Ryan Bruderer
STA371H

Class Notes 1/27

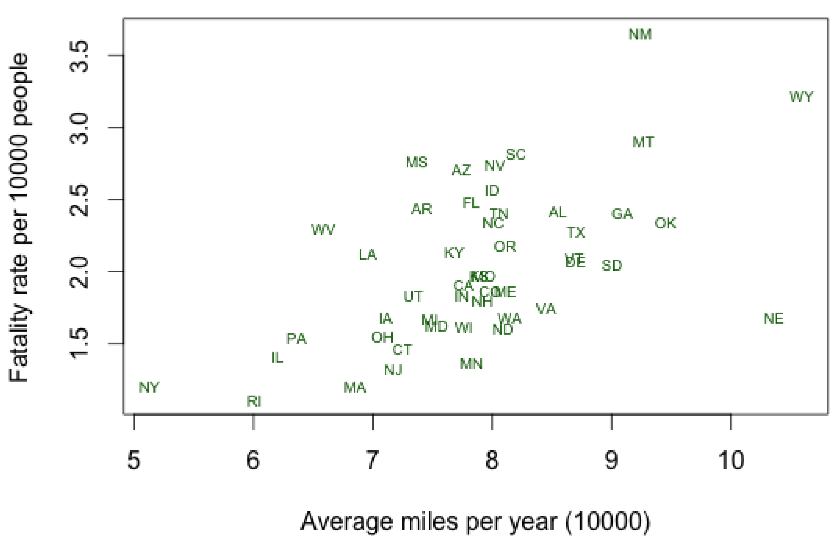
**Overall**
-Learned how to modify our assumption of simple linear regression with different models
-Learned how to merge data sets, aggregate variables, do and undo log transformations, fit non-linear models by least squares
-Learned about Power Laws

**Traffic Deaths**
Open ‘Trafficdeaths’ R Script and csv Data, as well as ‘fips’ csv Data

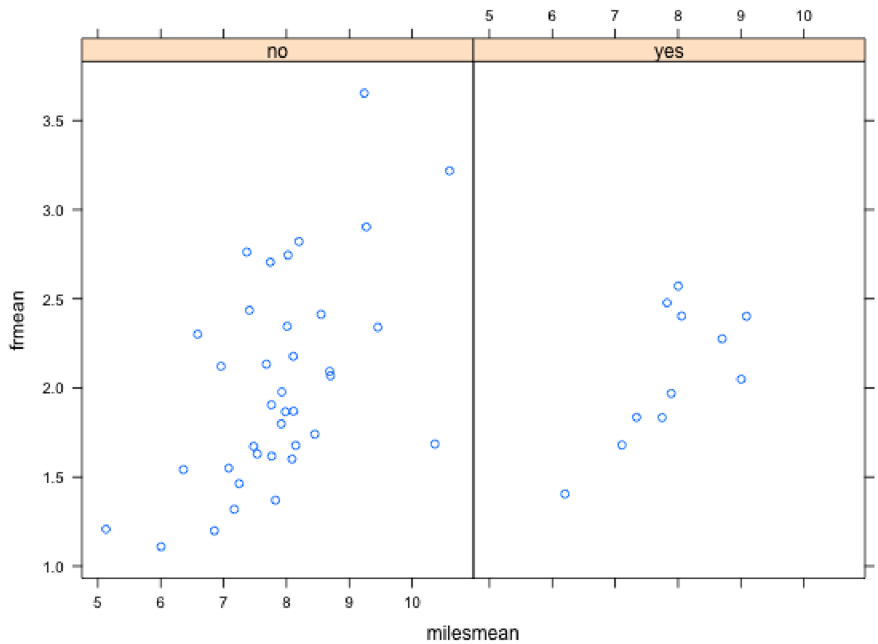
-You can merge the two data sets by using the ‘merge’ command
-The R code for this specific instance is:
traffic2 = merge(trafficdeaths, fips, by.x = "state", by.y="fipsnum")
-‘traffic2’ is the new, combined data set

-You can create new variables that aggregate data
-For example, the code: ‘frmean = mean(mrall~fipsalpha, data=traffic2)’ creates a variable that finds the average traffic deaths per 10,000 residents by state over all the years in the data set

