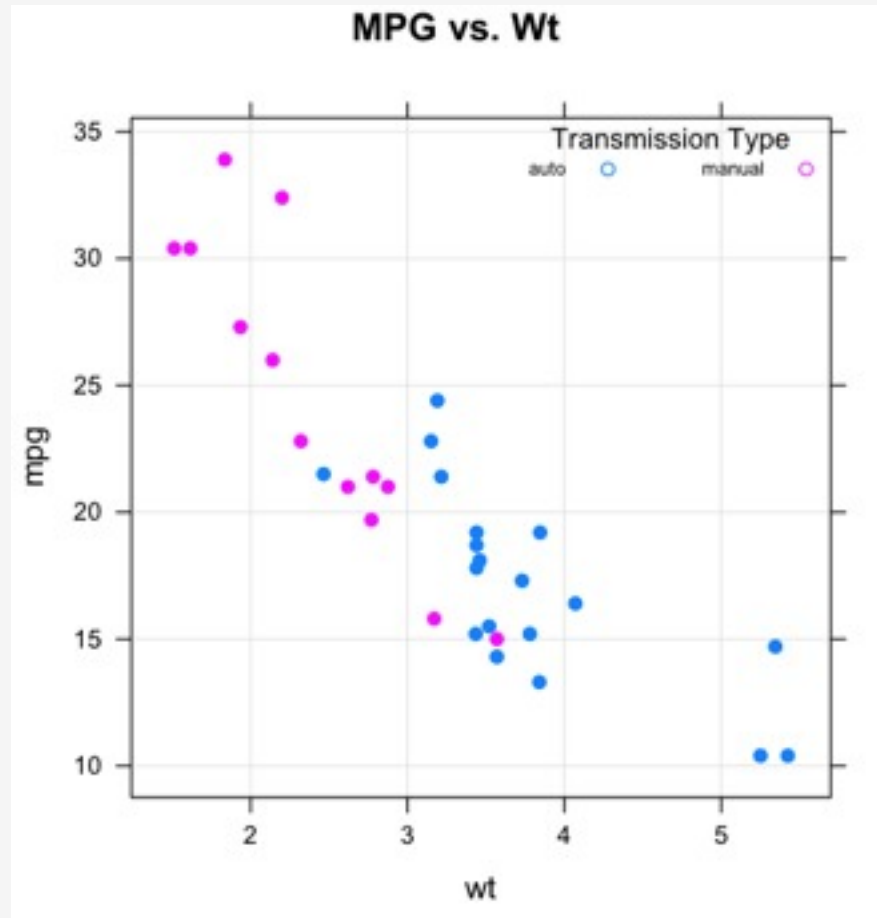
Tuesday, October 29, 13

# Graphics: lattice: legends

```
xyplot(mpg~wt, data=mtcars, groups=factor(am), main="MPG vs. Wt", pch=19,
        auto.key = list(title="Transmission Type",
                          cex=0.6,columns=2,space="right"))
```

Tuesday, October 29, 13

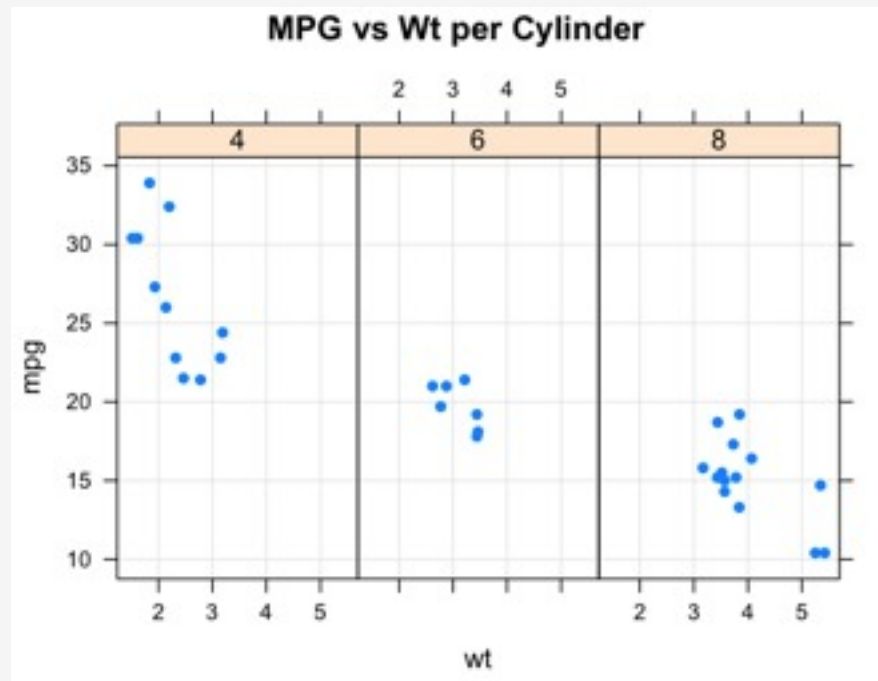# Graphics: lattice: legends

```
xyplot(mpg~wt, data=mtcars, groups=factor(am), main="MPG vs. Wt",
       pch=19, type=c("p","g"),
       auto.key = list(title="Transmission Type", cex=0.6, columns=2,
       corner=c(1,1))
```

Tuesday, October 29, 13

```
xyplot(mpg ~ wt | factor(cyl), data=mtcars,
       pch=16,layout=c(3,1), main="MPG vs Wt per Cylinder",
       type=c("p","g"))
```

In this case we have a conditioning variable, which in this example is the variable cyl . So we have three panels:

Tuesday, October 29, 13

# Graphics: lattice: grouping and conditions

```
xyplot(mpg ~ wt | factor(cyl), data=mtcars,
        pch=16,layout=c(3,1),
        groups = factor(am,labels=c("Auto","Manual")),
        type=c("p","g"),auto.key = TRUE)
```

In this case we have a grouping variable, which in this example is the variable am .
The grouping variable segregates data into subgroups within each panel via color.

Tuesday, October 29, 13
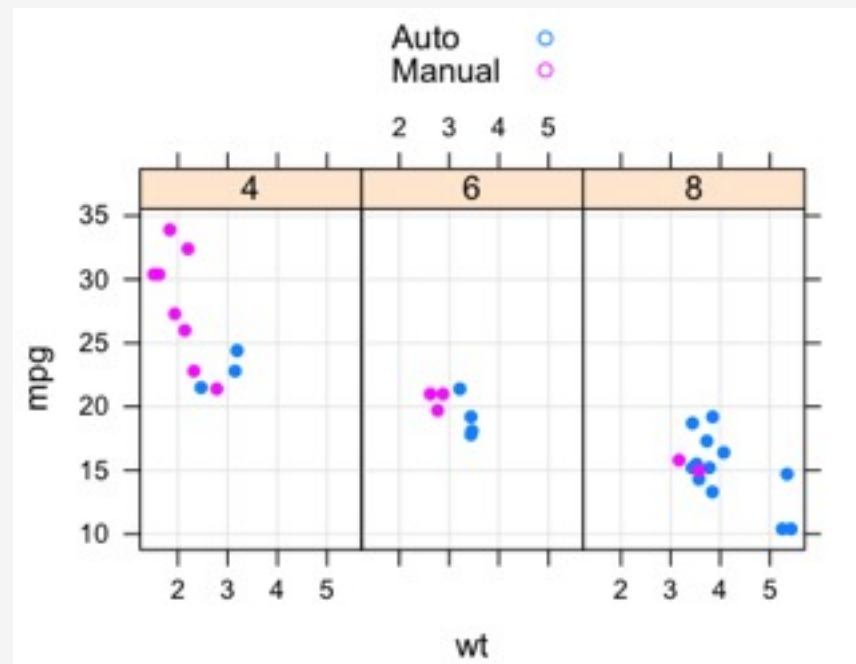
# Graphics: lattice: xyplot

```
xyplot(mpg ~ wt | factor(cyl), data=mtcars,
       main = "MPG vs Wt per cylinder Group",
       groups = factor(am), pch=21, type=c("p","g"),
       layout = c(1,3), auto.key=list(columns=2))
```

Tuesday, October 29, 13
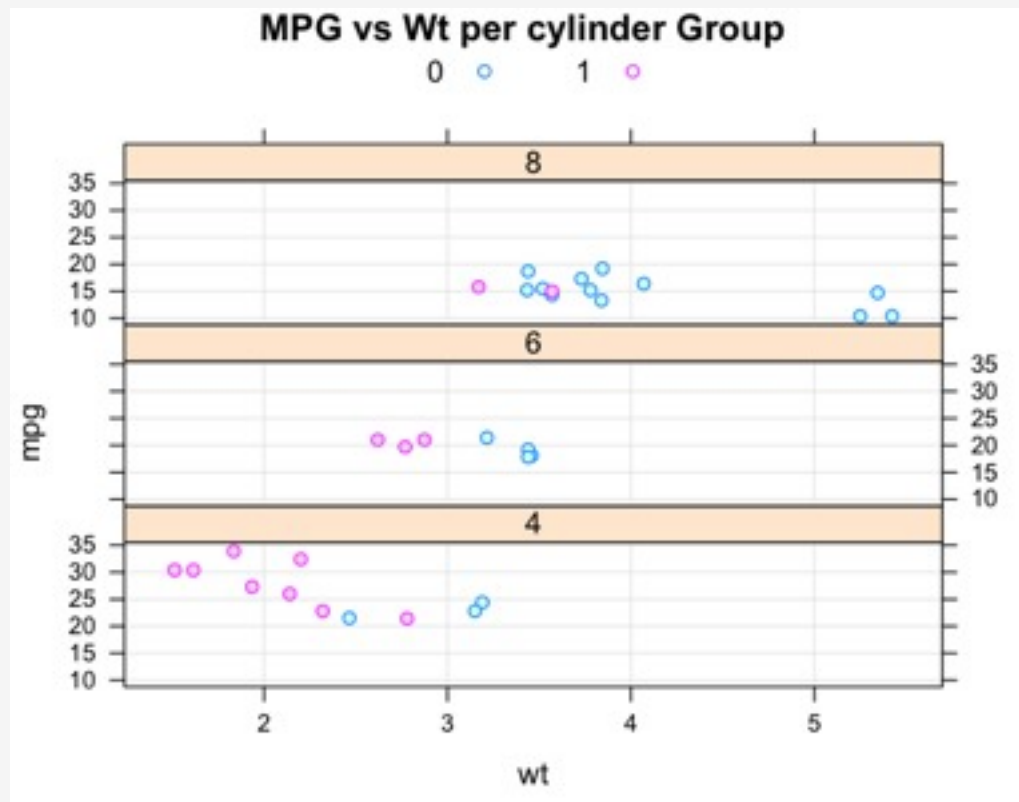
# Graphics

Some useful arguments that you will see again and again:

**data =** (this specifies the data frame of interest)

**xlab =** and **ylab =** (this indicates the X/Y axis labels

**aspect =** (sets the aspect ration of the plot – not so common but useful)

**xlim =** and **ylim =** (sets the numerical limits of the X and Y vals)

**main =** (sets the title for the plot)

**cex =** (changes the size of an element like text or the plotting symbol

```
http://polisci.msu.edu/jacoby/icpsr/graphics/lattice/Lattice,%20ICPSR%202012%20Outline,%20Ver
%201.pdf
```

Tuesday, October 29, 13

# Graphics

Some useful arguments that you will see again and again:

The **auto.key = T** argument gives you an automatic legend.

