Stats503Hw2

David Li

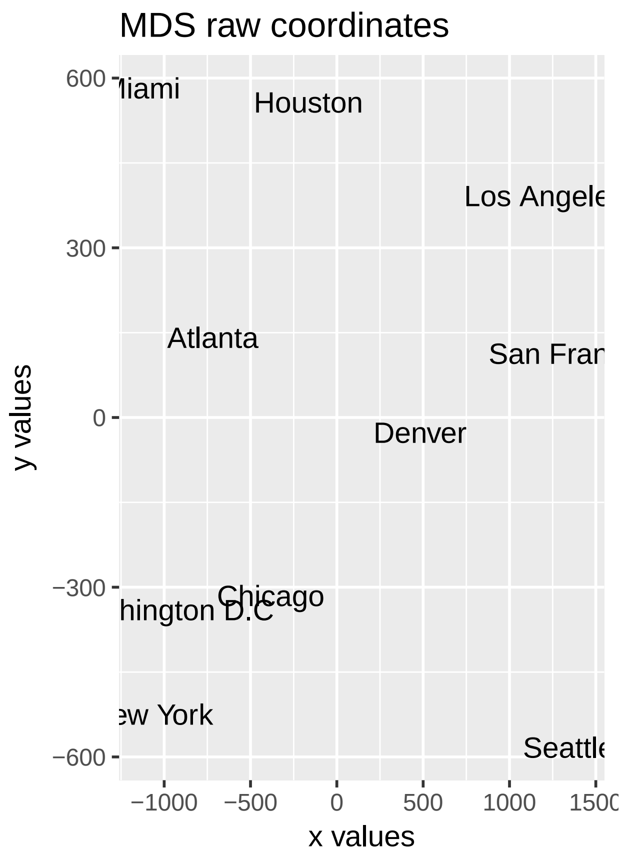
February 7, 2018

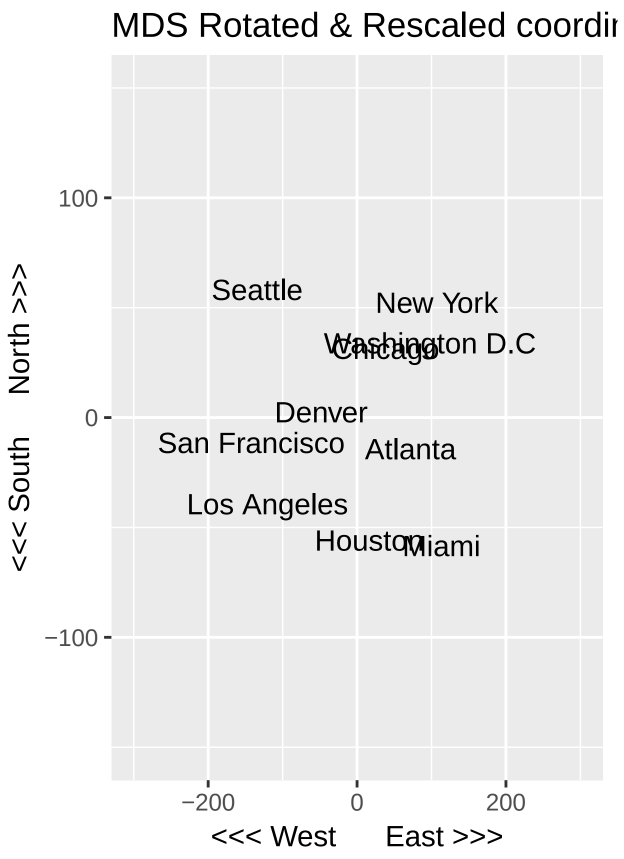
Data report for Stats503 Homework 2

Part 1 (Q2)

Multidimensional scaling is a very popular modern computational method as a way of depicting the level of similarity of individual cases in a dataset, usually by visualization. Typically for example, we could be given a distance matrix and we would be seeking to reduce the dimension of the problem; that is we try to place objects in the dataset within some n dimensional space. The goal is to preserve the "distance" between the objects as much as possible so we are minimizing the loss of data. Once the visualization is constructed, we can analyze the MDS plot and discover along with make conclusions upon the high dimensional data.

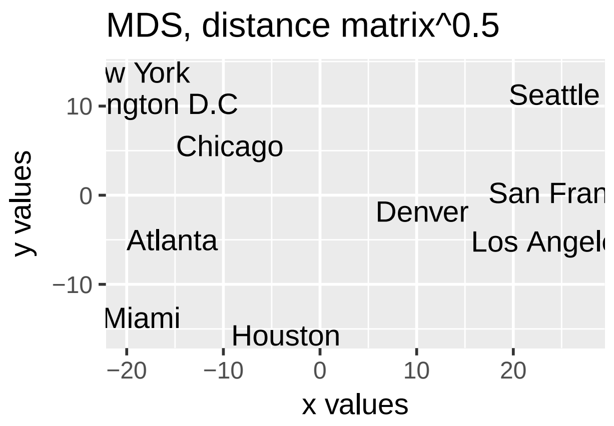
The algorithm can be implemented in many different ways; in fact there are many packages (some even built-in) in statistcal computing languages that specifically handle MDS processing. We fully self-implement code without package assistance which is attached in the appendix, but the results are below and we will provide some remarks. The output of the algorithm should be a visualization helping to provide insight into the data (usually n=2 or n=3 dimensions). We will apply the algorithm to an example 10x10 distance matrix between 10 U.S. cities.

