HW2

Due: 2/8/2018

Naomi Giertych

Question 1

We were asked to estimate the factor scores for a data point $x = (1, 1, 1, 1)$. In parts (a) and (b), we made additional assumptions and calculated the covariance matrix for the given models. Below, I calculate Λ, $\sum^{-1}$, and $\hat{f}_i$, respectively, in order to estimate the factor scores.

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    2   -1
## [3,]    1    2
## [4,]    3   -4
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2  2.0  1.0    3
## [2,]    2  5.5  0.0   10
## [3,]    1  0.0  5.5   -5
## [4,]    3 10.0 -5.0   27
```

```
##             [,1]       [,2]       [,3]        [,4]
## [1,]  0.93153153 -0.2306306 -0.2234234 -0.05945946
## [2,] -0.23063063  1.0126126 -0.3315315 -0.41081081
## [3,] -0.22342342 -0.3315315  0.4288288  0.22702703
## [4,] -0.05945946 -0.4108108  0.2270270  0.23783784
```

```
##      [,1]
## [1,]    1
## [2,]    1
## [3,]    1
## [4,]    1
```

```
##           [,1]
## [1,] 0.5819820
## [2,] 0.1837838
```

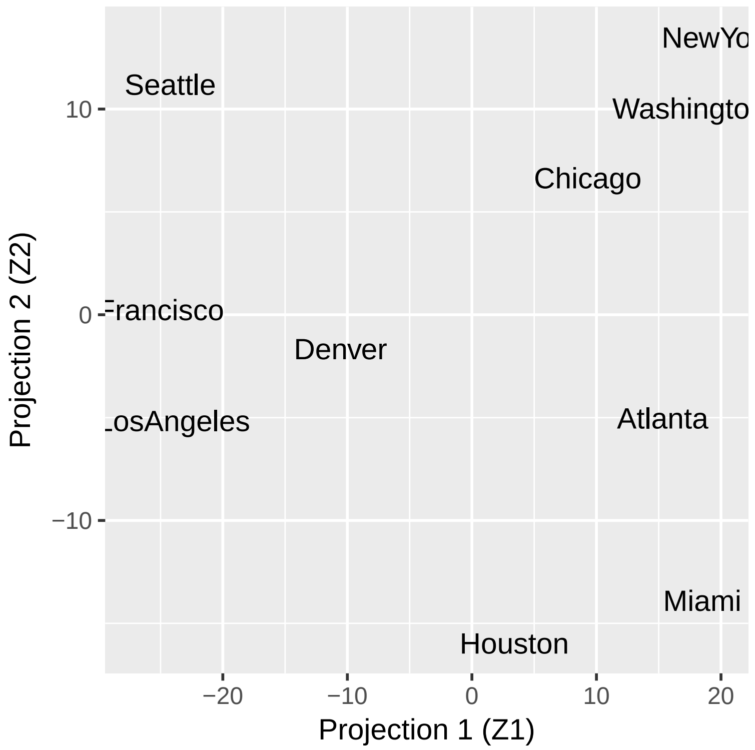
Question 2

The U.S. cities distance dataset contains "straight line", i.e. airline, distances (in miles) of all pairwise combinations of Atlanta, Chicago, Denver, Houston, LA, Miami, NY, San Francisco, Seattle, and Washington D.C.

We were asked to analyze the data using multidimensional scaling (MDS) methods that we developed. MDS relies on similarity or dissimilarity measures to reduce the number of dimensions of a dataset. In our case, the dissimilarities are the measurements of distances between the cities and we're assuming that they lie in a Euclidean space (although technically they don't). An MDS approximation of the data would allow us to preserve the similarities between the cities (e.g. how close they are to one another instead of how far apart they are).

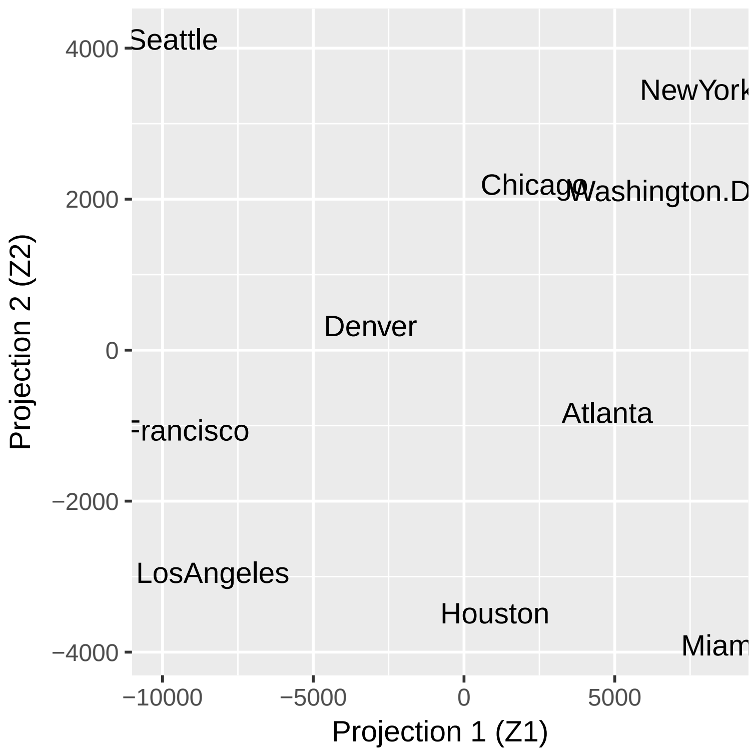
To create an MDS approximation of the US cities data, there are a few steps: 1) convert the distance matrix to a Gram matrix, 2) find the eigendecomposition of the Gram matrix, and 3) keep the first k λ (eigenvalues) and eigenvectors, where k is the number of dimensions we're interested in. The Gram matrix is made by "double centering" the data by the observations and the variables; this is performed by computing the following: $-\frac{1}{2}(I_n - \frac{1}{n}11^T)\mathbb{D}_{\Bbbk\times\Bbbk}(I_n - \frac{1}{n}11^T)$ where I_n is the $n \times n$ identity matrix, $1_{n\times 1} = (1, 1, \ldots, 1)$, and $\mathbb{D}$ is the original distance matrix. After finding the eigendecomposition (using the eigen function in R), we keep the k dimensions that we're interested in by multiplying the square-root of the eigenvalues with their corresponding eigenvectors.