The **scales =** argument uses a "mini-language" to modify the characteristics of the display axes such as ticks, and tick labels. Its basically a "list" (in the R sense)

The **group =** argument lets you pass a categorical variable that can be used to differentiate subgroups in a single display.

```
http://polisci.msu.edu/jacoby/icpsr/graphics/lattice/Lattice,%20ICPSR%202012%20Outline,%20Ver
%201.pdf
```

Tuesday, October 29, 13

# Summary

Strengths of Base Graphics:

    Oldest and Most Well Known

    Has Both High Level and Low Level Functions

    Can easily assemble a plot in stages

    Primitive functions (e.g. points, abline, lines) are simple to use

Weaknesses of Base Graphics:

    Lots of independent functions to keep up with

    Each high level plot command has its own specific arguments

    You have to pick colors and plot characters using basic programming tricks

Tuesday, October 29, 13

# Summary

Strengths of Lattice Graphics

    Developed according to a well-thought out philosophy

    Can generate colors, legends, and axes as part of the high level commands

    Can use "condition" variables

    Can group variables

Weaknesses of Lattice Graphics:

    Commands can get long and involved

    Commands can have many arguments

Tuesday, October 29, 13

# Graphics: lattice: xyplot

THE FOLLOWING SLIDES ARE SUPPLEMENTAL THOUGH ARE VERY USEFUL

Tuesday, October 29, 13
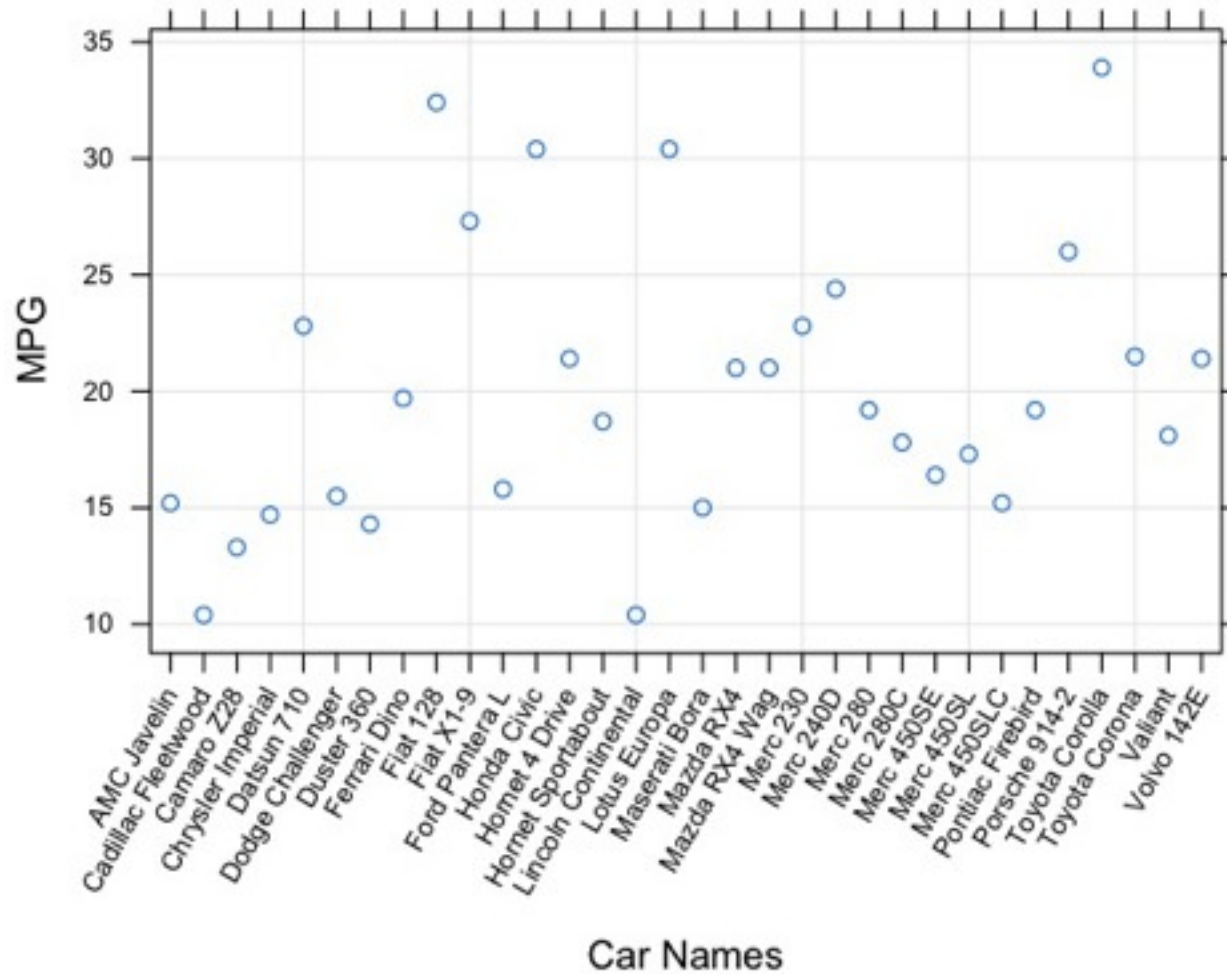
# Graphics: lattice: xyplot

Remember our early X/Y plot with Base graphics where we wanted to plot MPG vs Weight and have the point sizes to reflect the cylinder group (4,6, or 8) ? It was quite a bit of work to do with Base graphics. Here we can do it with one line:

```
xyplot(mpg ~ wt, groups = factor(cyl), cex=3:1, auto.key=T)
```
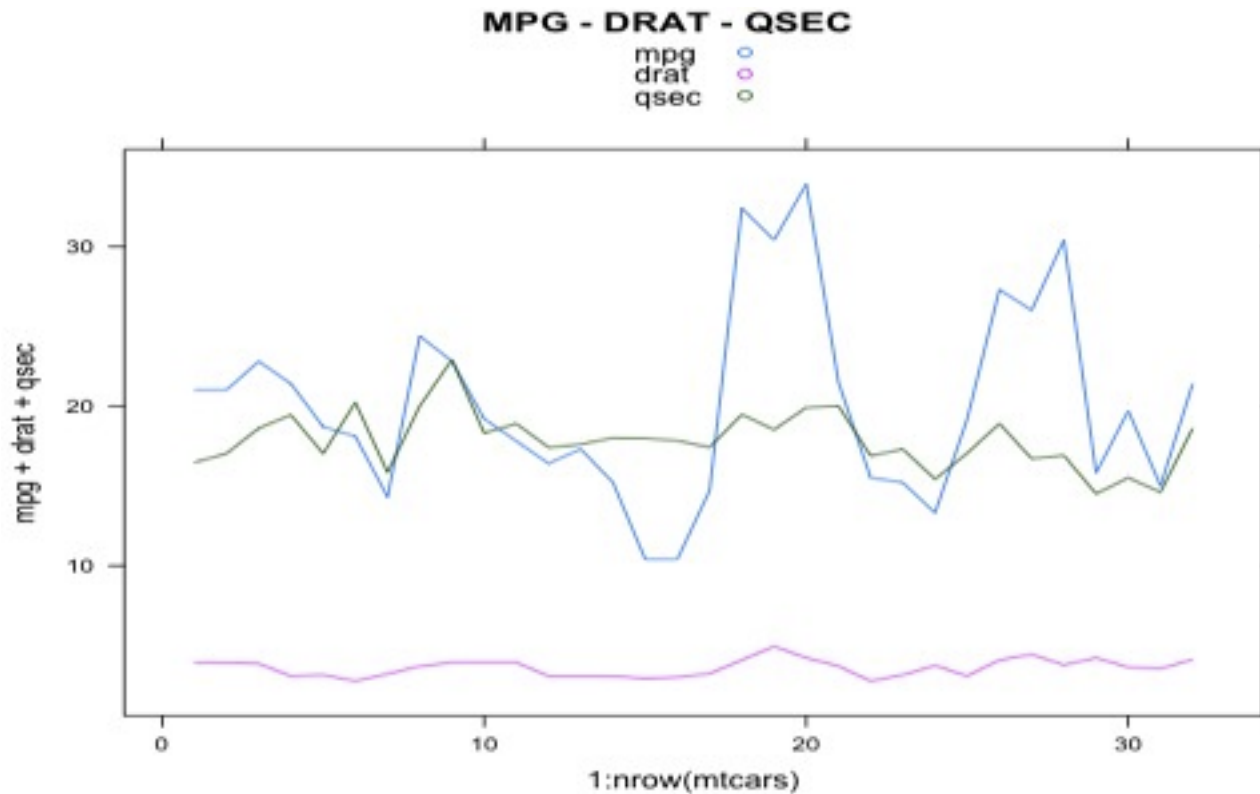
Tuesday, October 29, 13
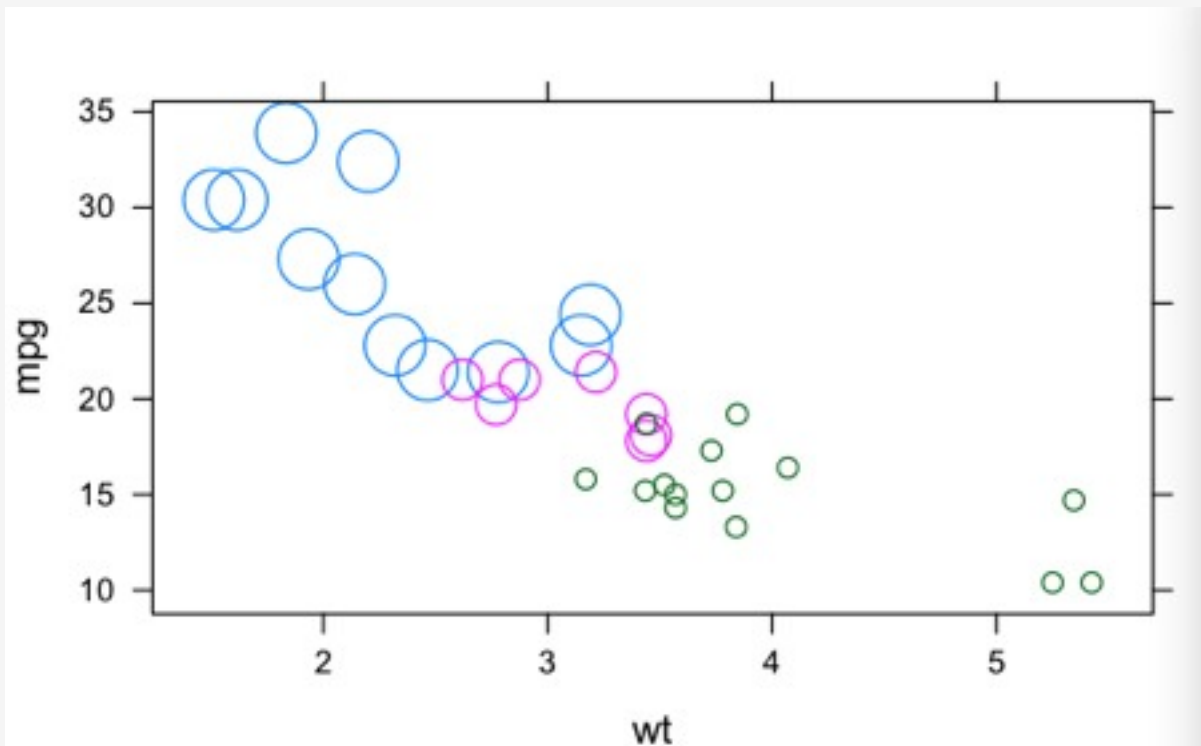
# Graphics: lattice: xyplot

```
xyplot(mpg ~ factor(rownames(mtcars)),
       mtcars, xlab = "Car Names", ylab = "MPG",
       scales = list(cex = 0.7,x = list(rot = 60)),type=c("p","g"))
```