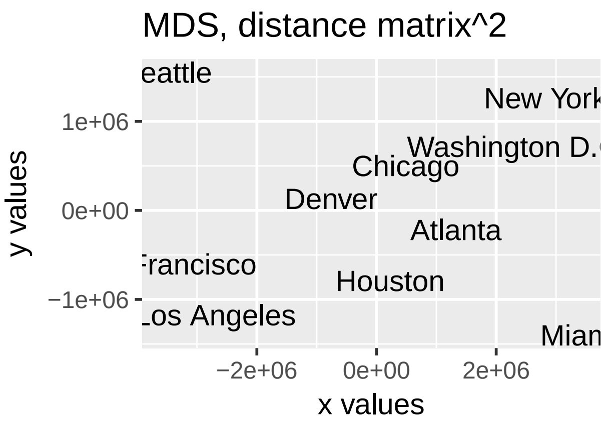


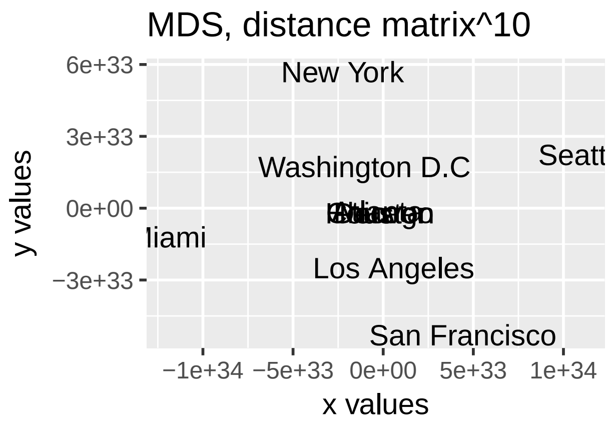


The raw dimension-reduced coordinates out of MDS (graph on left) present the cities with correct relative distances away from each other, which is available for analysis as we can clearly comment on which cities are far apart and which cities are close. But the locations are flipped to the viewer's inconvenience. We can easily fix this by using a band-aid fix of rotating and rescaling the coordinates to a reasonable scale, as the graph on the right shows. If we were further interested and wanted to decorate the visualization, we could overlay a map over the right graph to see the USA map border and do further adjustments to the coordinates. However for now, it is clear that MDS is a powerful and effective tool to receive multi-dimensional data (in this case, a 10 x 10 distance matrix) and to transform it into lower dimensions of a more viewable and digestable form of data.

One may ask "Can we only use the raw distances given, or are we able to transform the distances and have MDS perform on that as well?" The fact is that the distances can also be transformed if ever needed, and MDS can provide meaningful results as well albeit being cognizant of the nature of the transformation. For instance, we can try to exponentially adjust the data in the distance matrix and perform MDS as well. The results are seen below:







In our case, it looks like adjusting the distance matrix causes cities towards the center of the U.S. to be grouped tightly together while the outer cities stay fairly spread out if we are raising the distance matrix to a larger and larger power. Regardless of this effect, the relative locations are still quite accurate from the MDS process so the MDS can accept and deal with not just the original distances. We are quite happy that not only our MDS self-implementation works based on the foundation of the algorithm of MDS, but also with the power and usefulless that such a tool like MDS provides to us in analyzing high-dimensional data in statistics such as applied here.

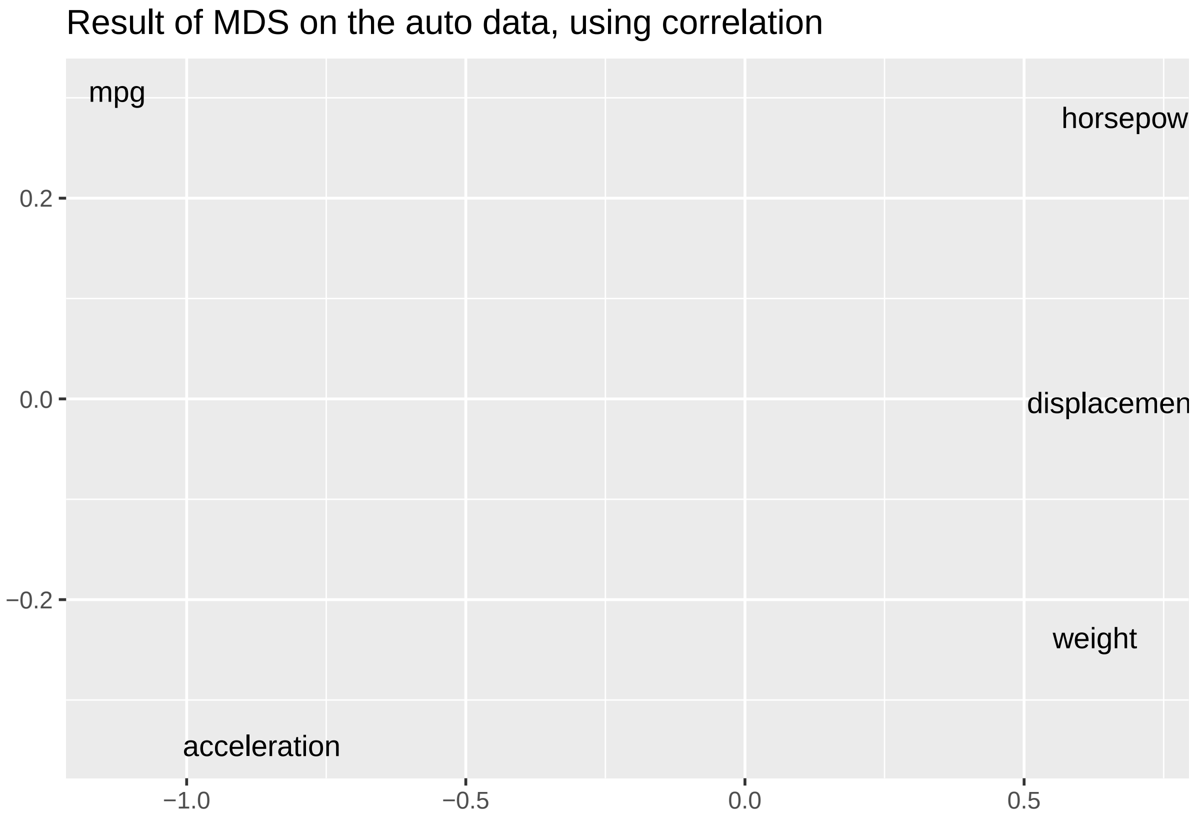
Part 2 (Q3)

