**Coursera Data Science Course 4 Week 1**

**Graphs**

**1. Principles of Analytic Graphics**

**Some Basic Rules ☺**

**Principle 1. Show comparisons**

-Evidence for a hypothesis is relative to another competing hypothesis.  
-“compared to what?”  
Example: Air cleaner affecting asthma children  
-“symptom free day”, has a mean of increase of asthma free day. This is in comparison to not having an air cleaner (that’s our control). It’s ALWAYS important to show some kind of control

**Principle 2: Show causality, mechanism, explanation, etc**

What’s the causal framework?  
Example: show that there’s a decrease of fine particulate matter with the air cleaner, therefore decreasing asthma symptoms in children

**Principle 3: Show multivariate data**

-there are definitely more than 2 variables in the real world. Can’t just have x and y all the time  
-i.e. can also look at air pollution level and mortality rates, can also look at different seasons  
-now we can see within each season, air pollution and mortality is slightly positive, but overall, it’s actually slightly negative. This is known as symptom paradox.

**Principle 4: Integration of evidence**

-completely integrate words, numbers, images, diagrams  
-as many modes of data presentation as possible

**Principle 5: Describe and document the evidence with appropriate labels, scales, sources, etc**

-a data graphic should tell a complete story that’s credible  
-source your stuff

**Principle 6: Content is king**

**2. Exploratory Graphs**

**Why graphs?**-understand data properties  
-patterns are easy to spot  
-modeling strategies  
-“debug” analyses  
-communicate results

**Exploratory graphs**  
-made quickly and a large number are made  
-for personal understanding of data  
-axes/legends are generally cleaned up (later)  
-color/size are primarily used for info

**Example**: Air pollution from EPA. Check for fine particle pollution and see if any county exceeds the national standard.. (downloading csv file)

**Pollution <- read.csv(“data/avgpm25.csv”, colClasses = c(“numeric”, “character”, “factor”, “numeric”, “numeric”))**

Always have some background questions kept in mind

**Simple Summaries of Data**

**1. Five-number summary – summary(pollution$pm25)**shows Min, 1st quantile, median, mean, 3rd quantile, and max (so actually 6 numbers)

**2. boxplots** **– boxplot(pollution$pm25, col = “blue”)  
abline(h = 12)** // can set a horizontal line at 12 because that’s the national ambient air quality standard, and we can see how many counties are above and below the line.

**3. Histogram – hist(pollution$pm25, col = “green”)** // can also add breaks = 100 to get a rougher histogram. Breaks are the number of bars going into the histograms. **rug(pollution$pm25)** // the rug shows distribution at the bottom of the graph by coloring something deeper vs brighter  
**abline(v = 12, lwd = 2)** // sets a vertical line at 12  
**abline(v = median(pollution$pm25), col = “magenta”, lwd = 4)** // sets a vertical line at the median, and sets the color to magenta

**4. density plots**

**5. Barplot – barplot(table(pollution$region), col = “wheat”, main = “Number of Counties in Each Region”)** // allows for showing the number of counties in east or west

**Two dimensions**

**Multiple/overlayed 1-D plots (Lattice, ggplot2)  
Scatterplots  
Smooth scatterplots**

**>2 dimensions**

**Overlayed/multiple 2-D plots; coplots  
Use color, size, shape to add dimensions  
Spinning plots  
Actual 3-D plots (which are not that useful….)**

**Multiple boxplots!  
boxplot(pm25 ~ region, data = pollution, col = “red”)** // will create two boxplots, one for each region

**Multiple Histograms!  
Par(mfrow = c(2,1), mar = c(4, 4, 2, 1))  
hist(subset(pollution, region == “east”)$pm25, col = “green”)  
hist(subset(pollution, region == “west”)$pm25, col = “green”)**

**Scatterplot  
with(pollution, plot(latitude, pm25))** // shows where the combination of pm25 and latitude is the most evident, you can also add colors for the regions ☺ **abline(h = 12, lwd = 2, lty = 2)**

**Multiple Scatterplots  
par(mfrow = c(1, 2), mar = c(5, 4, 2, 1))  
with(subset(pollution, region == “west”), plot(latitude, pm25, main = “West”))  
with(subset(pollution, region == “east”), plot(latitude, pm25, main = “East”))**