BIOS 560R - Graphics

Tuesday, October 29, 13

# Graphics: lattice: xyplot

```
xyplot(mpg + drat + qsec ~ 1:nrow(mtcars),
       data= mtcars, type="l",auto.key=T,
       main = "MPG - DRAT - QSEC")
```



MPG - DRAT - QSEC

Tuesday, October 29, 13

# Graphics: lattice: xyplot

Every lattice function has a default "panel" function. It is usually the name of the function prefixed by **panel.** As an example here is the basic xyplot command:

```
xyplot (y ~ x, data= dataset)
```

This is actually equivalent to:

```
xyplot (y ~ x, data= dataset, panel = panel.xyplot)
```

Yea, its a bit weird but if you have to stack up a bunch of options and arguments it becomes easier. You'll see eventually.

So a call like this:

```
xyplot(y ~ x, data=dataset, col="black")
```

will "silently" pass the "col='black' argument to panel.xyplot.

```
http://polisci.msu.edu/jacoby/icpsr/graphics/lattice/Lattice,%20ICPSR%202012%20Outline,%20Ver
%201.pdf
```

Tuesday, October 29, 13

# Graphics: lattice: xyplot

We can look at group interaction within the panels

```
xyplot(mpg ~ wt | factor(cyl), data = mtcars, pch=16,
        main = "MPG vs. wt by Cylinder Group",
        layout=c(1,3),
        groups = factor(gear),
        auto.key=TRUE)
```

Tuesday, October 29, 13

# Graphics: lattice: xyplot

So a call like this:

```
xyplot(y ~ x, data=dataset, col="black")
```

will "silently" pass the "col='black' argument to panel.xyplot. Its basically the equivalent of:

```
xyplot(y ~ x, data = dataset,
    panel = function (x, y) {
        panel.xyplot(x, y, col = "black") }
    )
```

```
http://polisci.msu.edu/jacoby/icpsr/graphics/lattice/Lattice,%20ICPSR%202012%20Outline,%20Ver
%201.pdf
```

Tuesday, October 29, 13
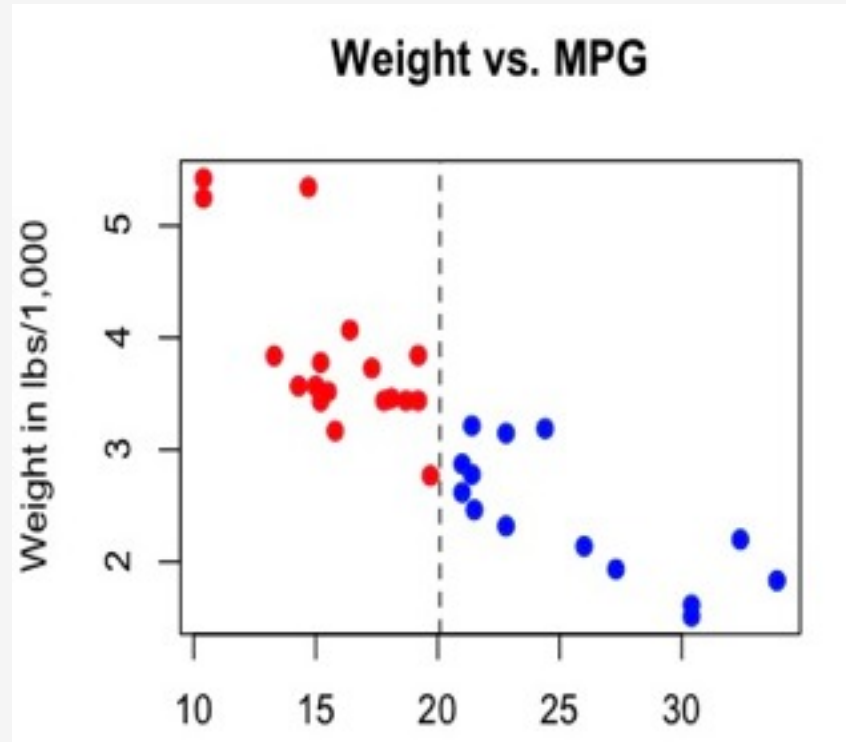
# Graphics: lattice: xyplot

So remember that we did this in BASE graphics to get a plot with points less than the mean MPG to be one color and the points greater than the mean to be another.

```
attach(mtcars)
plot(mpg,wt,type="n",
     main="Weight vs. MPG",
     ylab="Weight in lbs/1,000")


points(mpg[mpg > mean(mpg)],
       wt[mpg > mean(mpg)],
       col="blue",bg="blue",
       pch=21)

points(mpg[mpg <= mean(mpg)],
       wt[mpg <= mean(mpg)],
       col="red",bg="red",
       pch=21)

abline(v=mean(mpg),lty=2)
```



Weight vs. MPG

Tuesday, October 29, 13
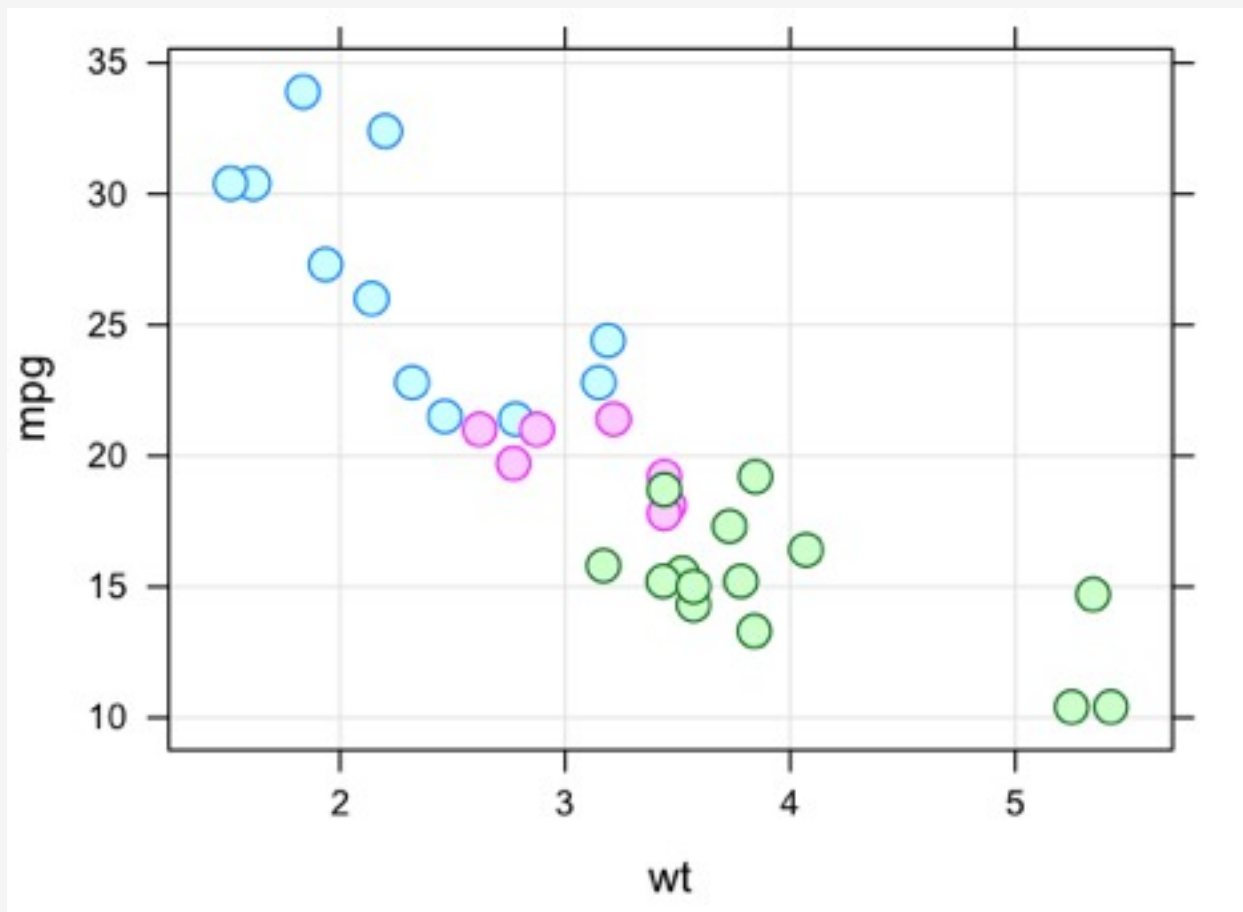
# Graphics: lattice: xyplot

So the two commands are basically the same:

```
xyplot(mpg~wt, data=mtcars, groups=factor(cyl),
    panel = function(x,y,...) {
      panel.xyplot(x,y, cex=1.5, pch=21,...)
      panel.grid()
    }
)

xyplot(mpg~wt, data=mtcars, groups=factor(cyl),
      cex=1.5, pch=21, type=c("p","g"))
```
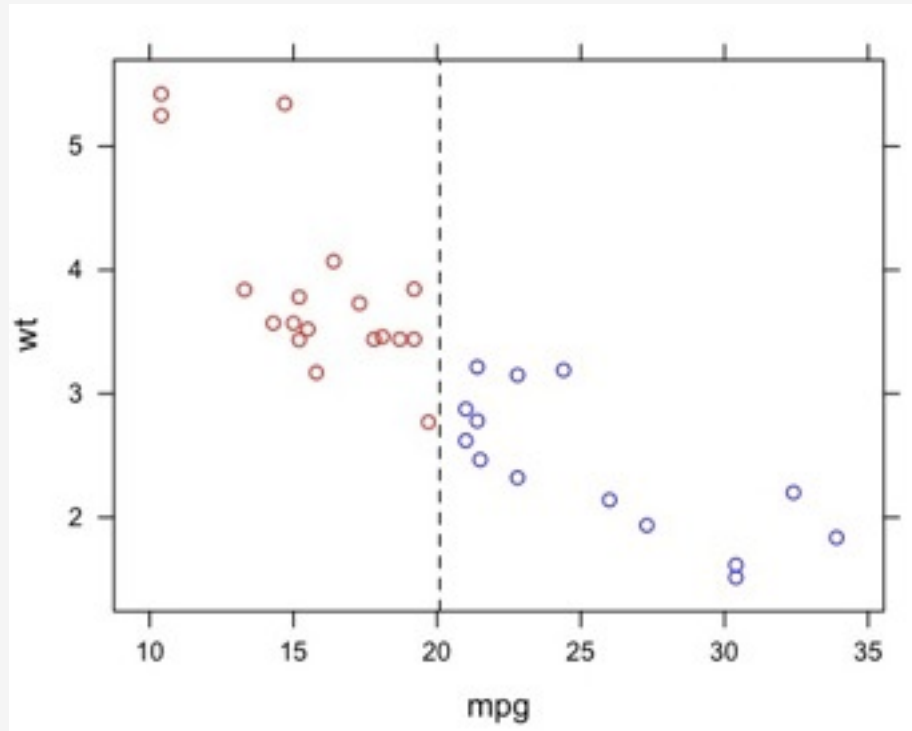
Tuesday, October 29, 13