In our analysis, we are only interested in the first two dimensions since those are likely to be easy to interpret in terms of directions (north-south and east-west). I estimate these dimensions using the process described above using the original distance matrix and graph the result.



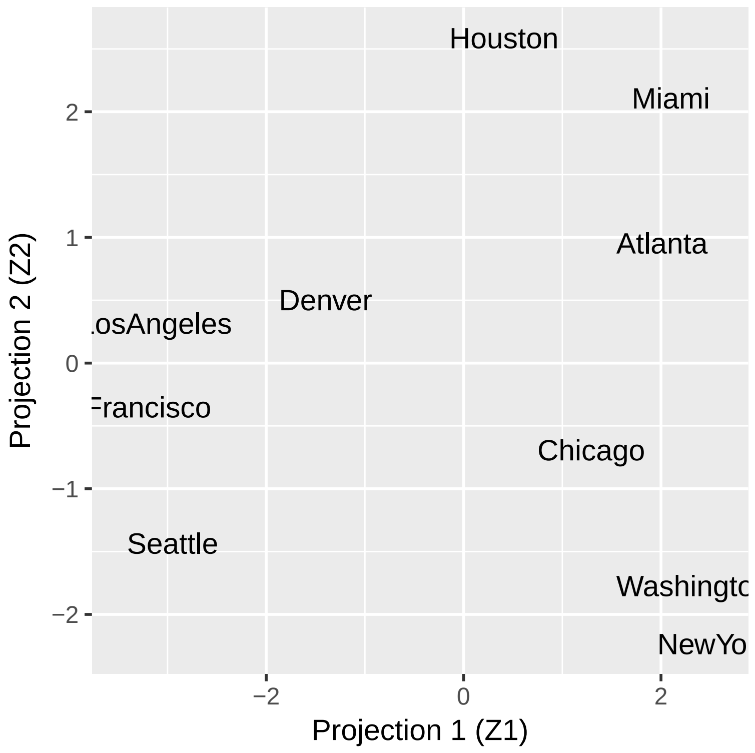
The plot shows the relative distances between the U.S. cities. Note, in the graph I swapped the signs of the dimensions for easier interpretation. It is interesting to note that the "map" of the U.S. is opposite of North-South and East-West without swapping the signs. It's unclear why that might be the case. A reasonable interpretation of the two dimensions is easy: dimension 1 is East-West and dimension 2 is North-South. This is expected given that the original space is not significantly different from Euclidean space. It is also interesting that Houston appears to be "south" of Miami, Washington DC appears to be "north" of Chicago, and Los Angeles is not far south compared to Denver.

To further understand how the choice of distance can change the MDS results and corresponding plots, I raise the original distances to different powers. (Note this is raising each component of the distance matrix to a power, not matrix multiplying the distance matrix a number of times.) First, I raise each distance by a power of 2.5. The new graph is shown below.



Notably, the "map" is the very similar as the one in part a with only the axes scales changed. More importantly, however, is the fact that the cities mentioned above are closer to their normal representation. This map also seems to suggested that MDS is relatively robust to the choice in distance measurement.

Second, I take the square root of each distance and graph the results below.



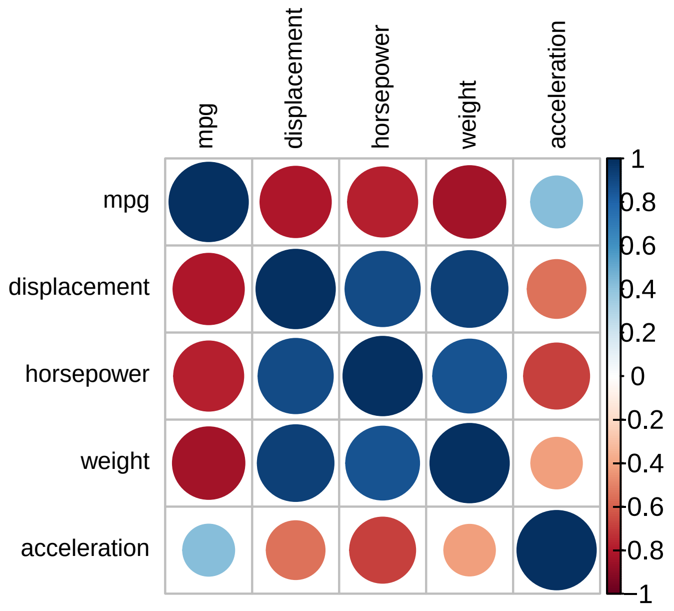
This third graph seems to contradict my statement about the MDS being robust to a choice in the distance measure. However, we also know that MDS is only unique up-to a rotation and both graphs are arguably rotations of the first. We can see that this graph has flipped the “north” and “south” directions compared to the first graph.

Question 3