**3. Plotting ☺**

**Base Plotting System**

-blank canvas and add things.  
-start with plot function  
-use annotation functions to add/modify (text, lines, points, axis)  
-convenient, intuitive  
-can’t go back, can add but can’t take away additions  
-difficult to “translate” once a new plot has been created (no systematic)  
-plot is just a series of R commands

**Library(datasets)  
data(cars)  
with(cars, plot(speed, dist))**

**The Lattice System**-plots created with a single function call (xyplot, bwplot, etc)  
-most useful for conditioning types of plots – looking at how y changes w/ x across levels of z  
-things like margins/spacing set automatically because the entire plot is specified at once  
-good for putting many many plots on a screen  
BUT….  
-sometimes awkward to specify the entire plot in a single function call  
-annotation in plot is not intuitive  
-use of panel functions and subscripts difficult to wield and requires intense preparation  
-cannot “add” to the plot once it is created

**Library(lattice)  
state <- data.frame(state.x77, region = state.region)  
xyplot(Life.Exp ~ Income | region, data = state, layout = c(4, 1))** // splitting into four quadrants

**The ggplot2 System**-splits the difference between base and lattice in a number of ways  
-automatically deals with spacings, text, titles, but also allows u to annotate by “adding” to a plot  
-superficial similarity to lattice but generally easier/ more intuitive to use  
-default mode makes many choices for you (but still CUSTOMIZABLE)

**Library(ggplot2)  
data(mpg)  
qplot(displ, hwy, data = mpg)**

**The Process of Making a Plot**-Use base, lattice, or ggplot2  
-base graphics are constructed piecemeal, each aspect of the plot handled separately through some function calls – conceptually simple  
-lattice graphics are created using a single function- ALL PARAMETERS have to be specified at once, allows R to automatically calculate the necessary spacings and font sizes.  
-ggplot2 combines concepts from both

**Base Graphics (2-D)**-two phases to create a base plot.  
 1. Initialize a new plot  
 2. Annotating an existing plot  
-plot(x,y) or hist(x) launches a graphics device and draw a new plot on the device  
-arguments to plot are not of some special class, then the default method of plot is called. There are A LOT of arguments including setting the title, x and y axis labels, etc.  
-base graphics system has many parameters that can set and tweaked- these parameters are documented in ?par (MEMORIZE THAT HELP PAGE!)

**Histogram  
library(datasets)  
hist(airquality$0zone)** // draws new plot

**Scatterplot  
library(datasets)  
with(airquality, plot(Wind, 0zone))**

**Boxplot  
library(datasets)  
airquality <- transform(airquality, Month = factor(Month))  
boxplot(Ozone ~Month, airquality, xlab = “Month”, ylab = “Ozone (ppb)”)**

**Important Base Graphics Parameters to Know**

**Pch** – plotting symbol (default = open circle)  
**lty** - line type (default = solid line)  
**lwd** – line width (specified with integer)  
**col** – the plotting color, specified as a number, string, or hex code (check colors() function)  
**xlab** – character string for x-axis label   
**ylab** – character string for y-axis label

**More Important Base Graphics Parameters**

**Las** – the orientation of the axis labels on the plot  
**bg** – the background color  
**mar** – the margin size  
**oma** – the outer margin size (default is 0 for all sides)  
To fit multiple plots:  
**mfrow** – number of plots per row, column (plots are filled row-wise)  
**mfcol** – number of plots per row, column (plots are filled column-wise)

**To check the default value:  
par(“lty”)** // returns “solid”

**Base Plotting Functions**

**Plot** – make a scatterplot, or other type of plot depending on the class of object plotted  
**lines** – add lines to a plot, given a vector x values and a corresponding vector of y values, this function connects the dots  
**points** – add points to a plot  
**text** – add text labels to a plot using specified x, y coordinates  
**title** – add annotations to x, y axis labels, title, subtitle, outer margin  
**mtext** – add arbitrary text to the margins (inner or outer) of the plot  
**axis** – adding axis ticks/labels

**Library(datasets)  
with(airquality, plot(Wind,Ozone))  
title(main = “Ozone and Wind in New York City”)** // adds title to the scatter plot