# Graphics: lattice: xyplot

So the two commands are basically the same:

Tuesday, October 29, 13

# Graphics: lattice: xyplot

Here is one way to do it with lattice graphics:

```
xyplot(mpg ~ wt, mtcars,
   panel = function(x,y) {
     panel.xyplot(x[x <= mean(x)], y[x <= mean(x)], col="red")
     panel.xyplot(x[x  > mean(x)], y[x  > mean(x)], col="blue")
     panel.abline(v=mean(x),lty="dashed")
   }
)
```
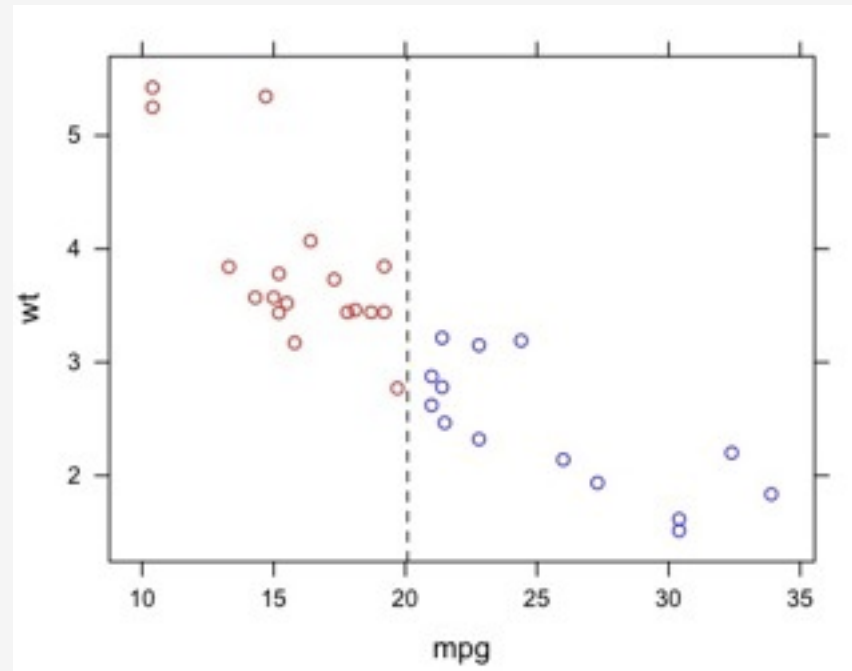
Tuesday, October 29, 13

# Graphics: lattice: xyplot

Or alternatively:

```
my.mean = ifelse(mtcars$mpg > mean(mtcars$mpg),"B","R")

xyplot(wt ~ mpg, mtcars, groups = my.mean, col=c("blue","red"),
    panel = function(...) {
        panel.abline(v=mean(mtcars$mpg), lty="dashed")
        panel.xyplot(...)
          }
)
```

Tuesday, October 29, 13

# Graphics: lattice: xyplot

The default panel function for the xyplot() function is panel.xyplot().  This is a "busy" example but it drives home the point that you can do a lot of customization within the panel function. You don't always HAVE to but its a good option to have.

```
xyplot(mpg ~ wt | factor(am,labels=c("auto","manual")), data=mtcars,
auto.key=TRUE,
        panel = function(x, y) {
          panel.grid(h = -1, v = 2)
          panel.xyplot(x, y)
          panel.loess(x, y, span=0.5, col='red')
          panel.lmline(x, y)
        }
      )
```
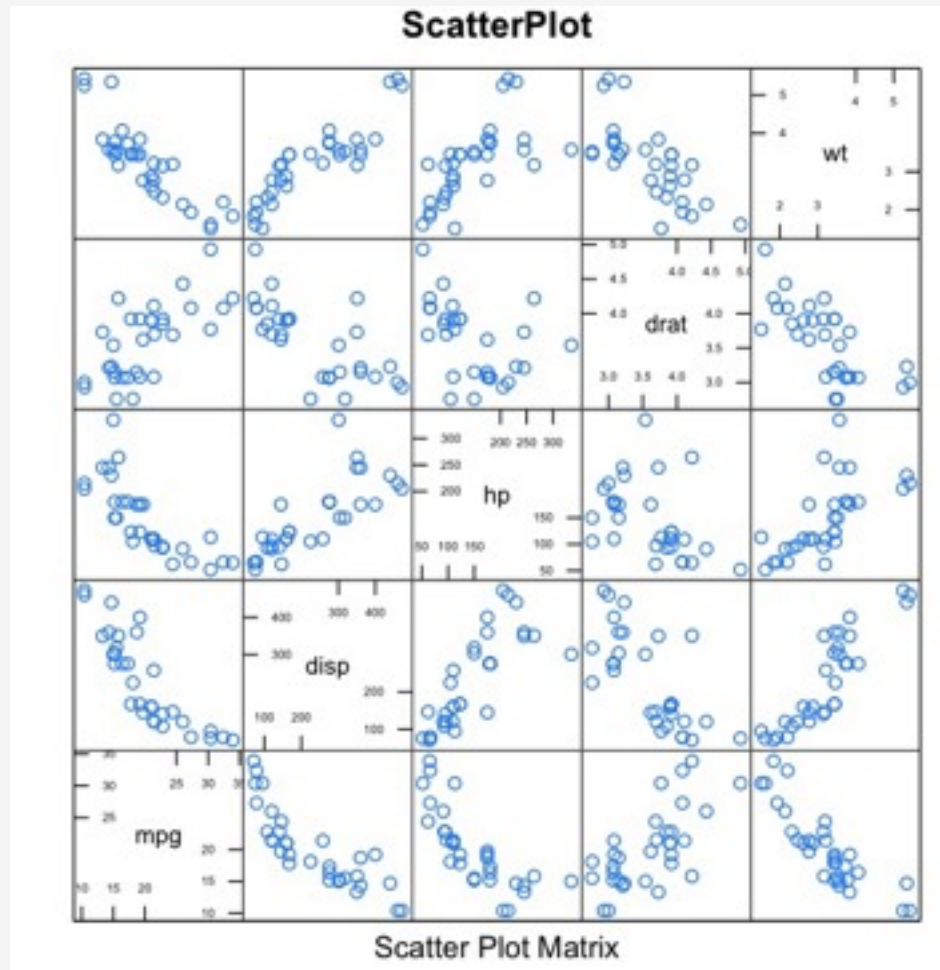
Tuesday, October 29, 13
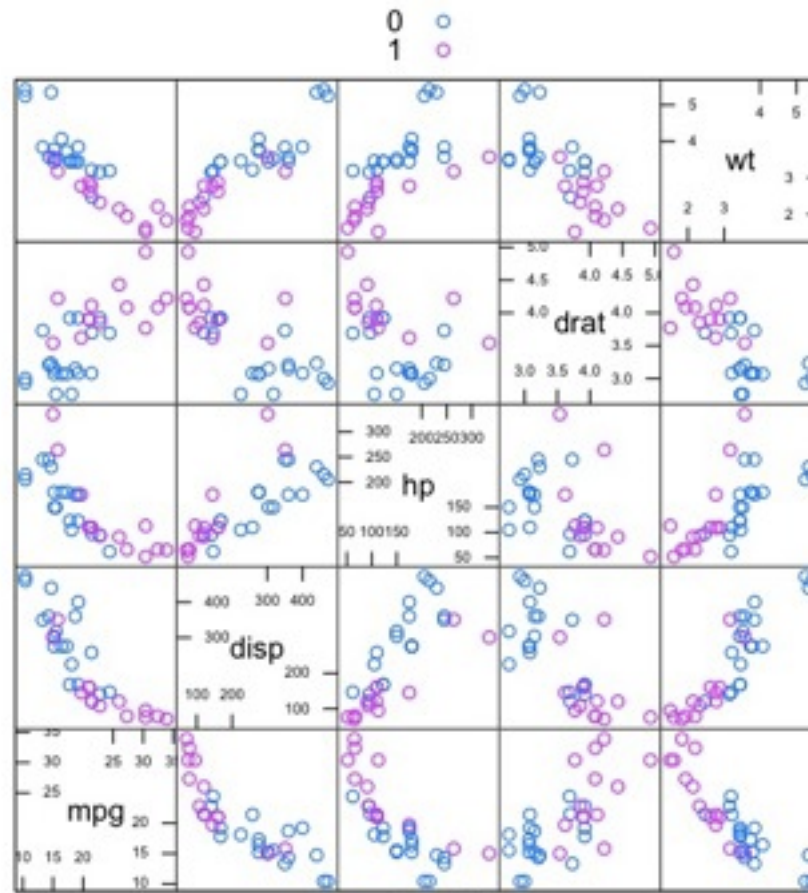
# Graphics: lattice: xyplot