Recall in homework 1 we were asked to analyze the auto-mpg.data using PCA. This week, we were asked to analyze this dataset using factor analysis and MDS. Before performing any data analysis, I clean the data by removing the “?”’s in horsepower and remove the NAs. Additionally, I change the variables cylinders, model year, and origin to factors. Finally, I create a different dataset with only the numerical variables which will be used for the factor analysis and MDS. Factor analysis can only be performed with all numerical data, and though MDS can handle categorical variables it gets tricky when the categorical variables are not binary. The categorical variables will still be used to help visualize aspects of the data.

Factor analysis for cars

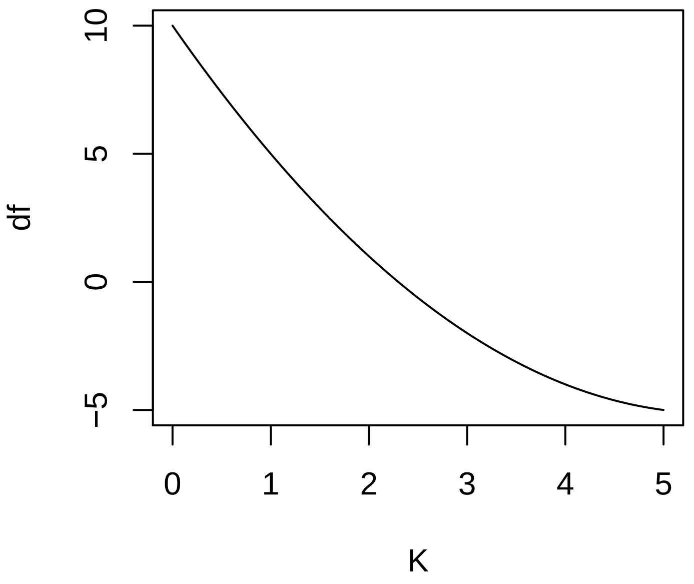
Before performing factor analysis, I examine what the correlation matrix of the variables is.



As we can see from the correlation matrix above, there are some distinct patterns. MPG and acceleration are somewhat positively correlated; whereas MPG and displacement, horsepower, and weight are strongly negatively correlated. Acceleration is somewhat strongly negatively correlated with horsepower and somewhat negatively correlated with displacement and weight. Finally, displacement, horsepower, and weight are all strongly positively correlated.

Next, we need to determine how many factors we should use for our factor analysis. Recall that $X_{p\times 1} = \Lambda_{p\times k}F_{k\times 1} + U_{p\times 1}$ where Λ is the matrix of the parameters, F is the common factors vector, and U is the error vector. Using the data we need to estimate Λ, F, and U given some assumptions (outlined in problem 1).

The degrees of freedom that we have to perform factor analysis is equal to $\frac{p(p+1)}{2} - (pK + p - \frac{K(K-1)}{2})$ where p is the number of parameters in the cars sub-dataset and K are the number of factors. Plugging in $p = 5$ and plotting the function, we see that the degrees of freedom is negative after 2 factors. This limits the number of factors that we could use to 2.



Next, we use hypothesis testing to determine whether 1 or 2 factors would be appropriate for the data.

$$H_0 : \text{model with } K \text{ common factors } H_a : \text{unconstrained model}$$

We want to fail to reject the null hypothesis at $\alpha = 0.05$.

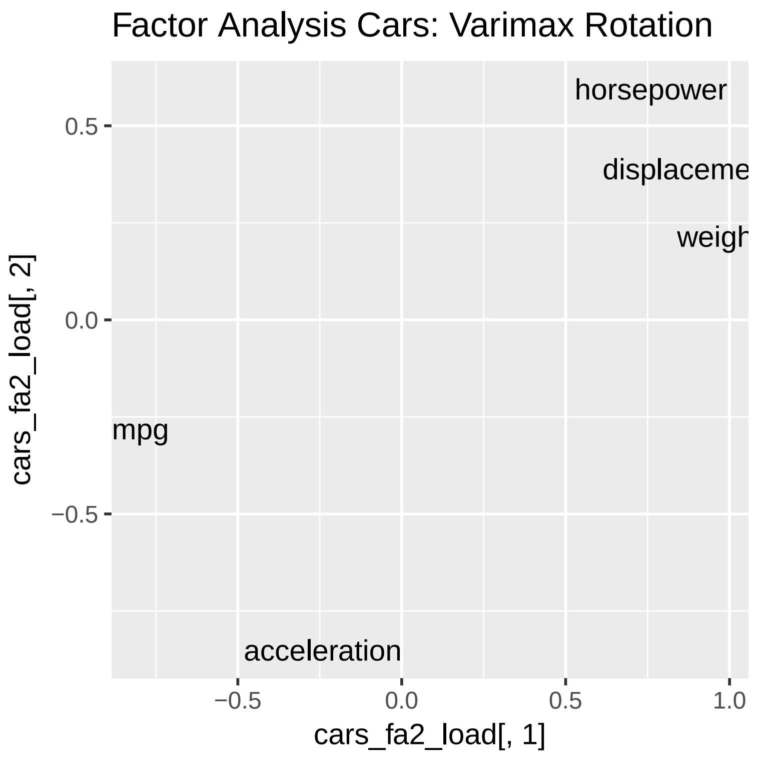
```
##
## Call:
## factanal(x = cars_num, factors = 1)
##
## Uniquenesses:
##          mpg displacement   horsepower       weight acceleration
##        0.291        0.050        0.151        0.092        0.697
##
## Loadings:
##              Factor1
## mpg          -0.842
## displacement  0.975
## horsepower    0.921
## weight        0.953
## acceleration -0.551
##
##                Factor1
## SS loadings      3.719
## Proportion Var   0.744
##
## Test of the hypothesis that 1 factor is sufficient.
## The chi square statistic is 261.33 on 5 degrees of freedom.
## The p-value is 2.04e-54
```