One other thing to explore in this report; how exactly does factor analysis and MDS compare in relation to PCA? They all involve some idea of reduction of complex data into a better analyz-able form (whether lower dimension or in a different form.) But the gold mine is that we want to observe the results of each in application, and pursue finding the advantages and disadvantages of each in appropriate utilization. Our main goal is to understand the results of each method and know clearly when to use which method; this is a vital critical thinking task for developing professional statisticians after all.

We will preface each method with a short introductory description for the layman, and apply the results to a given dataset auto-mpg.data. Then we will comment on both the results and what we have learned about the appropriateness of the technique.

Multidimensional Scaling (MDS)

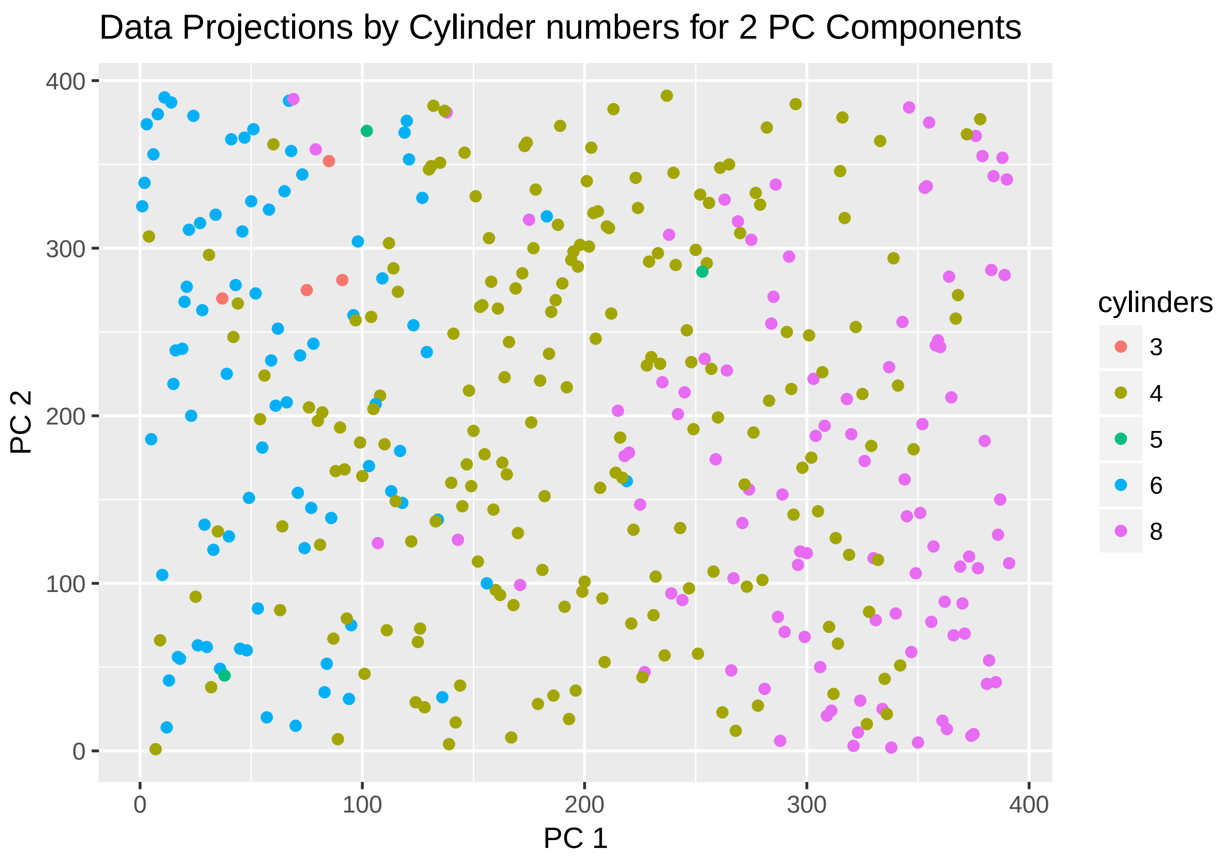
Recall from Part 1 that Multidimensional Scaling is a method to visualize similarities in high dimensional data, usually reducing it to lower dimensions for visualization and inference. It is based upon the idea that we reduce dimensions using the idea of preserving the distances contained between pairs of items or objects and minimizes the loss resulting from the compression, usually reliant on a loss function to calculate the loss and obtain optimization. It is particularly popular in cases where the data is of high complexity but requires or desires a reduction in dimensions for understanding or visualization. In effect, MDS essentially is a class for assorted statistical methods that seek to understand high dimensional data usually through dimension reduction; PCA and Factor analysis can be considered methods that hinge on the idea of MDS.



Of now, we can see that horsepower, displacement, and weight correlate in a similar direction while mpg and acceleration are each individually spread from everyone else. For now, this is all we will see since MDS is just the reduction of the data complexity in a compressed state.

Principal Component Analysis (PCA)

Principal Component Analysis is a statistical process based upon orthogonal transformations of observations to organize the values into principal components, which are linearly uncorrelated variables. The end goal is designed to reveal relationships and connectedness in the variables and observations through the linear compositions of the observed variables which compose as the principal components. PCA utilizes eigenvectors and eigenvalues to achieve its goal, and is one of the simplest methods of its type in the multivariate analysis field.