Tuesday, October 29, 13
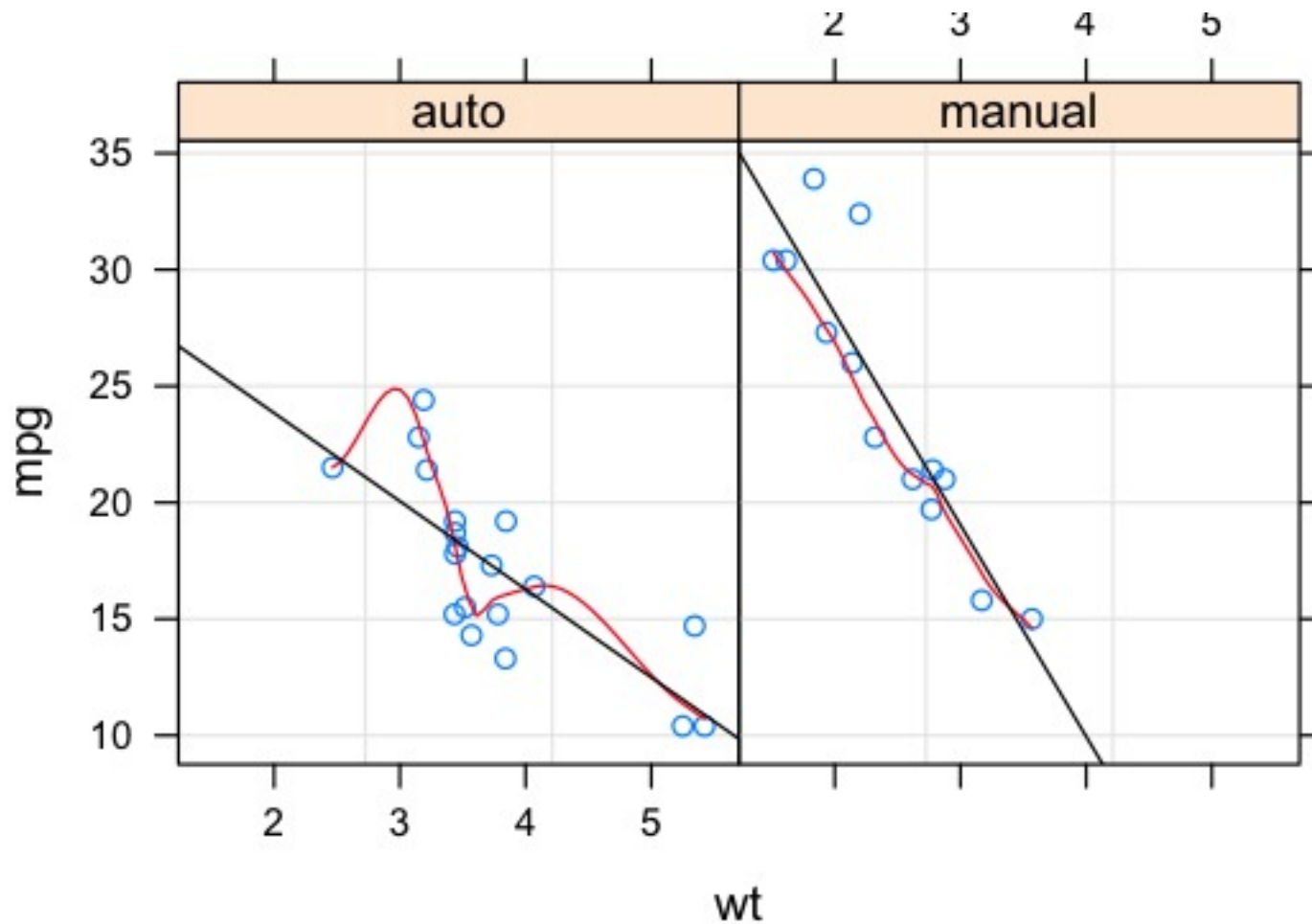
# Graphics: lattice: splom

```
splom(~mtcars[,c(1,3:6)], data=mtcars, main="ScatterPlot",
      varname.cex=.8,axis.text.cex=0.4)
```



**ScatterPlot**

Scatter Plot Matrix

Tuesday, October 29, 13

# Graphics: lattice: splom

```
splom(~mtcars[c(1,3:6)],groups = am, mtcars,
        auto.key = TRUE,axis.text.cex=0.5)
```



Scatter Plot Matrix

Tuesday, October 29, 13

# Graphics: lattice: splom

```
> splom(~mtcars[c(1:4)] |am,groups=am,mtcars,auto.key=TRUE,pscales=0)
```



Scatter Plot Matrix

Tuesday, October 29, 13
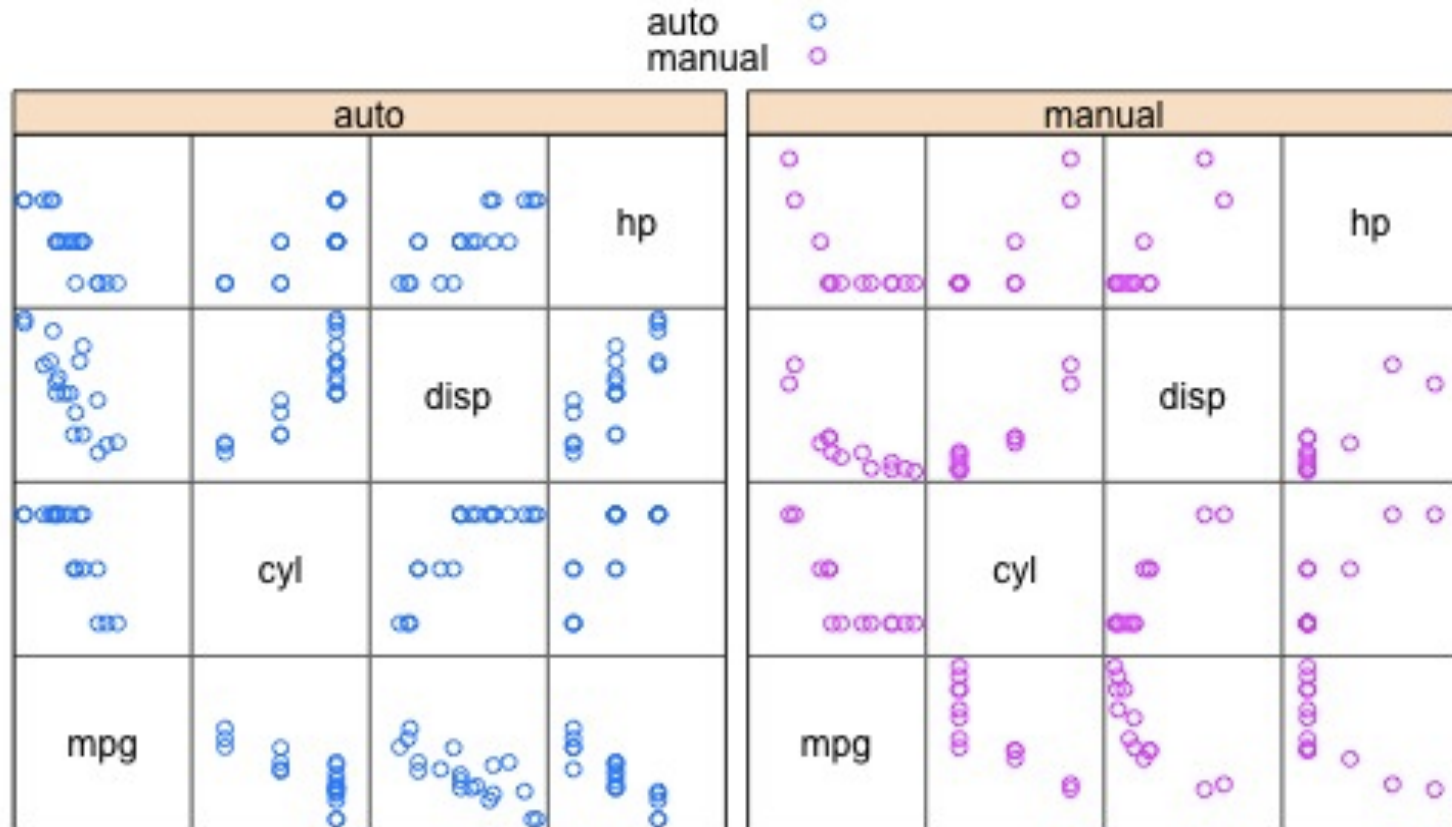
# Graphics: lattice: splom

```
splom(~data.frame(mpg, disp, hp, drat, wt, qsec),
      data = mtcars, groups = cyl, pscales = 0,
      varnames = c("Miles\nper\ngallon", "Displacement\n(cu. in.)",
                   "Gross\nhorsepower", "Rear\naxle\nratio",
                   "Weight", "1/4 mile\ntime"),
      auto.key = list(columns = 3, title = "Number of Cylinders"),
      varname.cex=0.7)
```

Tuesday, October 29, 13

# Graphics: lattice: splom

Tuesday, October 29, 13
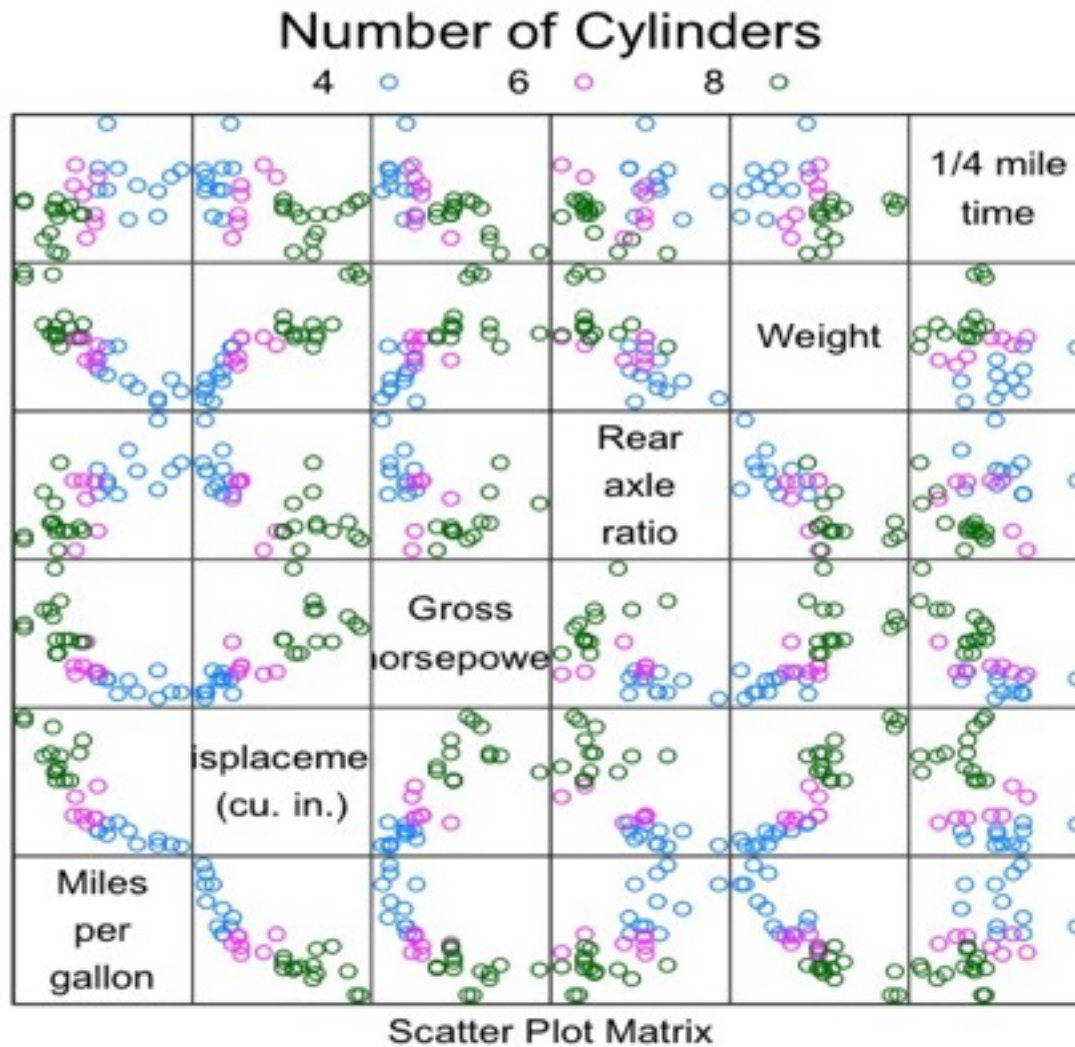
# Graphics: lattice: splom

```
splom(~iris[1:4], groups = Species, data = iris,auto.key=T)
```
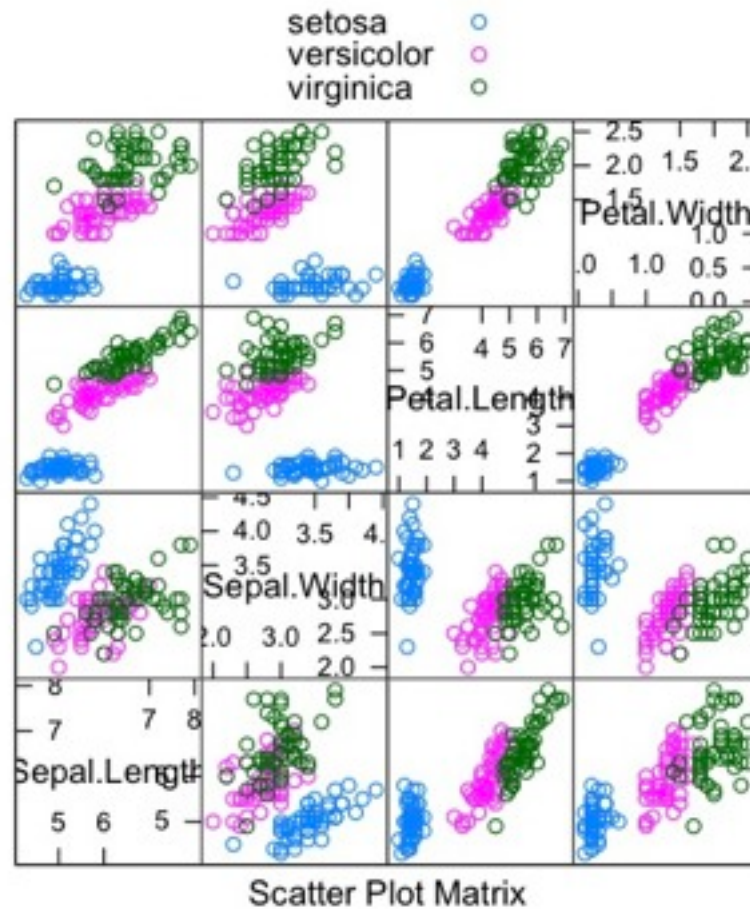