```
##
## Call:
## factanal(x = cars_num, factors = 2)
##
## Uniquenesses:
##          mpg displacement   horsepower       weight acceleration
##        0.286        0.080        0.068        0.015        0.218
```

```
##
## Loadings:
##              Factor1 Factor2
## mpg          -0.797  -0.281
## displacement  0.876   0.389
## horsepower    0.760   0.595
## weight        0.969   0.215
## acceleration -0.241  -0.851
##
##                Factor1 Factor2
## SS loadings      2.978   1.354
## Proportion Var   0.596   0.271
## Cumulative Var   0.596   0.866
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 1.15 on 1 degree of freedom.
## The p-value is 0.283
```

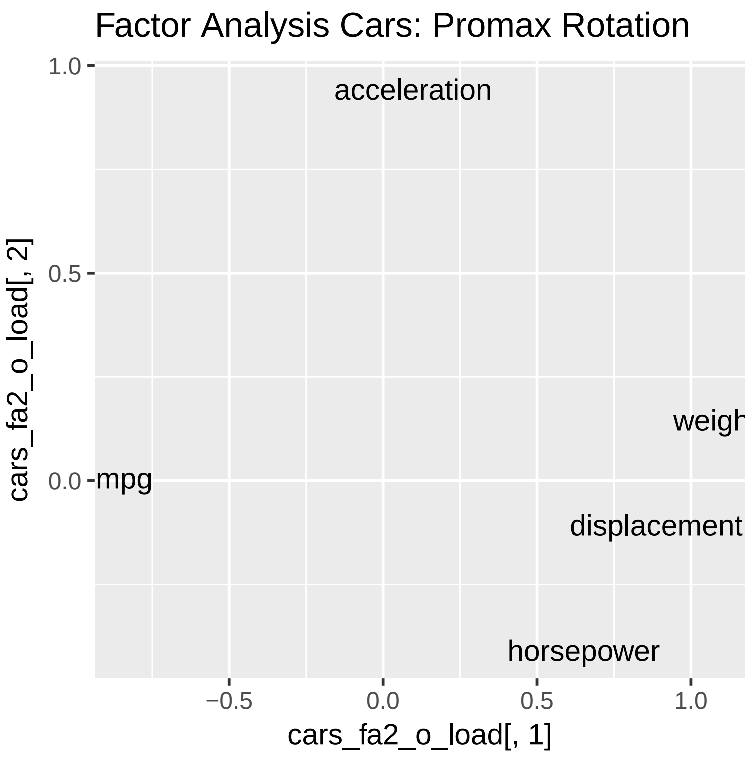
The model with 1 factor has a p-value less than $\alpha = 0.05$; therefore, we should reject the null hypothesis and try using 2 factors. The model with 2 factors has a p-value bigger than $\alpha = 0.05$, we fail to reject the null hypothesis and conclude that we should continue with two factors. This is similar to PCA in that we had previously concluded that 2 principal components was sufficient for the data.

Next, I compare factor analysis using different rotations: varimax, oblique, and none. I plot the corresponding factor loadings below. The loadings identify which variables have the largest effects on the factors; this is similar from PCA where the variable loadings identified the weights of each variable that was used to construct each component.



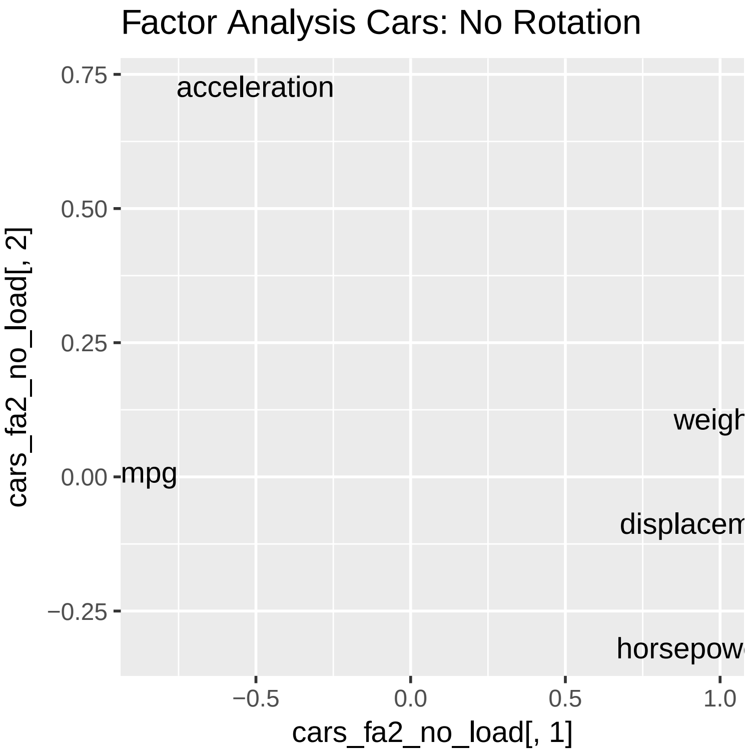
```
##
## Call:
## factanal(factors = 2, covmat = cov(cars_num), rotation = "promax")
```

```
##
## Uniquenesses:
##          mpg displacement   horsepower       weight acceleration
##        0.286        0.080        0.068        0.015        0.218
##
## Loadings:
##              Factor1 Factor2
## mpg          -0.841
## displacement  0.887  -0.107
## horsepower    0.652  -0.409
## weight        1.080   0.146
## acceleration          0.943
##
##                Factor1 Factor2
## SS loadings      3.094   1.090
## Proportion Var   0.619   0.218
## Cumulative Var   0.619   0.837
##
## Factor Correlations:
##         Factor1 Factor2
## Factor1   1.000  -0.639
## Factor2  -0.639   1.000
##
## The degrees of freedom for the model is 1 and the fit was 0.003
```