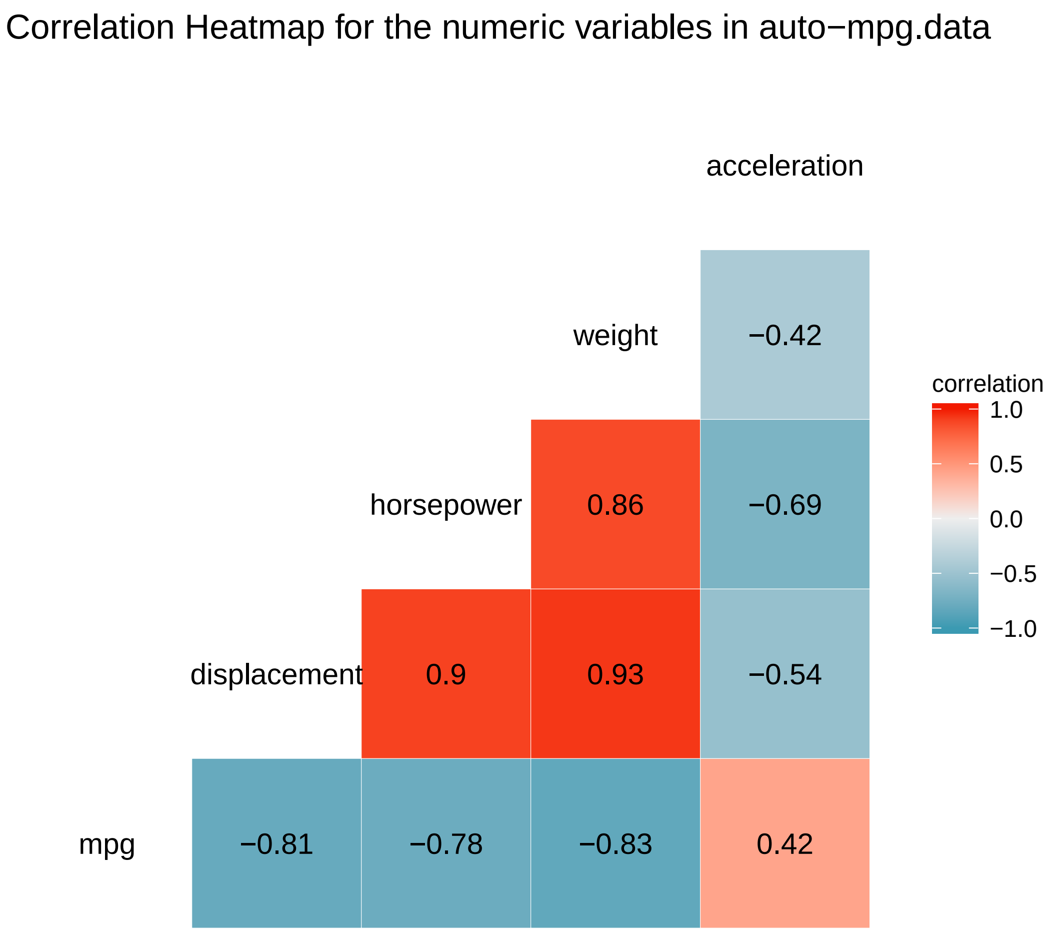
Note that as concluded in the last report, we chose to discard the categorical variables and focus upon the numerical variables for simplicity. We also discovered that cylinders was the most intriguing categorical variable to track, so the plot is based upon that. We will follow similarly for Factor Analysis to hold equal treatment and have grounds for comparison later.

From the plot and similar conclusions of last report, most of the data is for 4, 6, or 8 cylinders (presumably the most common configurations for cars). The PCA did quite a nice job of displaying the high-dimensional data into 2 dimensions (principal components), and has a fairly clear division in distribution of the points between the 3 main cylinder groups. Though we could have easily done a simple numerical summary and seen similar results, a very very large dataset would have been even more difficult to process with all of the variables and a simple numerical summary would discard many other variables of interest. PCA is ran on the whole dataset which preserves most information possible. Otherwise, there are only slightly few outstanding outliers in the main cylinder groups as seen on the PCA plot; this is seen as data points for one cylinder group that could easily be mistaken for another cylinder group based on the point placement in the graph.

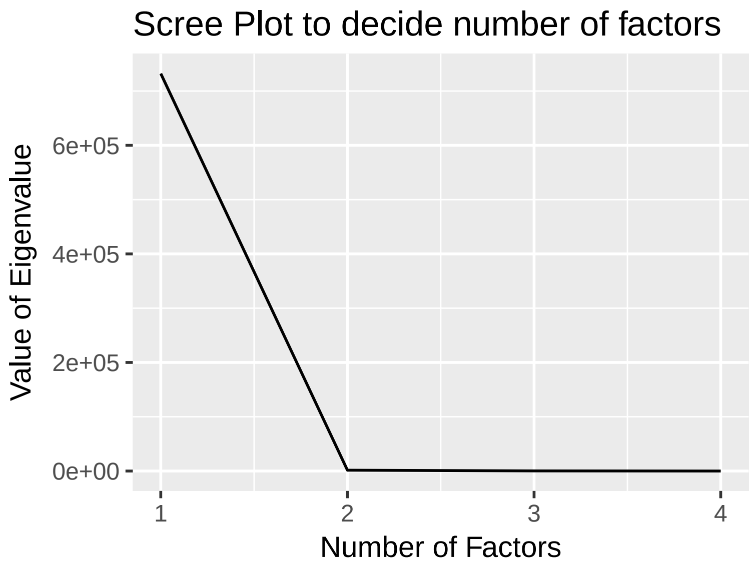
Factor Analysis (FA)