-When plotting points on graphs, you can replace the points with text labels identifying what is being plotted using the ‘text’ function
-This code: ‘text(frmean~milesmean, labels=names(frmean), cex=0.6, col='darkgreen')’ allowed us to use state abbreviations as plotted data points instead of simple circles



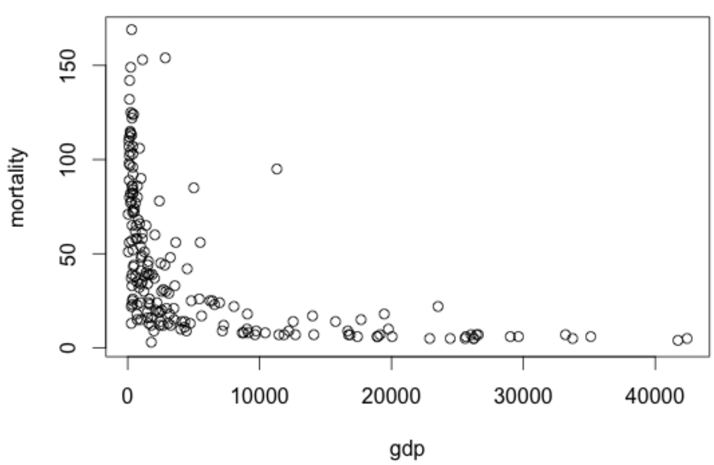
-You can stratify data by a third variable
-For example, we can compare states' fatality rates with how many miles their citizens drive per year, stratified by their drunk driving laws
-The code for this example is: 'xyplot(frmean~milesmean | jaild, data=traffic2)' and creates this plot:



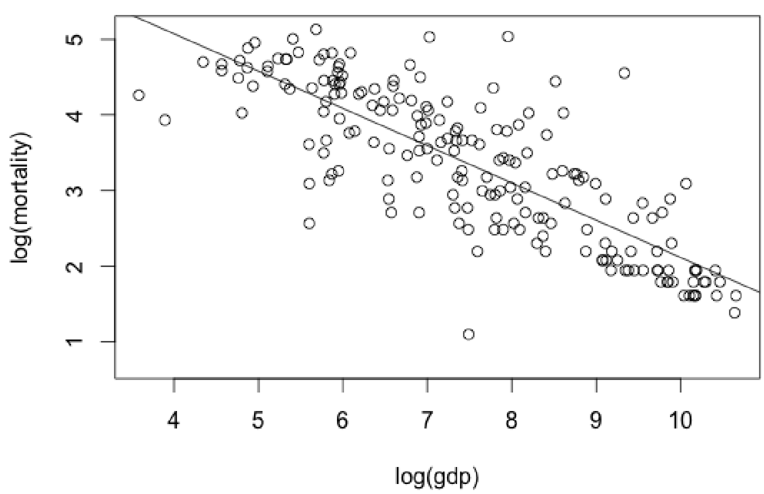

**Transformations**

Open 'Transformations' R Script and 'infmort' csv Data

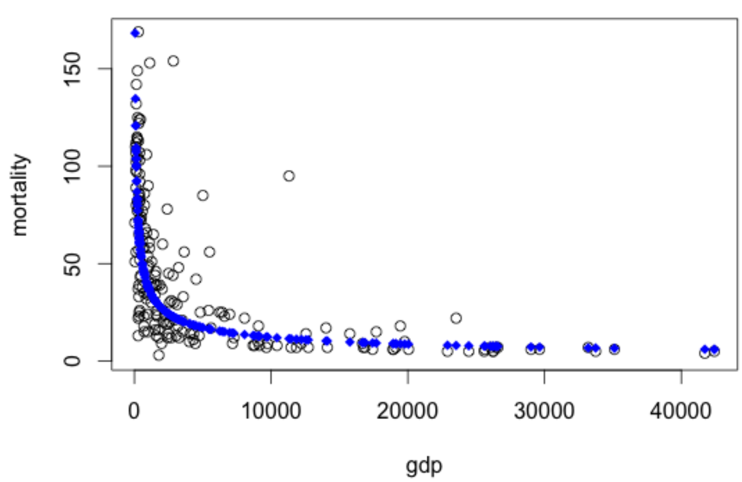
-Sometimes variables will have a non-linear relationship
-An example of this is infant mortality rates compared with per capita GDP by country