Scatter Plot Matrix

BIOS 560R - Graphics

Tuesday, October 29, 13

# Graphics: lattice

**plot(x,y) where x and y are continuous:**

X/Y, Scatterplot, Sunflower Plot

**plot(x,[y]) where x and y are categorical. Note that y can be optional:**

dotplot, barplot, stacked bar plot, pie chart

**plot(x) where x is a single continuous variable:**

dotplot, barplot, stripchart, boxplot, density, histogram, QQ Plot

**plot(x,y) where one of x and y is continuous and the other is discrete**

Side-by-Side DotPlot and BoxPlot, Notched BoxPlot

Tuesday, October 29, 13

# Graphics

Let's look at a different data set that is smaller. For small samples, summarizing is often unnecessary, and simply plotting all the data reveals interesting features of the distribution.

```
> head(quakes)
     lat    long depth mag stations
1 -20.42 181.62   562 4.8       41
2 -20.62 181.03   650 4.2       15
3 -26.00 184.10    42 5.4       43
4 -17.97 181.66   626 4.1       19
5 -20.42 181.96   649 4.0       11
6 -19.68 184.31   195 4.0       12

> sapply(quakes,class)
      lat      long     depth       mag  stations
"numeric" "numeric" "integer" "numeric" "integer"

> sapply(quakes,range)
        lat   long depth mag stations
[1,] -38.59 165.67    40 4.0       10
[2,] -10.72 188.13   680 6.4      132
```

Tuesday, October 29, 13

We surveyed people to see if they smoked ever, heavily, or previously. We were then given some idea of their socioeconomic status ("high","middle","low"). Let's check it out:

```
smokers = read.table("http://www.cyclismo.org/tutorial/R/
smoker.csv",header=T,sep=",")

head(smokers)
     Smoke  SES
1 former High
2 former High

( my.table = table(smokers$Smoke,smokers$SES) )

        High Low Middle
  current   51  43     22
  former    92  28     21
  never     68  22      9

> barchart(my.table,auto.key=list(space="right"),stack=F,horizontal=F)
```
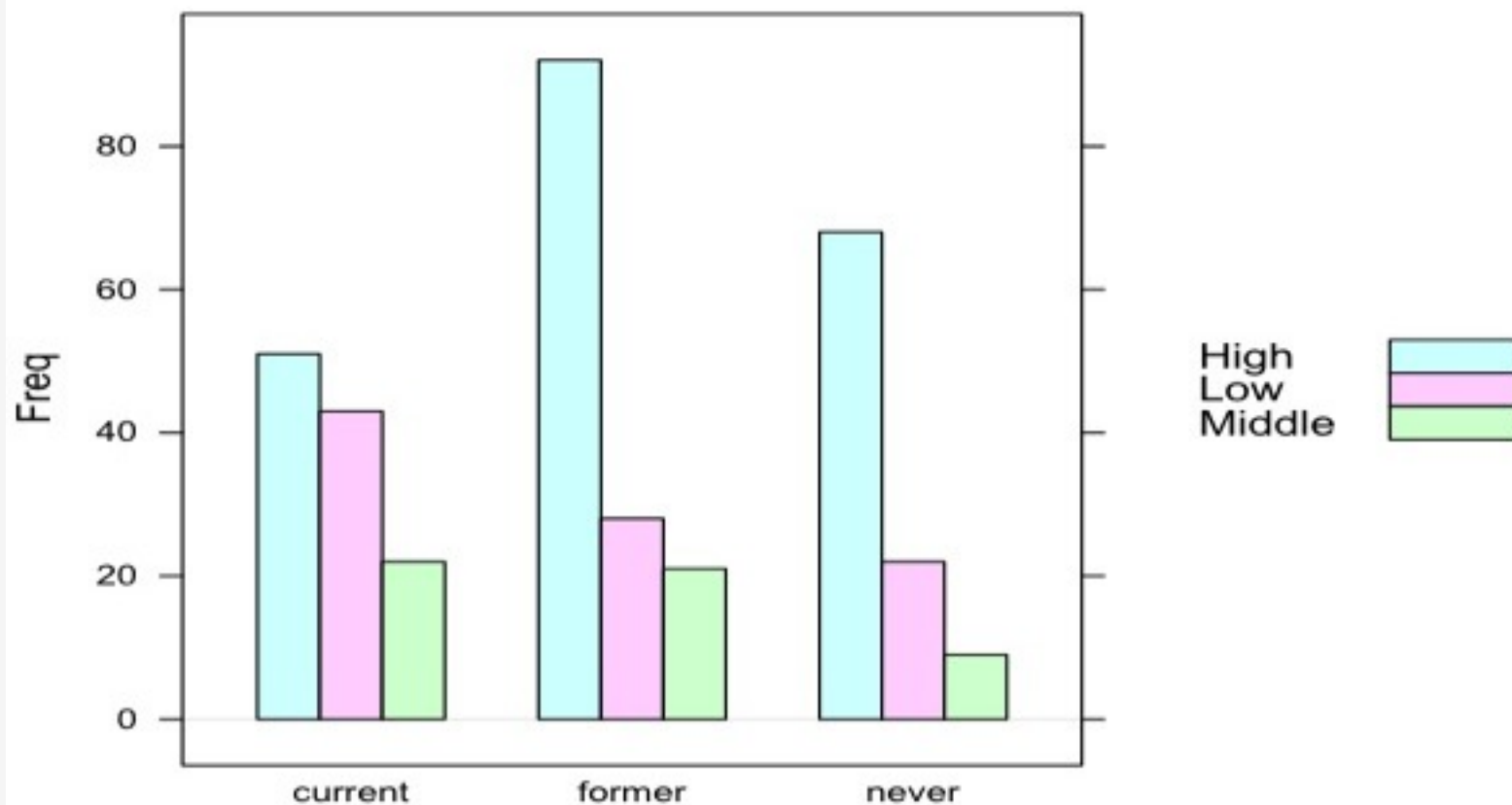
Tuesday, October 29, 13

# Graphics: lattice: barchart

Tuesday, October 29, 13
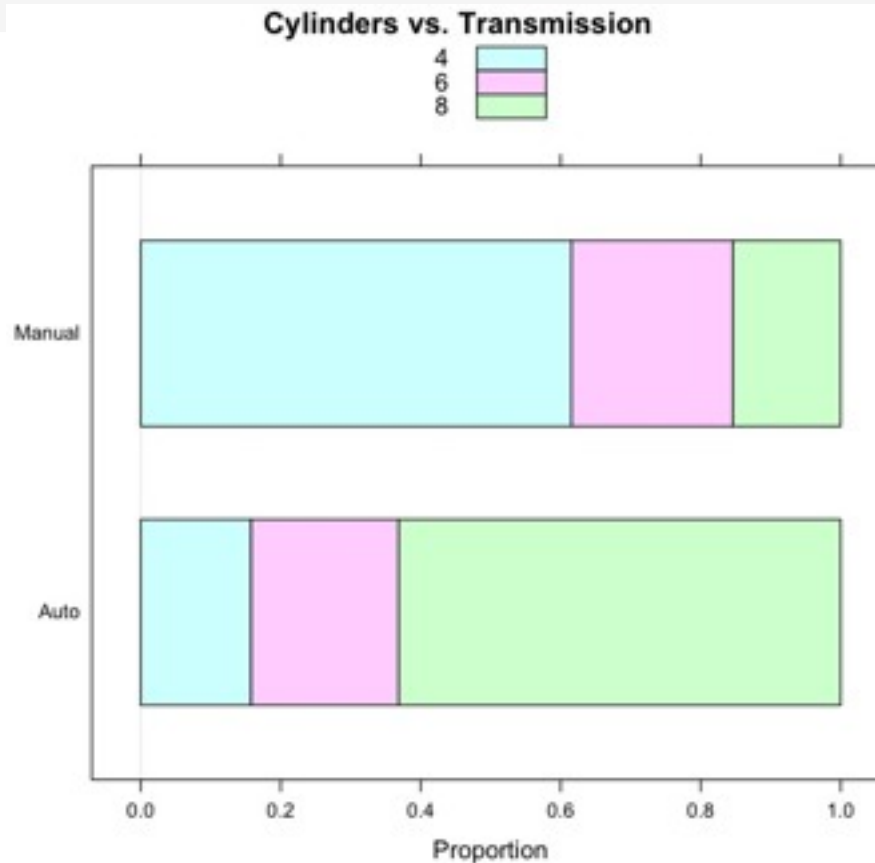
# Graphics: lattice: barchart

```
my.table = table(factor(mtcars$am,labels=c("Auto","Manual")),
            mtcars$cyl)

barchart(prop.table(my.table,margin=1),xlab="Proportion",
        auto.key = T, main = "Cylinders vs. Transmission")
```