```
##
## Call:
## factanal(factors = 2, covmat = cov(cars_num), rotation = "none")
##
## Uniquenesses:
```

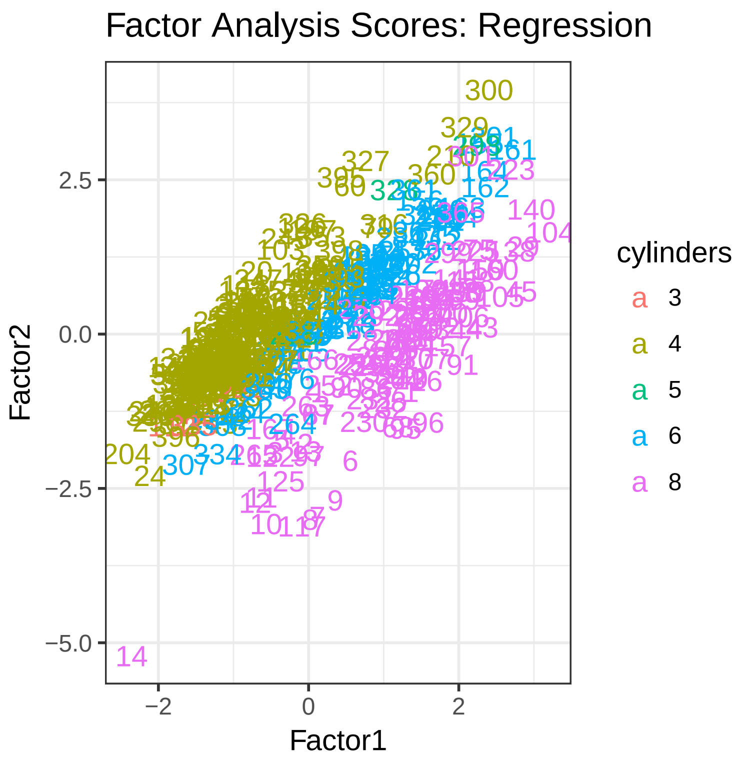
```
##          mpg displacement   horsepower       weight acceleration
##        0.286        0.080        0.068        0.015        0.218
##
## Loadings:
##              Factor1 Factor2
## mpg          -0.845
## displacement  0.955
## horsepower    0.911  -0.318
## weight        0.986   0.108
## acceleration -0.502   0.728
##
##                Factor1 Factor2
## SS loadings      3.682   0.650
## Proportion Var   0.736   0.130
## Cumulative Var   0.736   0.866
##
## The degrees of freedom for the model is 1 and the fit was 0.003
```

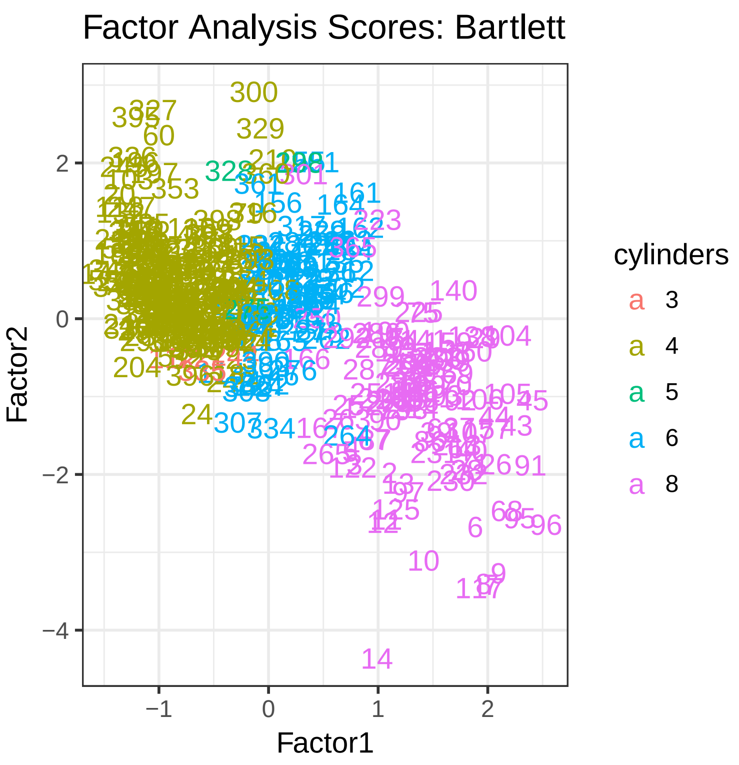


MPG, weight and displacement all strongly effect the first factor since their loadings are close to -1 or 1 in all of the graphs. Horsepower is also generally strongly effects the first factor but not as much, and acceleration does not effect the first factor much. However, horsepower and acceleration generally strongly effect the second factor since their loadings are close to -1 or 1 in all of the graphs. Again, this is similar to the first two principal components that we saw last week in that the first principal component had approximate equal weights of all the variables but with MPG and acceleration acting in the opposite direction as the others, and the second principal component dominated by acceleration.

Comparing the factor loadings across the different rotations it is clear that they are all pretty close in terms of loadings of each variable for both factors. However, the promax rotation exacerbates the extremes of the varimax rotation and in our case, makes it so that acceleration does not even effect the first factor and MPG does not effect the second factor. This is nice because we can interpret the first factor as almost equally weighted effects between displacement, horsepower, weight, and MPG on the first factor (with MPG acting in the opposite direction as the others) and the second factor being predominately affected by acceleration. For this reason, I stick with the Promax rotation for the examining the factor scores.

Below are the graphs of the factor scores for the promax rotation using Thompson's Method and Bartlett's Method. The scores are projecting the data onto the "factor directions" that we identified above. Additionally, I color code the points based on the number of cylinders the car has. I picked the number of cylinders to be the variable of interest because in the PC analyses performed last week, we found that the number of cylinders showed the best separation of the data.