Factor Analysis is a statistical process used to quantify the variability between correlated variables amongst observations in relation to a series of a lower number of unobserved variables, called factors. It is based upon the idea that we "exploit" the highly correlated structure of the variables which can lead to a reduction of the set of variables within the dataset. In theory, it is highly effective in areas of study where there tends to be large datasets with a large number of likely correlated variables such as biology for instance. We can proceed as follows:

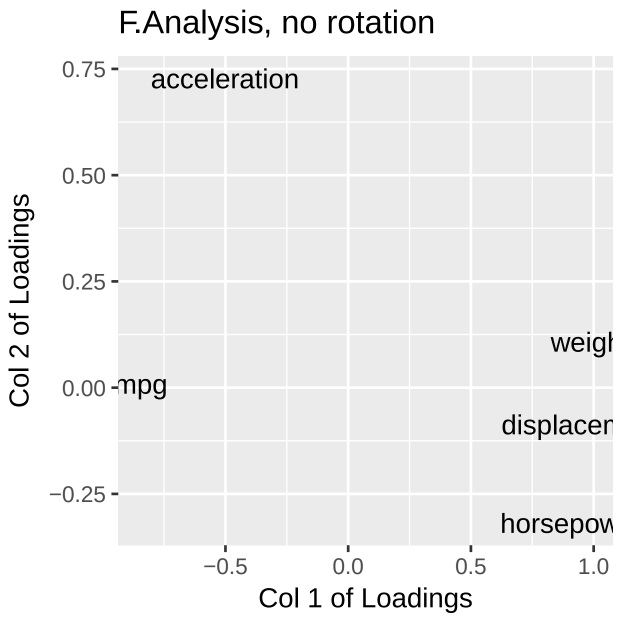
Remember that Factor Analysis relies that there are correlated variables. So we should verify this is the case, with a correlation heatmap. It turns out that weight, horsepower, and displacement are all highly correlated with each other so this is good reason to consider Factor Analysis.

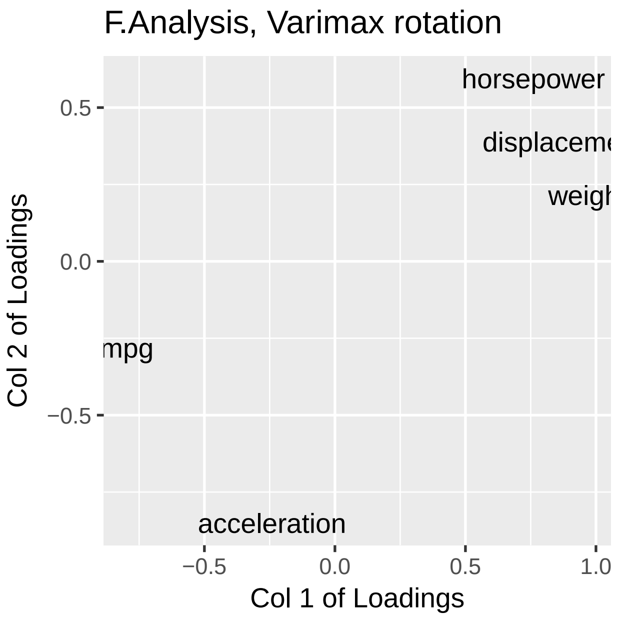


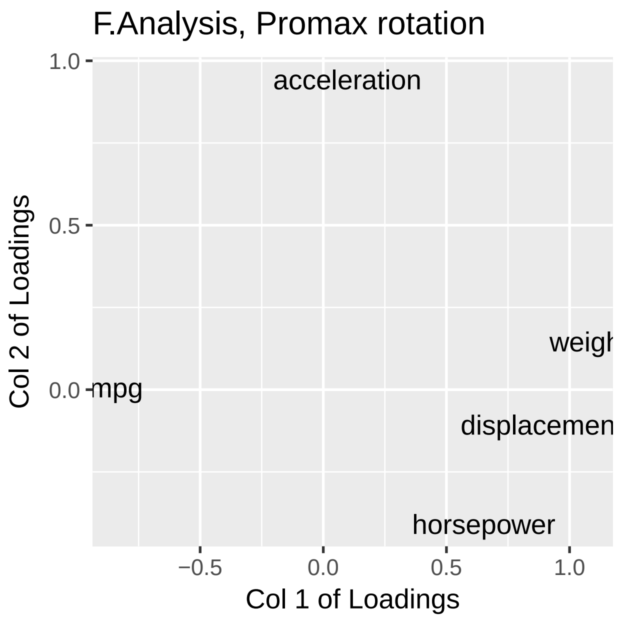
Next, we probably should consider how many factors to use. Similar to PCA, we want to capture the highest amount of variance explained with a minimal number of factors. The higher amount of variance explained in trade-off to number of factors matters for effective analysis; typically a scree plot is used in analysis to decide on the number of factors. From the scree plot given, it seems that 2 factors is the best way to go.