Tuesday, October 29, 13

# Graphics: lattice

**plot(x,y) where x and y are continuous:**

    X/Y, Scatterplot, Sunflower Plot

**plot(x,[y]) where x and y are categorical. Note that y can be optional:**

    dotplot, barplot, stacked bar plot, pie chart

**plot(x) where x is a single continuous variable:**

    dotplot, barplot, stripchart, boxplot, density, histogram, QQ Plot

**plot(x,y) where one of x and y is continuous and the other is discrete**

    Side-by-Side DotPlot and BoxPlot, Notched BoxPlot

Tuesday, October 29, 13
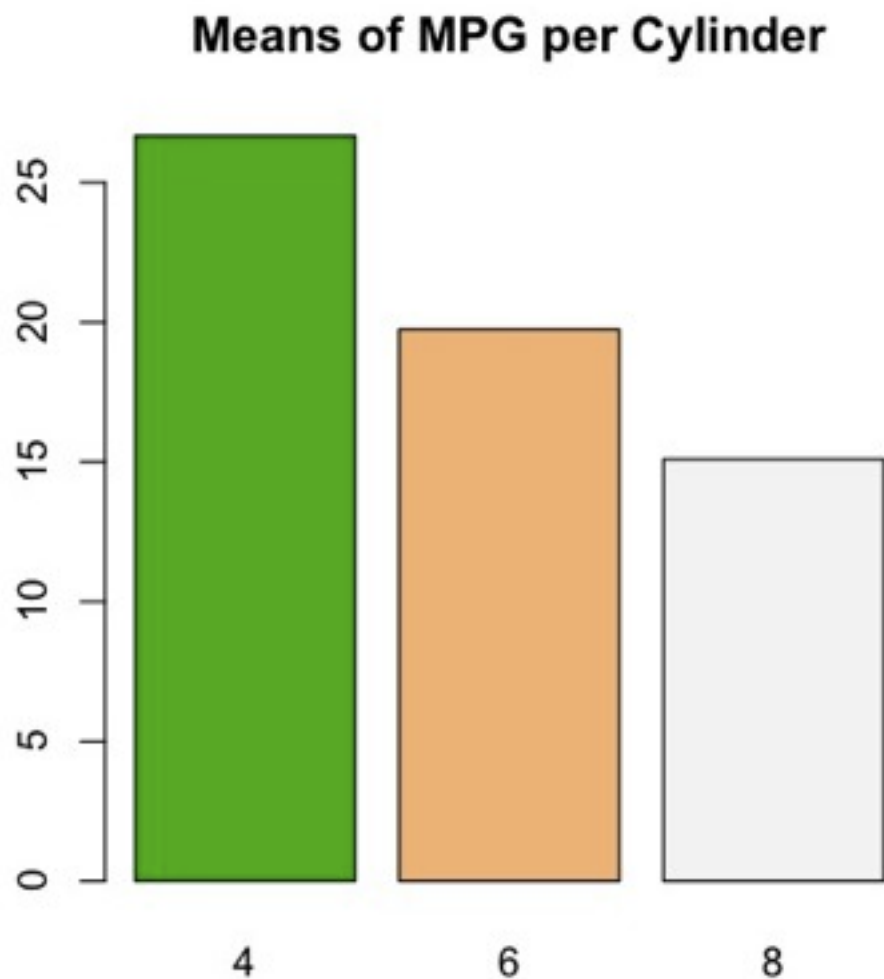
# Graphics: lattice: barchart

```
my.table = tapply(mtcars$mpg,mtcars$cyl,mean)
        4        6        8
26.66364 19.74286 15.10000


barplot(my.table, col=terrain.colors(3),
        main="Means of MPG per Cylinder")
```

Tuesday, October 29, 13

# Graphics: lattice: barchart

**Means of MPG per Cylinder**

Tuesday, October 29, 13

# Graphics: lattice: bwplot

A well-known graphical design that allows comparison between an arbitrary number of samples is the comparative box-and-whisker plot. They are related to the Q-Q plot: the values compared are five "special" quantiles, the median, the first and third quartiles, and the extremes.

More commonly, the extents of the "whiskers" are defined differently, and values outside plotted explicitly, so that heavier-than-normal tails tend to produce many points outside the extremes.

```
> bwplot(factor(score) ~ gcsescore | gender, Chem97,main="A-
level score vs. GCSES score \n by Gender")
```

The decreasing lengths of the boxes and whiskers suggest decreasing variance, and the large number of outliers on one side indicate heavier left tails (characteristic of a left-skewed distribution).

Tuesday, October 29, 13

# Graphics: lattice: bwplot

Next. Let's look at some other examples. In We'll be using the mtcars data set as well as the Chem97 dataset contained in the mlmRev package. If you don't have that package installed please install it using your GUI or at the command line by doing:

```
> install.packages("mlmRev")
```

Tuesday, October 29, 13

# Graphics: lattice: bwplot
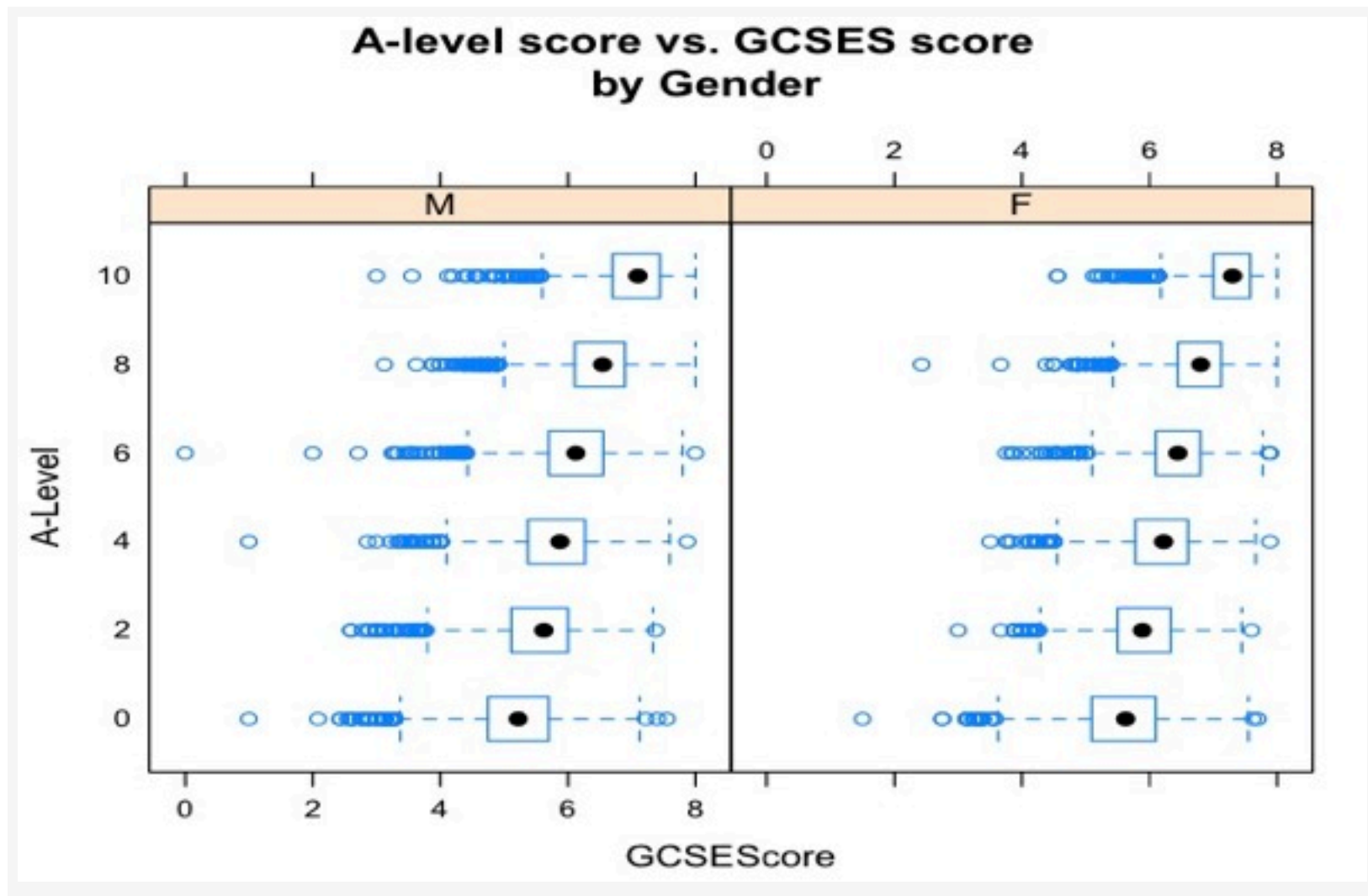
```
> data(Chem97, package="mlmRev")

> head(Chem97)
  lea school student score gender age gcsescore    gcsecnt
1   1      1       1     4      F   3     6.625  0.3393157
2   1      1       2    10      F  -3     7.625  1.3393157
3   1      1       3    10      F  -4     7.250  0.9643157
4   1      1       4    10      F  -2     7.500  1.2143157
5   1      1       5     8      F  -1     6.444  0.1583157
6   1      1       6    10      F   4     7.750  1.4643157
```

This dataset contains information in the 1997 A-level chemistry examination in Britain. Let's focus on the the following variables:

* score: point score in the exam, with six possible values (0,2,4,6,8)  (discrete)

* gcsescore: average score in GCSE exams, This is a continuous score that can be used to predict the A-level scores. (continuous)

* gender: Gender of the student  (discrete/binary)