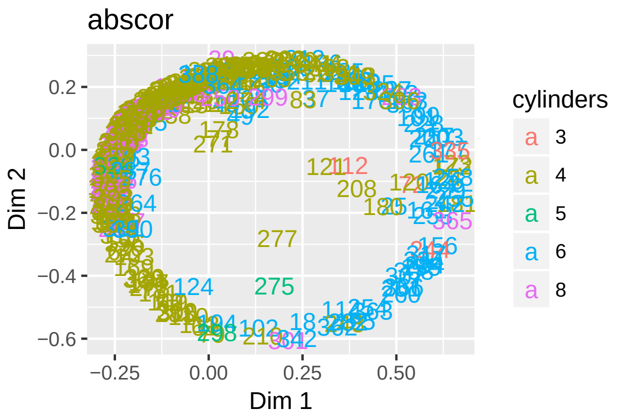


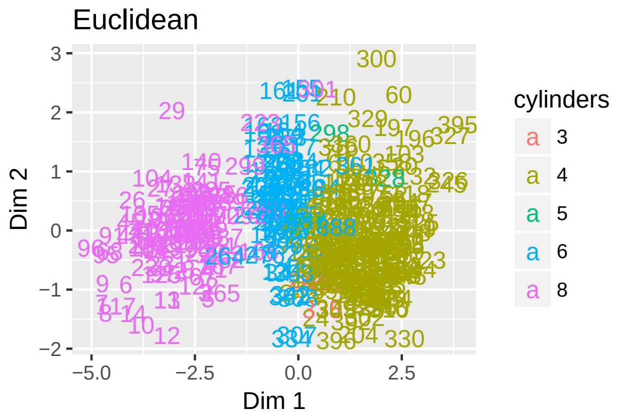
Bartlett's method appears to show the separation in the data slightly better than Thompson's regression method. However, the coloring of the number of cylinders shows a very distinct seperation of the data onto the factors.

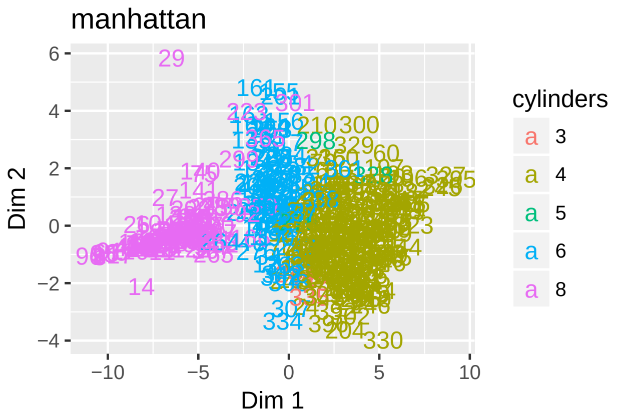
Multidimensional Scaling

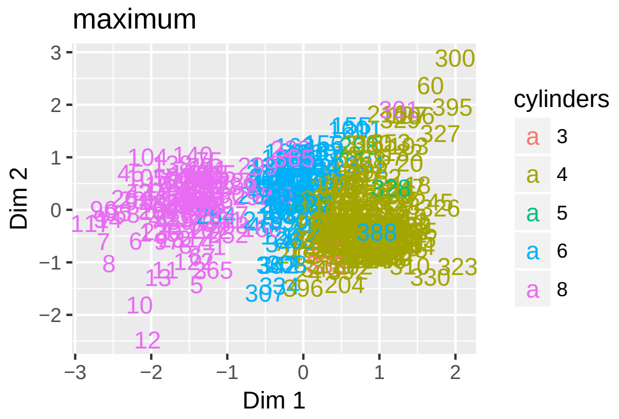
Finally, I perform MDS on the cars dataset. I want to examine the differences in the choice of distances so I perform MDS using the absolute value of the correlation matrix as the distance matrix, and the Euclidean, manhattan, maximum, and gower distances. The absolute value, Euclidean, manhattan, and maximum differences all only use the numerical variables from the cars dataset. The gower distance also includes the categorical variables of cylinders, model year, and origin. The gower distance is able to handle categorical variables by scaling them to take values between 0 and 1 and then taking a weighted average of these variables so that the categorical variables retain some of their meaning. For instance, the number of cylinders is between 3 and 8; gower allows cars with 3 cylinders to be "near" cars with 4 cylinders and "further way" from cars with 8 cylinders. This is different from PCA since PCA cannot handle categorical variables.

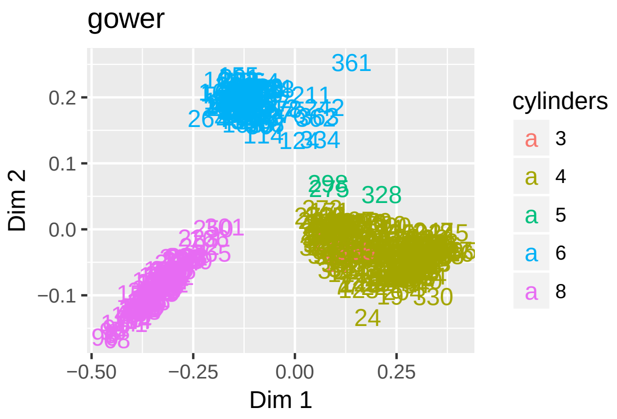
Below I graph the results of each of these distances.



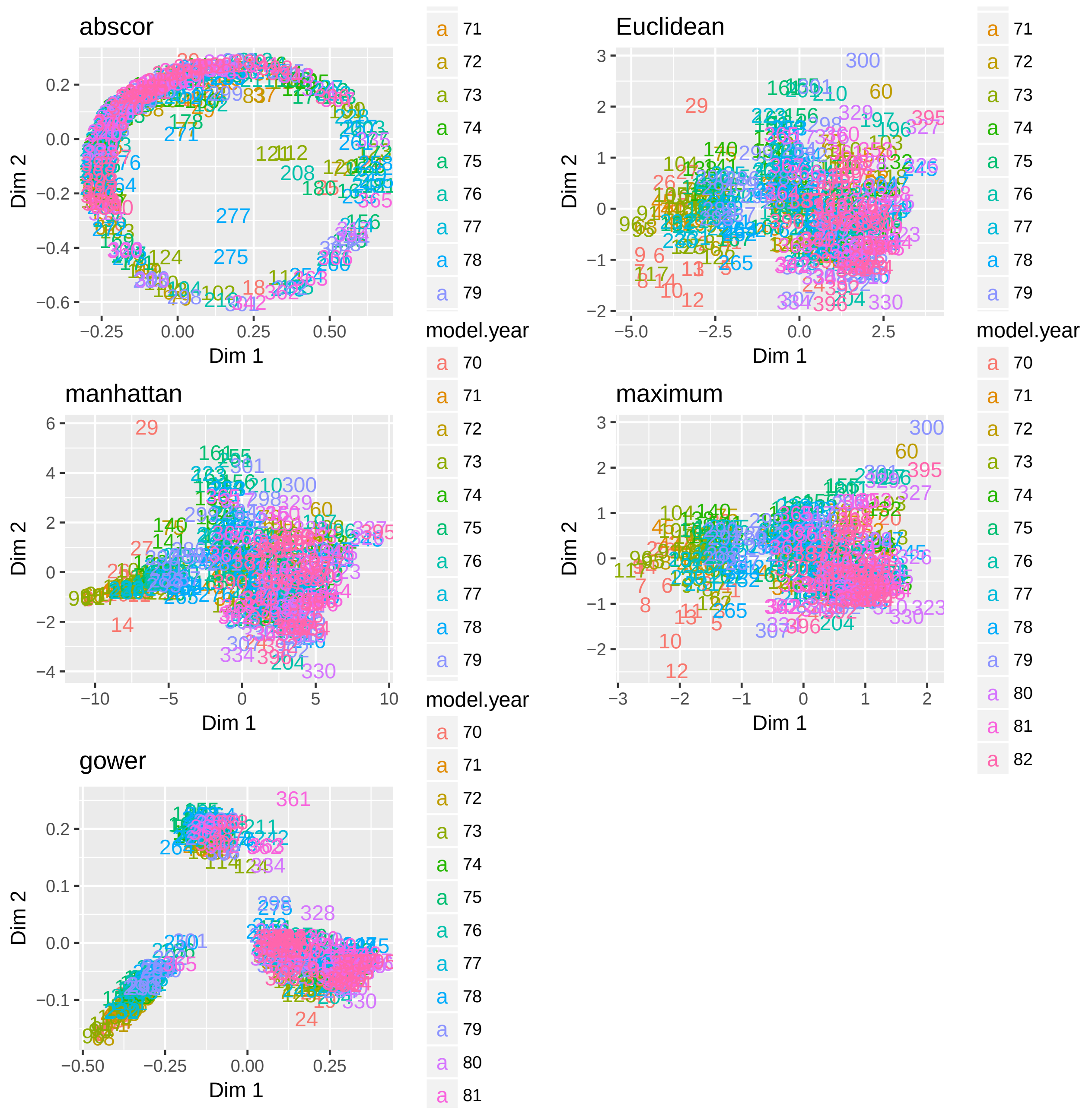








The abscor plot is interesting but doesn't show a lot of separation in the data, whereas the others do show at least some amount of separation based on the number of cylinders. It is interesting to note that observation 29 appears to be some outlier in the Euclidean, manhattan, and gower distances but not in the maximum distance. Comparing the graphs, the manhattan and the gower distances are similar, but the gower distance does a much better job of separating the data since it actaully takes into account the categorical variables, and the Euclidean and maximum graphs are also very similar. We note that since the (numerical) data are in a Euclidean space, the classical MDS method (performed above) using Euclidean distances is equivalent to performing PCA analysis. Finally, we note that using cylinders as the coloring method allows us to greatly separate the data as compared to other categorical variables (see below for an example using model.year).