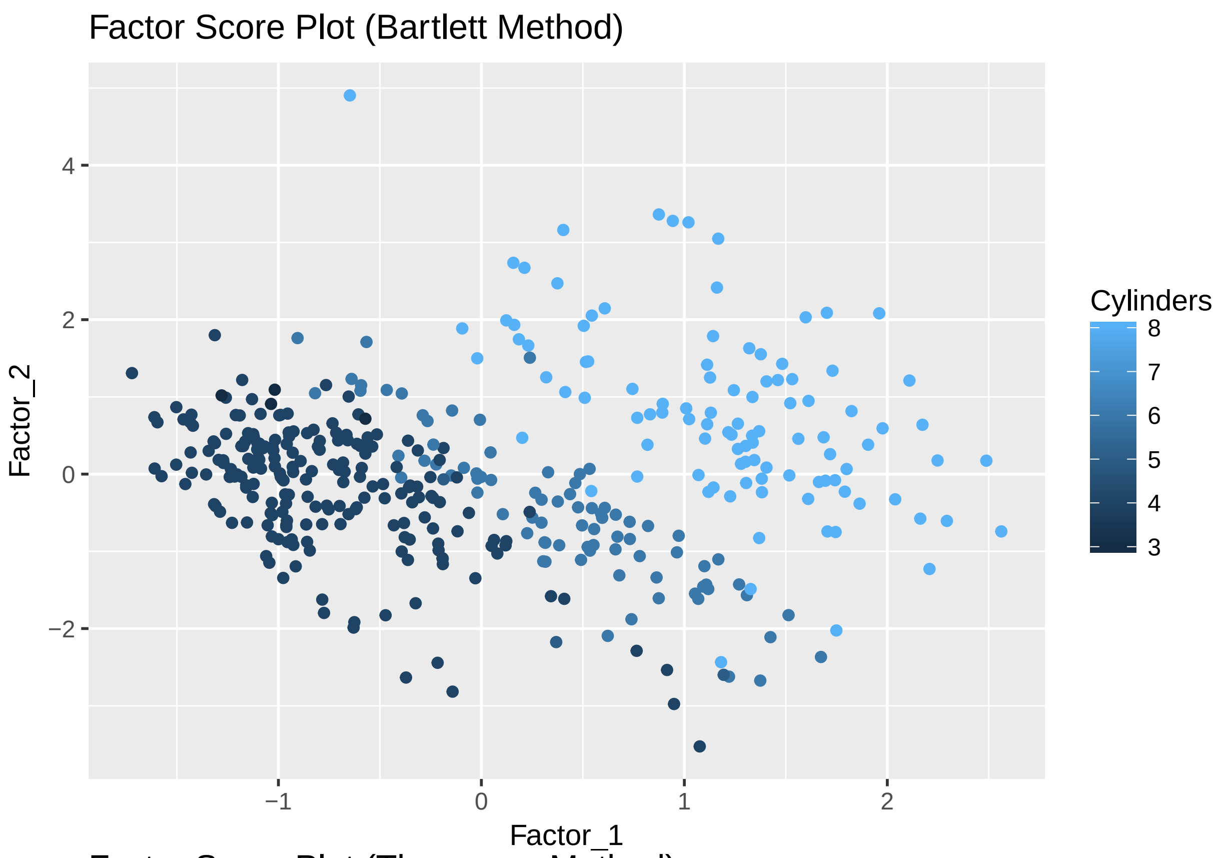
We can go ahead and proceed with the main bulk of factor analysis, which involves us finding the most appropriate latent factors for a model in explaining the observed variables. That is, the model involving random errors along with common factors and factor loadings. Rotations also need to be considered, which are simply reorganizing the factor loadings in a way that may be easier to interpret given the context of the dataset and the analysis. Two well known rotations are varimax and promax rotations, which respectively are orthogonal rotation and oblique rotations that we will consider in the process. Below are the plots of the loadings for each rotation type.