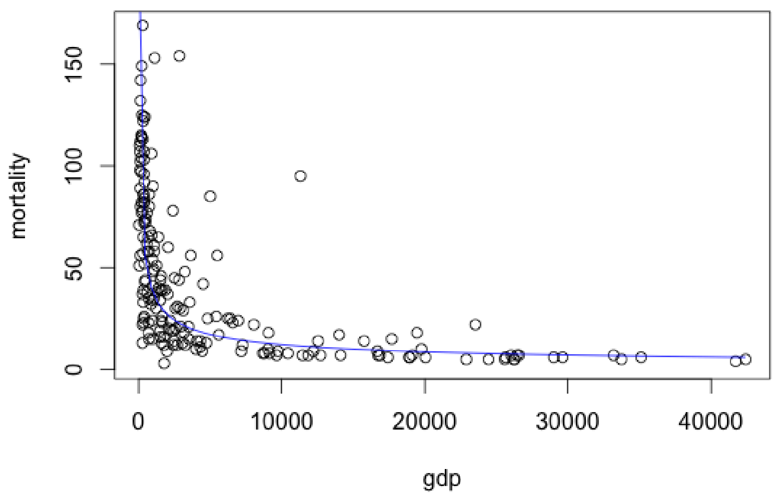
-When this is the case, you can finagle the data by using the 'log' function
-The 'log' transformation is most helpful when dealing with data that is bounded by zero but spans many orders of magnitude
-Data like this will often appear concentrated near the bottom/left corner of the plot with long tails stretching along each axis
-For this example, the code: 'plot(log(mortality) ~ log(gdp), data= infmort)' transforms the plot in a way that allows us to add a linear model




-You can predict using the model you have on the log scale
-Example code: 'logmort.pred = beta[1] + beta[2]*log(infmort$gdp)'
-After predicting, undo the log transformation using the 'exp' function
-Example code: 'mort.pred = exp(logmort.pred)'
-You can then add the predicted points to the original plot
-Example code: 'points(mort.pred ~ gdp, data=infmort, col='blue', pch=18)'



-You can also add a predictive curve on the original plot instead of only the points
-Example code for this is: 'curve(exp(beta[1]) * x^beta[2], add=TRUE, col='blue')' and creates this plot:




-If you have data between 0 and 1 with a non-linear relationship, the 'log' transformation may not be as effective

-In this case, use the 'logit' function

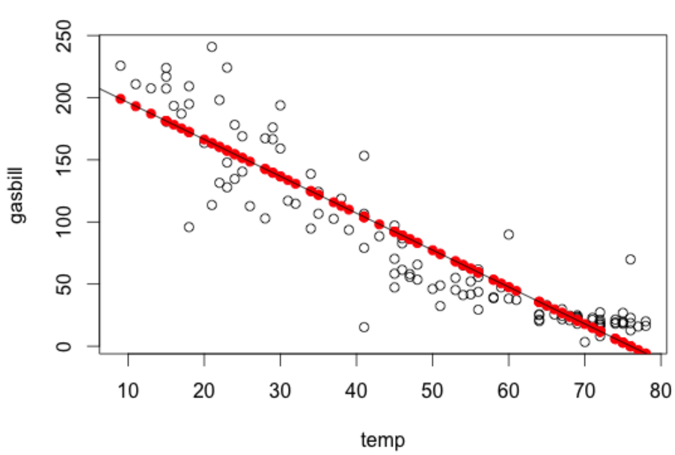
-In all cases, you do not have to transform both variables

