**Coursera Data Science Course 4 Week 2**

**1. Lattice Plotting System**

**What is it?**-the following packages are used in the implementation  
**lattice**: contains code for producing Trellis graphics, which are independent of the “base” graphics system; includes functions like xyplot, bwplot, levelplot  
**grid**: implements a different graphing system independent of the “base” system; the lattice package builds on top of grid  
 -we usually don’t call grid directly  
-lattice isn’t a “two phase” aspect because ALL functions are done in a single call

**Lattice Functions  
xyplot** – this is the main function for creating scatterplots  
**bwplot** – box-and-whiskers plots (“boxplots”)  
**histogram** - histograms  
**stripplot** – like a boxplot but with actual points  
**dotplot** – plot dots on “violin strings”  
**splom** – scatterplot matrix; like pairs in base plotting system  
**levelplot, contourplot** – for plotting “image” data

**Lattice Functions**-take formula for first argument  
**xyplot(y ~ x | f \* g, data)**~ represents formula notation  
the left ~ is y-axis variable, the right is x-axis variable  
“I want to look at every variable of x and y on the levels or the “categorical variables” of f + g”  
f and g are conditioning variable (therefore are optional)  
 - \* is just interaction between two variables  
second argument is the data frame or list from which the variables in the formula should be looked up  
 - if nothing is passed, then parent frame is used  
-defaults are used if no other arguments

**Simple Lattice Plot  
library(lattice)  
library(datasets)  
xyplot(Ozone ~ Wind, data = airquality)** // open circles by default

Can also do  
**airquality <- transform(airquality, Month = factor(Month))  
xyplot(Ozone ~ Wind | Month , data = airquality, layout = c(5, 1))**  // lists out the months by 5 different columns

**Lattice Behavior**-base graphics plots data directly to graphics device  
-lattice graphics return an object of class trellis  
-the print methods for lattice functions actually do the work of plotting the data on the graphics device  
-lattice functions return “plot objects” that can, in principle be stored.. (but we should save the code and data)  
-on the command line, trellis objects are auto-printed so it appears the function is plotting the data  
tl;dr lattice returns object trellis, then print on the graph. but yeah in theory you can save the object

**P <- xyplot(Ozone ~ Wind, data = airquality)  
print(p)**  // have to type print(p) for p to actually print   
OR  
**xyplot(Ozone ~ Wind, data = airquality)** // automatically prints

**Lattice Panel Functions**-the panel functions controls what happens inside each panel of the plot  
-the lattice package comes with default panel functions, but you can supply your own if you want to customize what happens in every panel  
-panel functions receive the x/y coordinates of the data points in their panel (along with any optional arguments)

Ex:  
**set.seed(10)  
x <- rnorm(100)  
f <- rep(0:1, each = 50)  
y <- x + f – f \* x + rnorm(100, sd = 0.5)   
f <- factor(f, labels =c (“Group 1”, “Group 2”))** // labels the two panel group 1 and group 2 **xyplot(y ~ x | f, layout = c(2, 1))** // ends up plotting two panels

**Some Custom Panel Function  
xyplot(y ~ x | f, panel = function(x, y, …) {  
 panel.xyplot(x, y, …)** // this panel function will plot x and y as usual **panel.abline(h = median(y), lty = 2)** // this panel function will add a horizontal line that is equal to the median of y, with the line type being 2, which is a dash line **})**

**Can also do Regression line  
xyplot(y ~ x | f, panel = function(x, y, …) {  
 panel.xyplot(x, y, …)   
 panel.lmline(x, y, col = 2)** // some simple regression line is added with the color 2  
**})**

**Example**: Mouse Allergen and Asthma Cohort Study (MAACS) for children with asthma B’more  
-> Use a many panel lattice plot, allows us to use a small function and creates a really huge plot!

Benefits of Lattice  
-margins and spacings are automatically handled  
-lattice system is ideal for creating conditioning plots

**2. ggplot2 Plotting System**

**What is it?**“grammar of graphics”  
think about “verb”, “noun”, “adjective” for graphics  
This “grammar” means that a statistical graphic is a mapping from data to aesthetic attributes (colour, shape, size) of geometric objects (points, lines, bars).

**qplot()**-similar to plot function in base graphics system  
-looks for data in a **data frame**, similar to lattice, or in the parent environment  
-plots are made up of aesthetics and geoms

-**factors** are important for indicating subsets of data (i.e. male or female) and should be labeled  
-qplot() hides what goes on underneath for ggplot()  
-ggplot() is a lot more flexible

Example dataset  
**library(ggplot2)  
str(mpg)** // miles per gallon data for different cars

**The most simple ggplot2**

**library(ggplot2)  
qplot(displ, hwy, data = mpg)  
qplot(displ, hwy, data = mpg, color = drv)** // automatically adds the color+legend placement, different types of drv variables will be assigned different colors

Can also add geoms: **qplot(displ, hwy, data = mpg, geom = c(“point”, “smooth”))**

**Histograms: qplot(hwy, data = mpg, fill = drv)**

**Facets “panels”:   
qplot(displ, hwy, data = mpg, facets = .~ drv)** // different types of drv variables will be put in different panels. The . for .~drv indicate that there’s only supposed to be one row for the plot and the drv in .~drv indicates the columns for the panels  
**qplot(hwy, data = mpg, facets = drv~., binwidth = 2)** // this time 3 rows and no columns

**Ex:** Back to Mouse Allergens..

**Qplot(log(eno), data = maacs)** // basic histogram for eNO, one of the variables  
**qplot(log(eno), data = maacs, fill = mopos)** // now two separate histograms are made that overlap each other via mopos as a separator