Tuesday, October 29, 13

# Graphics: lattice: bwplot



A-level score vs. GCSES score by Gender

Tuesday, October 29, 13

# Graphics: lattice: bwplot

Note that we change the plot to get a different view. We do this by switching the order of the variables. The previous example looks like:
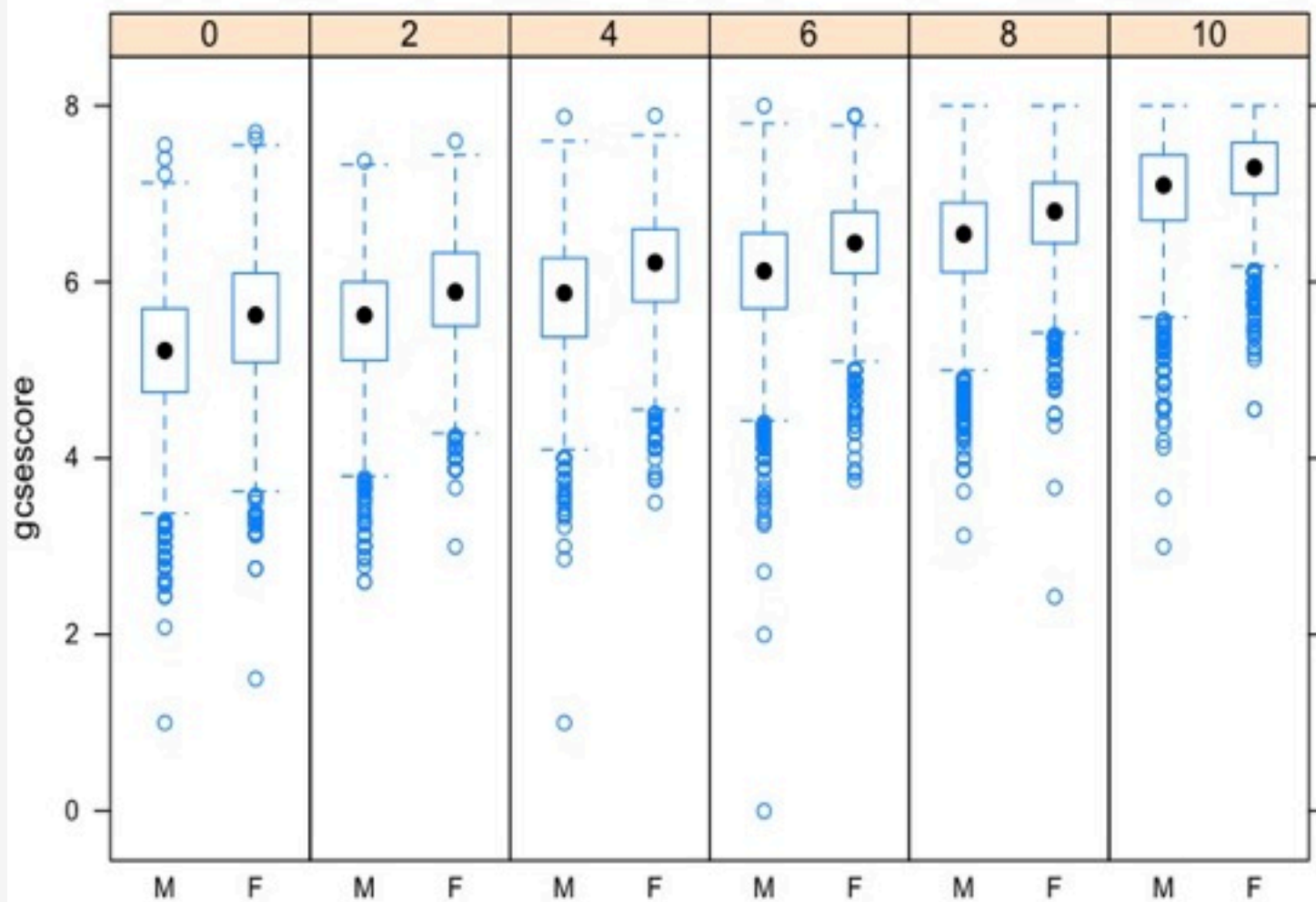
```
> bwplot(factor(score) ~ gcsescore | gender, Chem97,main="A-
level score vs. GCSES score \n by Gender")
```

Let's do this:

```
> bwplot(gcsescore ~ gender | factor(score), Chem97,
layout=c(6,1))
```
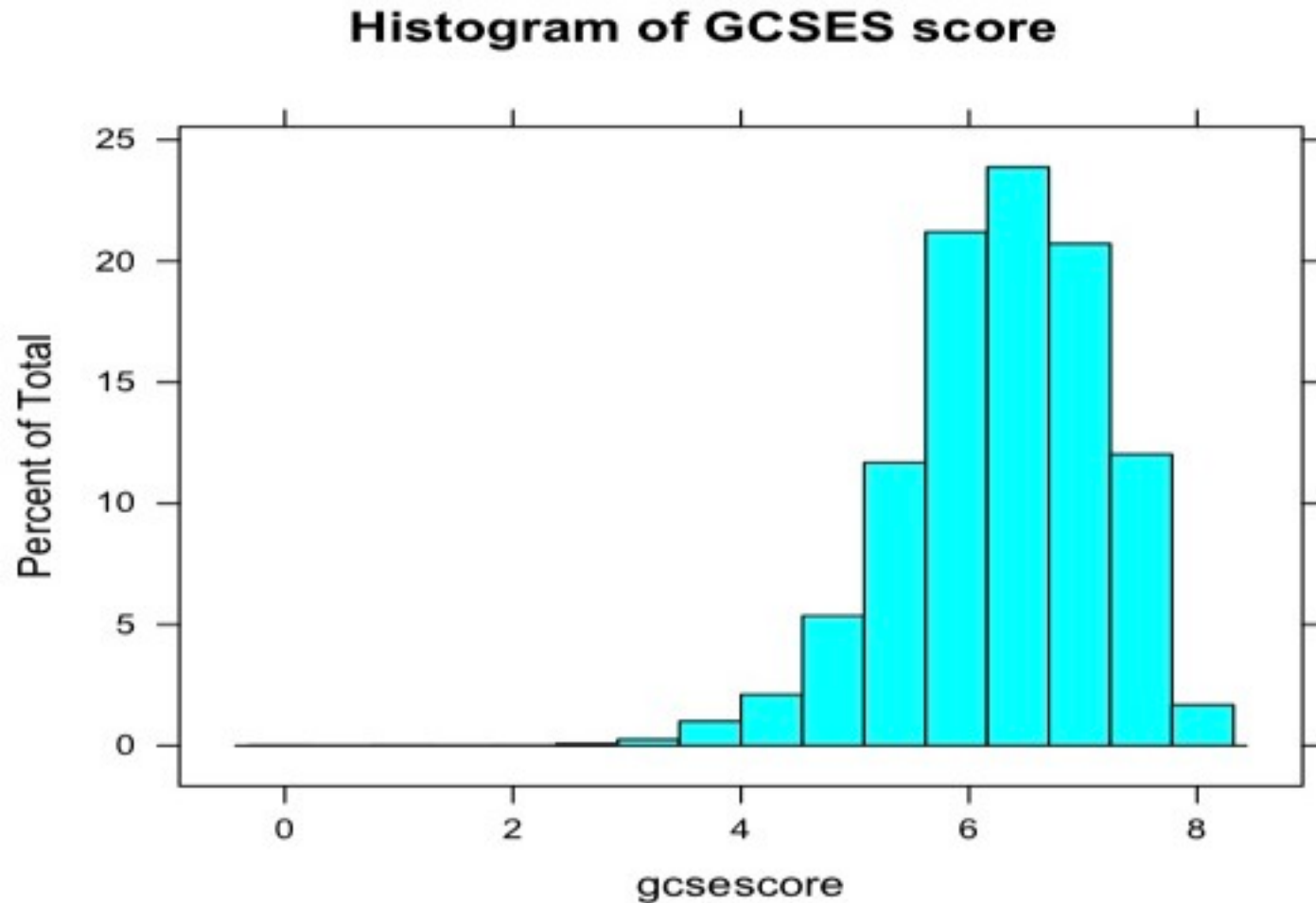
We get a different view of this information:

Tuesday, October 29, 13

# Graphics: lattice: bwplot

BIOS 560R - Graphics

Tuesday, October 29, 13

# Graphics: lattice: histogram

```
> histogram(~ gcsescore,data=Chem97,
+          main="Histogram of GCSES score")
```



**Histogram of GCSES score**

Tuesday, October 29, 13

# Graphics: lattice: histogram

Okay, that's cool but it is kind of boring and uninformative. It might be better to look at a histogram across the scores, which could be considered as a factor grouping.

```
> histogram(~ gcsescore | factor(score), data = Chem97)
```

Tuesday, October 29, 13
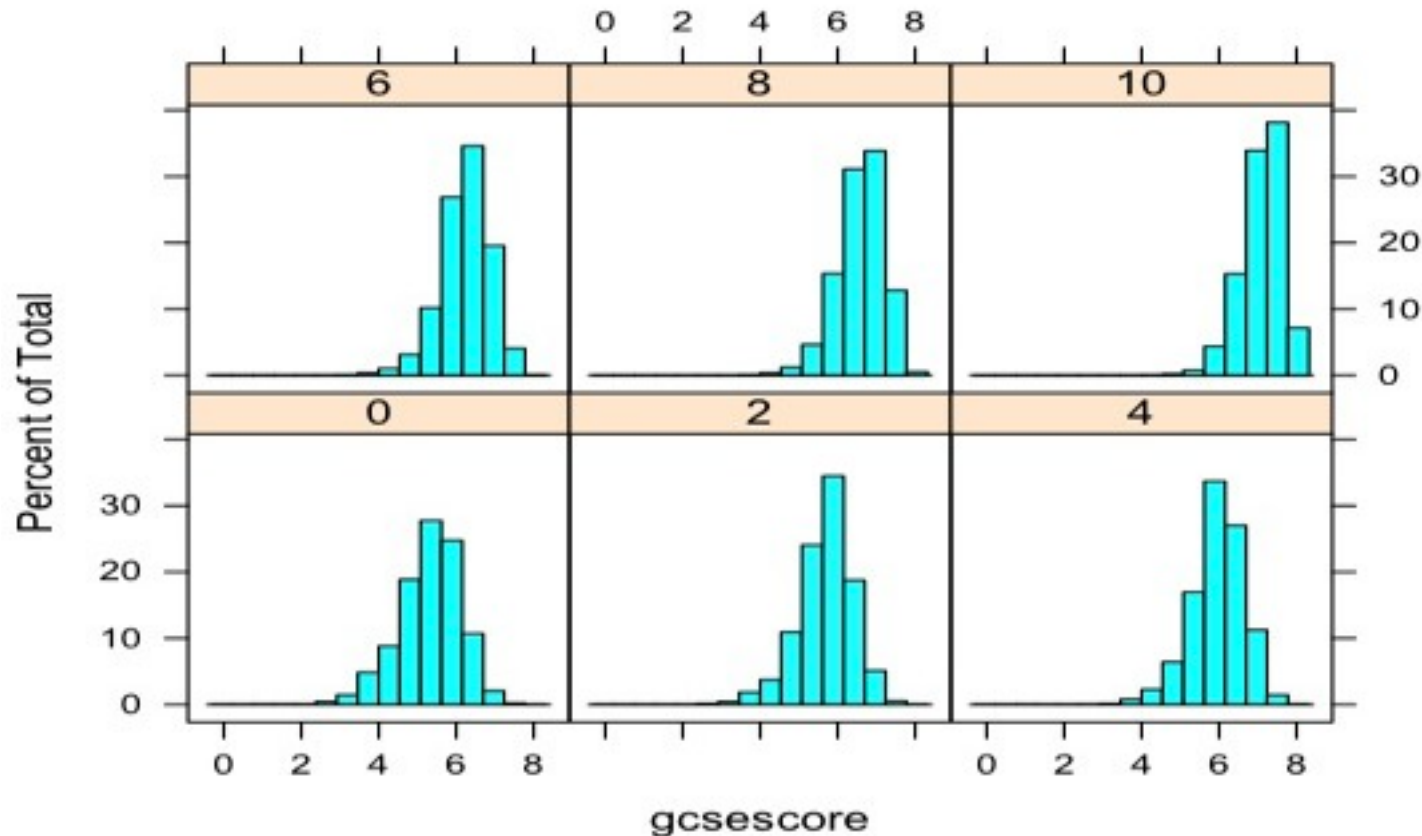
# Graphics: lattice: histogram

```
histogram(~ gcsescore | factor(score), data = Chem97,
     xlab = "Height (inches)", type = "density",
     panel = function(x, ...) {
          panel.histogram(x, ...)
          panel.mathdensity(dmath = dnorm, col = "black",
                               args = list(mean=mean(x),sd=sd(x)))
     })
```

Tuesday, October 29, 13

# Graphics: lattice: histogram

Tuesday, October 29, 13