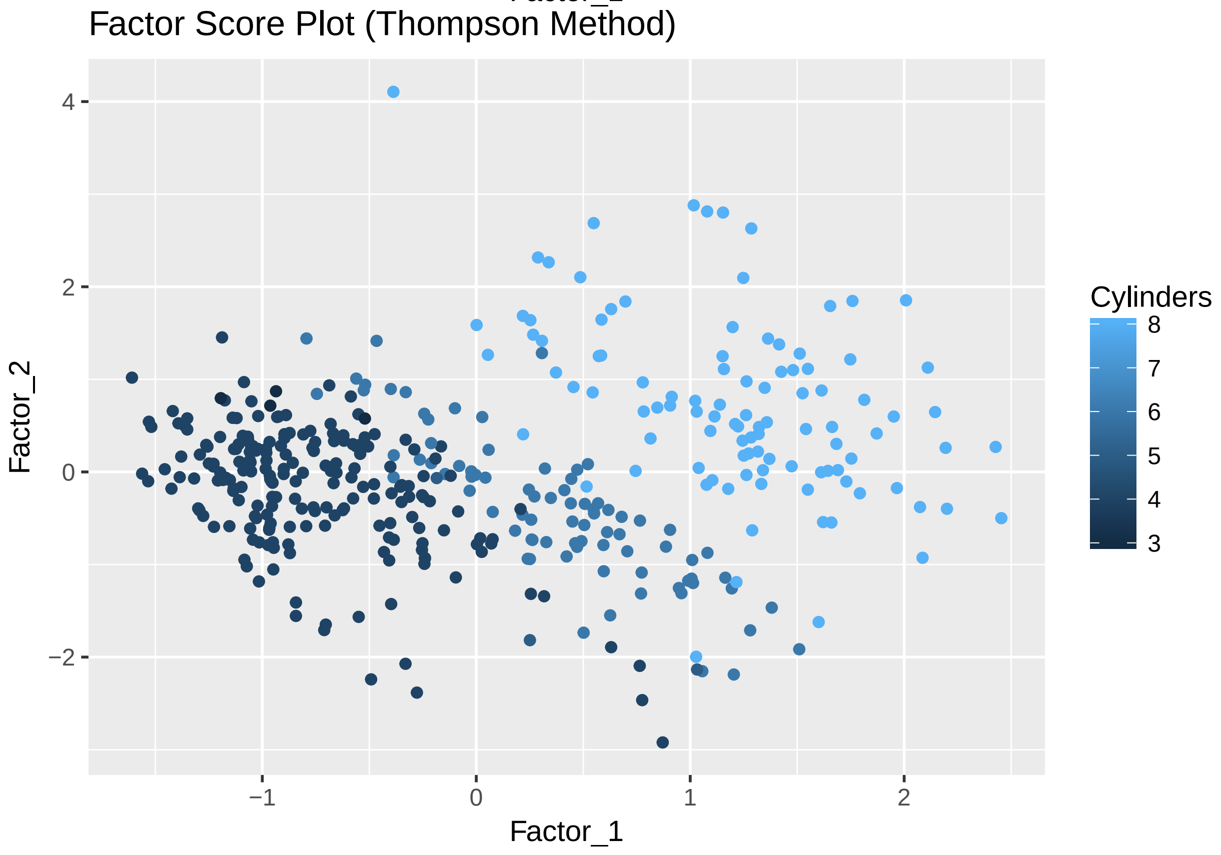






As figured, the differences between the rotation types are minimal since they all support the same notion: weight, displacement, and horsepower have similar direction of correlation against mpg and acceleration which are independently spread from each other. Notice this is the same pattern observed in the MDS analysis above, even though there are very small differences in the plots due to the nature of each analysis.





These are excellent plots to display the Factor Scores for both Thompson's Method and Bartlett's Method;

the difference between the two methods is merely the procedure of estimating the scores whether by a Bayesian approach or not. As we can see, the Factor Analysis (of using 2 factors in both approaches) proves to show there is quite a noticable difference in the number of cylinders. Overall, Bartlett and Thompson show the same pattern which also matches the observations noticed in the PCA approach - the number of cylinders definitely do matter in nature!

In conclusion, FA and PCA are actually very similar with a few small differences since they both are designed to reduce the data to a simpler degree. However, Factor Analysis is more reliant on other assumptions about the structure of the data to yield valid discoveries while PCA is a more generic method that does not necessarily require strong correlation. The nature of each is different as well, since PCA is a pure reduction into composite variables while FA is geared towards testing how well latent factors form a theoretical model for their observed variables. In that sense, Factor Analysis and PCA are not true twins either since they do serve different purposes. As for MDS, it simply has a label as more of a type/class of analysis rather than it's own method. Therefore PCA can be thought of as a "branch" method or "case" of MDS usage. Factor analysis has shown to be similar to PCA, and it wouldn't be incorrect to consider Factor analysis as another MDS method. But as a reminder, PCA and FA have different purposes though they belong to the same "family" of MDS.