**Can also check density  
qplot(log(eno), data = maacs, geom = “density”)   
qplot(log(eno), data = maacs, geom = “density”, color = mopos)** // can see that the two peaks each belong to a different group, the non-mopos group had a smaller peak

**Scatterplots: e NO vs PM2.5 (comparing the two variables to see if related)**

**qplot(log(pm25), log(eno), data = maacs)  
qplot(log(pm25), log(eno), data = maacs, shape = mopos)  
qplot(log(pm25, log(eno), data = maacs, color = mopos)**

Can also “smooth” the relationship  
**qplot(log(pm25), log(eno), data = maacs, color = mopos) + geom\_smooth(method = “lm”)** // can see that no mopos has negative relationship, while yes mopos looks like a positive relationship, you can also separate the two…:  
**qplot(log(pm25), log(eno), data = maacs, facets = .~mopos) + geom\_smooth(method = “lm”)**

**Basic Components of a ggplot2 Plot**

-**a data frame**  
-**aesthetic mappings:** how data are mapped to color / size  
-**geoms:** geometric objects like points, lines, shapes  
-**facets:** for conditional plots  
-**stats:** statistical transformations like binning, quantiles, smoothing  
-**scales:** what scale an aesthetic map uses (ex: male = red, female = blue)  
-**coordinate system**

**How to build a plot with ggplot2**-think about it as “artist’s palette” again  
-plots are built in layers  
 - plot the data  
 - overlay a summary  
 - metadata and annotation

Example: BMI, PM2.5, Asthma  
-Question answered: Does BMI modify the relationship between PM2.5 and asthma symptoms?  
**qplot(logpm25, NocturnalSympt, data = maacs, facets = . ~ bmicat, geom = c(“point”, “smooth”), method = “lm”)** // two facets are created and we can see that for normal weight, there seems to be no correlation between nocturnal symptoms and logpm25, but however for overweight, logpm25 positively correlates with nocturnal symptoms

**No Plot Yet!  
g <- ggplot(maacs, aes(logpm25, NocturnalSympt))  
print(g)** // will return no layers in plot

**P <- g + geom\_point()  
print(p)** // will return a plot after adding the points

**G + geom\_point()** // auto prints without saving

**Add more layers: Smooth  
g + geom\_point() + geom\_smooth()** // does make data smoother, but not very smooth because theres not much data and thus a lot of noise **g \_ geom\_point() + geom\_smooth(method = “lm”)** //simple linear model makes it smoother

**Add more layers: Facets  
g + geom\_point() + facet\_grid(. ~ bmicat) + geom\_smooth(method = “lm”)** // compare the normal weights and the overweights   
Labels come from facet variable, so name them appropriately in the first place. Make sure data is labeled the right way BEFORE plotting.

**Annotation**Labels: xlab(), ylab(), labs(), ggtitle()  
Each of the “geom” functions has options to modify  
-for things that only makes sense globally, use theme()  
 Ex: theme(legend.position = “none”)  
Two standard appearance themes are included  
 - theme\_gray(): the default theme (gray background)  
 - theme\_bw(): more stark/plain

**Modifying Aesthetics  
g + geom\_point(color = “steelblue”, size = 4, alpha = ½)** // constant **g + geom\_point(aes(color = bmicat), size = 4, alpha = ½)** // depends on what the color of bmicat is in the first place

**Modifying Labels  
g + geom\_point(aes(color = bmicat)) + labs(title = “MAACS Cohort”) + labs(x = expression(“log “ \* PM[2.5]), y = “Nocturnal Symptoms)**

**g + geom\_point(aes(color = bmicat), size = 2, alpha = ½) + geom\_smooth(size = 4, linetype = 3, method = “lm”, se = FALSE)**

**Can also change overall theme  
g + geom\_point(aes(color = bmicat)) + theme\_bw(base\_family = “Times”)**

**Axis Limits (have to be careful!)  
testdat <- data.frame(x = 1:100, y = rnorm(100))  
testdat[50, 2] <- 100** // make ur own outlier **plot(testdat$x, testdat$y, type = “1”, ylim = c(-3, 3))**

**g <- ggplot(testdat, aes(x = x, y = y))  
g + geom\_line()**

**g + geom\_line() + ylim(-3 , 3)** // outlier will be missing  
**g + geom\_line() + coord\_cartesian(ylim = c(-3, 3))** // will include outlier

**ex:** How does the relationship between PM2.5 and nocturnal symptoms vary by BMI and NO2. But NO2 is continuous. So we’ll have to cut them

**Making NO2 Tertiles**

## Calculate the deciles of the data  
**cutpoints <- quantile(maacs$logno2\_new, seq(0, 1, length = 4), na.rm = TRUE)**

## Cut the data at the deciles and create a new factor variable  
**maacs$no2dec <- cut(maacs$logno2\_new, cutpoints)** //becomes three categories

## See the levels of the newly created factor variable  
**levels(maacs$no2dec)**

**Final Plot:**  Creates 8 plots by splitting logno2 , the last two are the missing data  
//added things like transparent points, smoother, multiple panels, labels, non-default font etc

Code:

**g <- ggplot(maacs, aes(logpm25, NocturnalSympt))**

**LAYERS:  
g + geom\_point(alpha = 1/3)** // add points  
 **+ facet\_wrap(bmicat ~ no2dec, nrow = 2, ncol = 4)**  // make panels  
 **+ geom\_smooth(method=”lm”, se=FALSE, col =”steelblue”)** // add smoother  
 **+ theme\_bw(base\_family = “Avenir”, base\_size = 10)** // changed font  
 **+ labs(x = expression(“log “ \* PM[2.5])** // two more lines for y and title to add labels