```r
library(knitr)
knitr::opts_chunk$set(echo = FALSE, warning = FALSE, fig.align = "center", fig.height = 4, fig.width = 

############################################
# Set working directory and load libraries
############################################

setwd("~/Documents/UMich/Classes/2018 Spring/STATS 503/Homework/HW2/")
library(ggplot2)
library(corrplot)
library(cluster)
library(gridExtra)
lambda_1 <- matrix(c(1,2,1,3,0,-1,2,-4), nrow = 4)
lambda_1

sigma_1 <- matrix(c(2,2,1,3, 2,5.5,0,10, 1,0,5.5,-5, 3,10,-5,27), nrow = 4)
sigma_1

sigma_1_inv <- solve(sigma_1)
sigma_1_inv

x_i <- matrix(c(1,1,1,1))
x_i

f_i <- t(lambda_1) %*% sigma_1_inv %*% x_i
f_i


###########################################
# Code-it-yourself MDS
###########################################

usd <- as.matrix(UScitiesD)
# currently the data is in miles between each pairwise of cities

# Initial choice of distance is the same

## Steps for creating the MDS
# 1) Convert D to a Gram matrix using the double centering approach

# Create the identity matrix
I = diag(nrow(usd))

# Create the column of ones
one = as.matrix(rep(1, nrow(usd)))

# Double center the data
gram <- function(distance) {
  (-1/2) * (I - (1/nrow(usd))*(one %*% t(one))) %*% distance %*% (I - (1/nrow(usd))*(one %*% t(one)))
}

gram_usd <- gram(usd)

# Pick the number of dimensions that we want

mds_mine <- function(gram, k) {

  # 2) Find the eigendecomposition of G = U(lambda)U^t

  eigen_gram_usd <- eigen(gram)

  # We want just the top 2 eigenvalues and vectors
  # Keep the negative??
  # Why does it make it negative??
  cities_mds <- as.data.frame(NA)

  for (i in 1:k) {
    z1 <- sqrt(eigen_gram_usd$values[i]) * eigen_gram_usd$vectors[,i]
    z1 <- as.data.frame(z1)
    cities_mds <- cbind(cities_mds, z1)
  }

  # remove the column of NAs
  cities_mds <- cities_mds[,-1]

  # add rownames
  rownames(cities_mds) <- rownames(usd)
  colnames(cities_mds) <- sapply(1:k, function(i) {paste("z", i, sep = "")})

  qplot(x = -cities_mds$z1, y = -cities_mds$z2, label = row.names(cities_mds), geom = "text",
        xlab = "Projection 1 (Z1)", ylab = "Projection 2 (Z2)")

}

mds_mine(gram_usd, 2)


##########################################
# Choose a different choice of distance and recalculate MDS
##########################################

# Different choice of distance could be a power that is bigger than one
alpha = 2.5

d1_usd <- usd ^ alpha

gram_d1_usd <- gram(d1_usd)

mds_mine(gram_d1_usd, 2)


##########################################
# Make a 3rd choice of distance and recalculate MDS
##########################################

# Different choice of distance could be a power that is bigger than one
alpha = (.5)

d2_usd <- usd ^ alpha

gram_d2_usd <- gram(d2_usd)

mds_mine(gram_d2_usd, 2)


##########################################
# Bring in auto-mpg data from HW 1
```

```
##########################################

# Read in data and clean
cars <- read.table("../HW1/auto-mpg.data", col.names = c("mpg", "cylinders", "displacement", "horsepowe

# contains "?"
cars$horsepower <- as.numeric(as.character(cars$horsepower))

# Converting to factors
cars$cylinders <- as.factor(cars$cylinders)
cars$model.year <- as.factor(cars$model.year)
cars$origin <- as.factor(cars$origin)

# Remove NA's for further analysis
cars_na.o <- na.omit(cars)

# Keep numerical values of cars (factor analysis can only handle numerical values)
cars_num <- cars_na.o[,c(-2, -7:-9)]


# Look at colorful correlation matrix
corrplot(cor(cars_num), tl.col="black", tl.cex=0.75)


## recall X = Lambda x F + U, where Lambda is the matrix of parameters, F is the common factors and U a
## E(F)=E(U)=0
## Cov(x) = sigma-hat = lambda-hat(lambda-hat-t) + (psi-hat)

# Number of parameters in sigma: (p)*(p+1)/2 = 15
# degrees of freedom are (p)*(p+1)/2 - (pK+p-K(K-1)/2)
# solving for K we find K = 3 is too many
df = function(x) (5*(5+1)/2 - (5*x + 5 - x*(x-1)/2))
curve(df, 0, 5, xlab = "K", ylab = "df")

####
#The fit of the model is the ratio of the maximized log likelihood under $H_a$ and the same under $H_0$
####
# Use hypothesis testing to determine the number of factors
# Null hypothesis: model with K common factors
# alternative: unconstrained model
# fit tells me whether I can reject or fail to reject H0

# loadings with varimax rotation; orthogonal rotation
# varimax tries to make large things larger and small things smaller; tends to have larger values in th
# Factor loadings indicate how much a factor explains a variable
# Try with 1
cars_fa1 = factanal(cars_num, factors = 1)
print(cars_fa1)

# Try with 2
cars_fa2 = factanal(cars_num, factors = 2)
print(cars_fa2)
```

```r
cars_fa2_load <- cars_fa2$loadings
qplot(x=cars_fa2_load[,1], y=cars_fa2_load[,2], label=rownames(cars_fa2_load),
      geom="text", main = "Factor Analysis Cars: Varimax Rotation")

# See what happens when we use the oblique rotation
# oblique rotation; makes it sparse and helps with interpretation; tends to have larger values in the f
cars_fa2_o = factanal(covmat = cov(cars_num), factors = 2, rotation = "promax")
print(cars_fa2_o)

cars_fa2_o_load <- cars_fa2_o$loadings
qplot(x=cars_fa2_o_load[,1], y=cars_fa2_o_load[,2],
      label=rownames(cars_fa2_o_load), geom="text",
      main = "Factor Analysis Cars: Promax Rotation")

# See what happens when we have no rotation
cars_fa2_no = factanal(covmat = cov(cars_num), factors = 2, rotation = "none")
print(cars_fa2_no)

cars_fa2_no_load <- cars_fa2_no$loadings
qplot(x=cars_fa2_no_load[,1], y=cars_fa2_no_load[,2],
      label=rownames(cars_fa2_no_load), geom="text",
      main = "Factor Analysis Cars: No Rotation")