**Annotations  
With(airquality, plot(Wind, Ozone, main = “Ozone and Wind in New York City))** // can already add certain features like the title **with(subset(airquality, Month == 5), points(Wind, Ozone, col = “blue”))** //use the 5th month of data and turns the points into blue  
**with(subset(airquality, Month != 5), points(Wind, Ozone, col = “red”))**  // use the non-5th month of data and turns the points into red  
**legend(“topright”, pch = 1, col = c(“blue, “red”), legend = c(“May”, “OtherMonths”))** // adds a legend to the top right of the graph

**Regression Line  
with(airquality, plot(Wind, Ozone, main = “Ozone and Wind in New York City”, pch = 20))  
model <- lm(Ozone ~ Wind, airquality), abline(model, lwd = 2)**

**Multiple Base Plots  
par(mfrow = c(1, 2))  
with (airquality, {  
 plot(Wind, Ozone, main = “Ozone and Wind”)  
 plot(Solar.R, Ozone, main = “Ozone and Solar Radiation”)  
})**

**Mtext(“Ozone and Weather in New York City”, outer = TRUE)** // creates a label that covers the whole plot

Summary: Easy to use and lots of control

Check out “Base Plotting Demonstration” in week 1 for how to use basic devices such as margins and plot  
**plot(x, y, type = “n”)** // don’t actually put x and y in the plot  
if we want to make the two groups two different colors  
plot each group separately and color them before adding the other group

**4. Graphics Device**-anything that makes a plot appear  
-a window on computer, PDF, JPEG, scalable vector graphics (SVG) file  
-a plot in R has to be “sent” to a specific graphics device  
-most common place for a plot to be “sent” is the screen device (**windows()**)

-check **?Devices** for a list of devices  
-quick visualizations and exploratory analysis -> use screen device  
-plot in base, xyplot in lattice, qplot in ggplot2 will default to sending a plot to the screen device  
-plots that may be printed out/incorporated into a document -> use file device  
-a lot of different file devices…

**Two ways to create a plot**  
1.  
 -> i. call plotting function (plot, xyplot, qplot)  
 -> ii. Plot appears on screen device  
 -> iii. Annotate plot if necessary

2.   
 -> i. explicitly launch a graphics device  
 -> ii. Call a plotting function in make a plot (Note: if you’re using a file device, no plot will appear on the screen)  
 -> iii. Annotate plot  
 -> iv. Remember to EXPLICITLY CLOSE graphics device with dev.off()  
ex:  
**pdf(file = “myplot.pdf”)** // open PDF device and create ‘myplot.pdf’ in my working directory **with(faithful, plot(eruptions, waiting))** // create plot and send to a file **title(main = “Old Faithful Geyser data”)  
dev.off()**The file “myplot.pdf” will be created on computer

**Graphic File Devices  
pdf** – useful for line-type graphics, resizes well, usually portable, not efficient if a plot has many objects/points  
**svg** – XML-base scalable vector graphics; supports animation and interactivity, potentially useful for web-based plots  
**win.metafile** – windows metafile format (only on Windows)  
**postscript** – older format, also resizes well, usually portable, can be used to create encapsulated postscript files; Windows systems often don’t have a postscript viewer

Bitmap formats – represents data points as a series of pixels  
**png** – bitmapped format, good for line drawings or images with solid colors, uses lossless compression (like the old GIF format), most web browsers can read this format natively, good for plotting a LARGE number of points, does not resize well  
**jpeg** – good for photographs or natural scenes, uses lossy compression, good for plotting a LARGE number of points, does not resize well, can be read by almost any computer/web browser, meh line drawings  
**tiff** – creates bitmap files in the TIFF format; supports lossless compression  
**bmp** – a native Windows bitmapped format

**Multiple Open Graphics Devices**- can open multiple graphics devices   
-can only plot one graphics device at a time  
-currently active graphics device can be checked by dev.cur()  
-every open graphics device is assigned to a distinct integer >= 2  
-you can change the active graphics device with dev.set(<int>)

**Copying Plots  
dev.copy** – copy a plot from one device to another  
**dev.copy2pdf** – specifically copy a plot to .PDF file  
**dev.copy(png, file = “geyserplot.png”)** // copy plot to a PNG file from screen device