Appendix: R Code

```r
# Some preliminary loading
library("gridBase", lib.loc="~/R/x86_64-pc-linux-gnu-library/3.4")
library("gridExtra", lib.loc="~/R/x86_64-pc-linux-gnu-library/3.4")
library("ggplot2", lib.loc="~/R/x86_64-pc-linux-gnu-library/3.4")
library("dplyr", lib.loc="~/R/x86_64-pc-linux-gnu-library/3.4")

# Assuming input is a valid distance matrix, and number of desired dimensions is 2
own_mds = function(distmat, dimen = 2){

# Gram Matrix - The functionality of the Gram Matrix is simply a
# conversion of the distance matrix, and is involved in the later eigen-decomposition
# for the 2-D solution representation. Notice that the distance matrix is approximately euclidean;
# so we square this in the gram matrix calculation
numrows = dim(distmat)[1] # obtains rows
vecones = rep(1, numrows)
vectrans = t(vecones)
h = diag(1, numrows) - (1/numrows)*(vecones%*%vectrans)

gram = -0.5 * (h) %*% (distmat^2) %*% (h)

# Eigen-decomposition for the Gram Matrix - Used to achieve the solution for multi-dimensional scaling
eigenvalues = eigen(gram)$values
eigenvectors = eigen(gram)$vectors

reducedsolution = eigenvectors %*% diag(eigenvalues^0.5)

# Retain the first dimen columns, user will need to supply the names and compile with a map themselves.
mdsmat = as.data.frame(reducedsolution[, 1:dimen])
return(mdsmat)
}

# Test-drive on the given distance matrix
namesofcities = c("Atlanta","Chicago","Denver","Houston","Los Angeles",
            "Miami","New York","San Francisco","Seattle","Washington D.C")

distmat = matrix(c(0, 587,1212, 701,1936, 604, 748,2139,2182, 543,
                 587,   0, 920, 840,1745,1188, 713,1858,1737, 597,
                1212, 920,   0, 879, 831,1726,1631, 949,1021,1494,
                 701, 840, 879,   0,1374, 968,1420,1645,1891,1220,
                1936,1745, 831,1374,   0,2339,2451, 347, 959,2300,
                 604,1188,1726, 968,2339,   0,1092,2594,2734, 923,
                 748, 713,1631,1420,2451,1092,   0,2571,2408, 205,
                2139,1858, 949,1645, 347,2594,2571,   0, 678,2442,
                2182,1737,1021,1891, 959,2734,2408, 678,   0,2329,
                 543, 597,1494,1220,2300, 923, 205,2442,2329,   0),
                nrow = 10, byrow = T)

mdsmatrix = own_mds(distmat)

# Original coordinates straight out of decomposition
mdsgraph1 = ggplot(data = mdsmatrix, aes(x=V1, y=V2)) +
            geom_text(label=namesofcities) +
            ggtitle("MDS raw coordinates") +
            labs(x = "x values", y = "y values")

# Rotated and Rescaled for better viewing
mdsgraph2 = ggplot(data = mdsmatrix, aes(x=-0.1*V1, y=-0.1*V2)) +
            geom_text(label=namesofcities) +
            xlim(-300, 300) +
            ylim(-150, 150) +
            ggtitle("MDS Rotated & Rescaled coordinates") +
            xlab("<<< West      East >>>") +
            ylab("<<< South     North >>>")

grid.arrange(mdsgraph1, mdsgraph2, nrow=1)

scaleddistmat_0.5 = distmat^0.5
scaleddistmat_2 = distmat^2
scaleddistmat_10 = distmat^10

results1 = own_mds(scaleddistmat_0.5)
results2 = own_mds(scaleddistmat_2)
results3 = own_mds(scaleddistmat_10)

mdsgrapha = ggplot(data = results1, aes(x=V1, y=V2)) +
            geom_text(label=namesofcities) +
            ggtitle("MDS, distance matrix^0.5") +
            labs(x = "x values", y = "y values")
mdsgraphb = ggplot(data = results2, aes(x=V1, y=V2)) +
            geom_text(label=namesofcities) +
            ggtitle("MDS, distance matrix^2") +
            labs(x = "x values", y = "y values")
mdsgraphc = ggplot(data = results3, aes(x=V1, y=V2)) +
            geom_text(label=namesofcities) +
            ggtitle("MDS, distance matrix^10") +
            labs(x = "x values", y = "y values")

grid.arrange(mdsgrapha, mdsgraphb, mdsgraphc, nrow=2, ncol = 2)

# Import the dataset first
auto = read.table("~/Desktop/School/Stats503/Datasets/auto-mpg.data", stringsAsFactors = FALSE)
colnames(auto) = c("mpg", "cylinders", "displacement", "horsepower", "weight", "acceleration", "modelyea
# Using non-categorical variables
PCAset = auto %>%
  transmute(mpg, displacement, horsepower, weight, acceleration)
# Coercions for component analysis to work properly
PCAset$horsepower <- as.numeric(as.character(PCAset$horsepower))
PCAset2 = na.omit(as.matrix(PCAset))

# MDS, using a correlation matrix approach
MDSAuto = cmdscale(1 - cor(PCAset2))
CorrelationPlot = qplot(x=-MDSAuto[,1],y=-MDSAuto[,2],
label=rownames(MDSAuto), geom = "text")+
labs(title = "Result of MDS on the auto data, using correlation", x = "", y = "")
```

```r
CorrelationPlot

# PCA based on correlation matrix
PCAcorr = princomp(PCAset2, cor = T)

# Establishing a new dataset with the PCA scores attached
newauto = auto
newauto$horsepower = as.numeric(as.character(newauto$horsepower))
newauto = na.omit(as.matrix(newauto))
newdata = as.data.frame(cbind(newauto, PCAcorr$scores[,1], PCAcorr$scores[,2]))

# Displaying the PCA scores in first two PCs, colour-coded by number of cylinders
ggplot(newdata, aes(x = as.numeric(V10), y = as.numeric(V11), colour = cylinders)) +
  geom_point() + labs(title = "Data Projections by Cylinder numbers for 2 PC Components", x = "PC 1", y

ggcorr(PCAset2, label_round =2, label = T, name = "correlation")+
ggtitle("Correlation Heatmap for the numeric variables in auto-mpg.data")

# Scree plot of eigenvalues
eigenmat = data.frame(eigenvalues = eigen(cov(PCAset2))$values[1:4], index=1:4)
screeplot =
ggplot(data=eigenmat, aes(x=index)) +
geom_line(aes(y=eigenvalues))+
labs(title = "Scree Plot to decide number of factors", x = "Number of Factors", y = "Value of Eigenvalu
screeplot

# Executing Factor Analysis for loadings, with varied rotations
FAreg = factanal(x = PCAset2, factors=2, rotation="none")
FAvarimax = factanal(x = PCAset2, factors=2, rotation="varimax")
FApromax = factanal(x = PCAset2, factors=2, rotation="promax")

# No Rotation
plot1 = qplot(x=FAreg$loadings[,1], y=FAreg$loadings[,2], label=rownames(FAreg$loadings), geom="text") 
labs(title = "F.Analysis, no rotation", x = "Col 1 of Loadings", y = "Col 2 of Loadings")

# Varimax rotation
plot2 = qplot(x=FAvarimax$loadings[,1], y=FAvarimax$loadings[,2], label=rownames(FAvarimax$loadings), g
labs(title = "F.Analysis, Varimax rotation", x = "Col 1 of Loadings", y = "Col 2 of Loadings")

# Promax Rotation
plot3 = qplot(x=FApromax$loadings[,1], y=FApromax$loadings[,2],
              label=rownames(FApromax$loadings), geom="text") +
              labs(title = "F.Analysis, Promax rotation", x = "Col 1 of Loadings", y = "Col 2 of Loadin
grid.arrange(plot1, plot2, plot3, nrow = 2, ncol = 2)

# Making a new dataset with cylinders attached
omitted = auto %>%
  transmute(mpg, displacement, horsepower, weight, acceleration, cylinders)
# Coercions for component analysis to work properly
omitted$horsepower = as.numeric(as.character(omitted$horsepower))
omitted = na.omit(as.matrix(omitted))

# Performing Factor Analysis for Bartlett's method
```

```r
BartScores = factanal(x=omitted[,1:5], factors=2, scores='Bartlett')
scores2 = BartScores$scores
BartMat = data.frame(Cylinders= omitted[,6], Factor_1=BartScores$scores[,1],
Factor_2 = BartScores$scores[,2])
FABartCylin =
ggplot(BartMat,aes(x=Factor_1,y=Factor_2)) + geom_point(aes(colour=Cylinders))+
labs(title = "Factor Score Plot (Bartlett Method)")

# Performing Factor Analysis for Thompson's Method
ThompScores = factanal(x=omitted[,1:5], factors=2, scores='regression')
ThompMat = data.frame(Cylinders= omitted[,6], Factor_1=ThompScores$scores[,1],
Factor_2 = ThompScores$scores[,2])
FAThompCylin = ggplot(ThompMat,aes(x=Factor_1,y=Factor_2)) + geom_point(aes(color=Cylinders))+
labs(title = "Factor Score Plot (Thompson Method)")

FABartCylin
FAThompCylin
```