-Sometimes taking the 'log' or 'logit' of only one variable will work

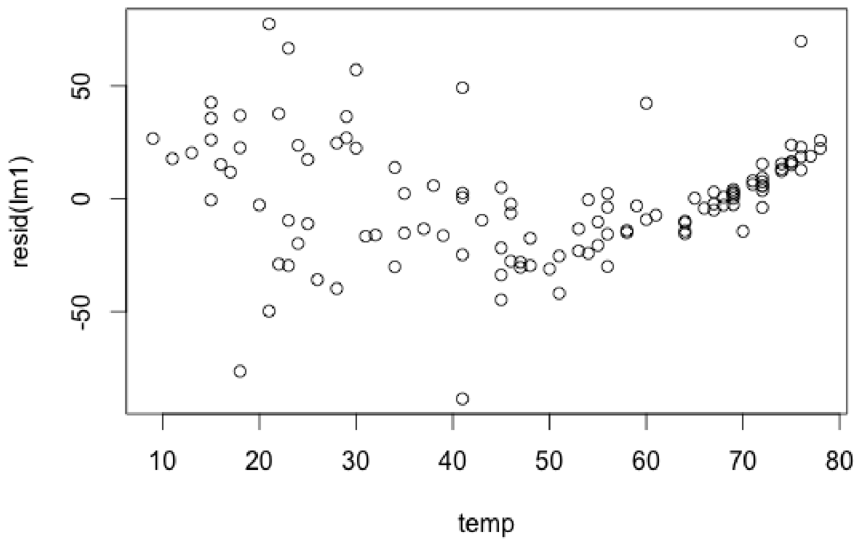
**Utilities**

Open 'utilities' R Script and cvs Data

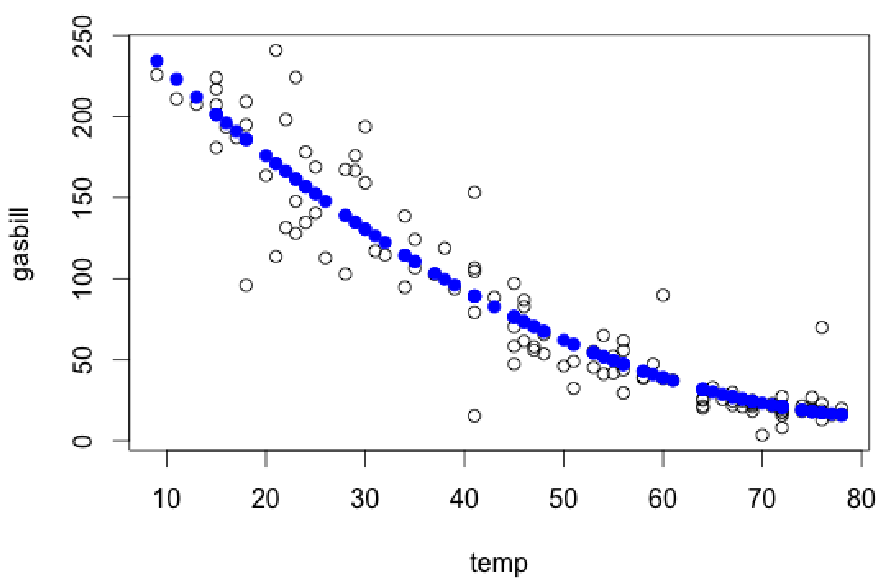
-The relationship between temperature and cost of gas bill is another example in which a linear model doesn't quite fit



-If the points of a residual graph from a linear model have a 'u' or 'smiley face' pattern, a quadratic model may be a better fit
-These patterns indicate there is still 'x-ness in y'




-You can create a quadratic model by adding a power to the predictor (or 'x') variable
-Example code: 'lm2=lm(gasbill ~ temp + I(temp^2), data=utilities)'
-This new quadratic model fits the data much better:



**Power Law**

-The power law is a relationship between two variables in which one varies as a power of the other

-The original equation for fitting a linear model is $y_i = B_0 + B_1x_i + e_i$

-When doing a log transformation, this model equation is changed and becomes:

-$\log(y_i) = B_0 + B_1\log(x_i) + e_i$

-The log transformation is undone by using the exp function:

-$e^\wedge\log(y_i) = e^\wedge(B_0 + B_1\log(x_i) + e_i)$

-After simplifying, the equation is:

-$y_i = e^\wedge B_0 * x_i^\wedge B_1 * e^\wedge e_i$

-This proof shows that the relationship between x and y varies as a power of x

-It shows how you can fit a non-linear model to log-transformed data

-Key difference between beginning and ending equations is the error, or 'e'

-In the initial equation it is added

-In the final equation it is multiplied