### with scores
# Factor scores identify the relative weight of each variable in the component in a factor analysis
# Sticking with promax rotation

# bayesian approach: Thompson's Method
fa_scores_reg = factanal(x = cars_num, factors = 2, rotation = "promax", scores = 'regression')
re_scores = fa_scores_reg$scores
df_reg = data.frame(Factor1 = re_scores[,1], Factor2 = re_scores[,2])
df_reg$cylinders = cars_na.o$cylinders
ggplot(df_reg,aes(x=Factor1,y=Factor2))+geom_text(aes(label=rownames(df_reg), col = cylinders))+
  theme_bw() + ggtitle("Factor Analysis Scores: Regression")

# Include colors by different categorical variables to determine maximum separation

# very similar to regression method
fa_scores_b = factanal(x = cars_num, factors = 2, rotation = "promax", scores = 'Bartlett')
b_scores = fa_scores_b$scores
b_df = data.frame(Factor1 = b_scores[,1], Factor2 = b_scores[,2])
b_df$cylinders = cars_na.o$cylinders
ggplot(b_df,aes(x=Factor1,y=Factor2))+geom_text(aes(label=rownames(b_df), col = cylinders))+
  theme_bw() + ggtitle("Factor Analysis Scores: Bartlett")


# mds doesn't need to only have numerical variables-- use gowers distance
# start with dataset without the NAs

#Perform MDS
# dist calculates the distances between cars
# Euclidean is the default method
# MDS requires it to be scoared first
cars_scaled <- scale(cars_num)
cars_scaled_gower <- as.data.frame(cars_scaled)
cars_scaled_gower$cylinders <- cars_na.o$cylinders
cars_scaled_gower$model.year <- cars_na.o$model.year
cars_scaled_gower$origin <- cars_na.o$origin

# Use a bunch of different scaling methods to compare

dist_cars_abscor <- 1-abs(cor(t(cars_scaled))) #variables that are correlated are closer together than
dist_cars_eu <- dist(cars_scaled, method = "euclidean")
dist_cars_man <- dist(cars_scaled, method = "manhattan")
dist_cars_sup <-  dist(cars_scaled, method = "maximum")
dist_cars_gower <-  daisy(cars_scaled_gower, metric = "gower")

# Perform MDS on each of the distance methods

cmd_cars_abscor <- as.data.frame(cmdscale(dist_cars_abscor))
cmd_cars_eu <- as.data.frame(cmdscale(dist_cars_eu))
cmd_cars_man <- as.data.frame(cmdscale(dist_cars_man))
cmd_cars_sup <- as.data.frame(cmdscale(dist_cars_sup))
cmd_cars_gower <- as.data.frame(cmdscale(dist_cars_gower))

# add cylinders to all the MDS data
cmd_cars_abscor$cylinders <- cars_na.o$cylinders
cmd_cars_eu$cylinders <- cars_na.o$cylinders
cmd_cars_man$cylinders <- cars_na.o$cylinders
cmd_cars_sup$cylinders <- cars_na.o$cylinders
cmd_cars_gower$cylinders <- cars_na.o$cylinders

# Create a bunch of plots
# combine them all together in a list
# label the plots
abscor_plot = ggplot(cmd_cars_abscor, aes(x = V1, y = V2)) +
    geom_text(aes(label=rownames(cmd_cars_abscor), col = cylinders)) + labs(title = "abscor") +
    xlab("Dim 1") + ylab("Dim 2")

eu_plot = ggplot(cmd_cars_eu, aes(x = V1, y = V2)) +
    geom_text(aes(label=rownames(cmd_cars_eu), col = cylinders)) + labs(title = "Euclidean") +
    xlab("Dim 1") + ylab("Dim 2")

man_plot = ggplot(cmd_cars_man, aes(x = V1, y = V2)) +
    geom_text(aes(label=rownames(cmd_cars_man), col = cylinders)) + labs(title = "manhattan") +
    xlab("Dim 1") + ylab("Dim 2")

sup_plot = ggplot(cmd_cars_sup, aes(x = V1, y = V2)) +
    geom_text(aes(label=rownames(cmd_cars_sup), col = cylinders)) + labs(title = "maximum") +
    xlab("Dim 1") + ylab("Dim 2")

gower_plot = ggplot(cmd_cars_gower, aes(x = V1, y = V2)) +
    geom_text(aes(label=rownames(cmd_cars_gower), col = cylinders)) + labs(title = "gower") +
    xlab("Dim 1") + ylab("Dim 2")
```

```r
plot_list <- list(abscor_plot, eu_plot, man_plot, sup_plot, gower_plot)

# create plots and arrange them
do.call("grid.arrange", c(plot_list, nrow = 3, ncol =2))


# Create a bunch of plots
# combine them all together in a list
# label the plots

# add model.year to all the MDS data
cmd_cars_abscor$model.year <- cars_na.o$model.year
cmd_cars_eu$model.year <- cars_na.o$model.year
cmd_cars_man$model.year <- cars_na.o$model.year
cmd_cars_sup$model.year <- cars_na.o$model.year
cmd_cars_gower$model.year <- cars_na.o$model.year

abscor_plot_m = ggplot(cmd_cars_abscor, aes(x = V1, y = V2)) +
    geom_text(aes(label=rownames(cmd_cars_abscor), col = model.year)) + labs(title = "abscor") +
    xlab("Dim 1") + ylab("Dim 2")

eu_plot_m = ggplot(cmd_cars_eu, aes(x = V1, y = V2)) +
    geom_text(aes(label=rownames(cmd_cars_eu), col = model.year)) + labs(title = "Euclidean") +
    xlab("Dim 1") + ylab("Dim 2")

man_plot_m = ggplot(cmd_cars_man, aes(x = V1, y = V2)) +
    geom_text(aes(label=rownames(cmd_cars_man), col = model.year)) + labs(title = "manhattan") +
    xlab("Dim 1") + ylab("Dim 2")

sup_plot_m = ggplot(cmd_cars_sup, aes(x = V1, y = V2)) +
    geom_text(aes(label=rownames(cmd_cars_sup), col = model.year)) + labs(title = "maximum") +
    xlab("Dim 1") + ylab("Dim 2")

gower_plot_m = ggplot(cmd_cars_gower, aes(x = V1, y = V2)) +
    geom_text(aes(label=rownames(cmd_cars_gower), col = model.year)) + labs(title = "gower") +
    xlab("Dim 1") + ylab("Dim 2")

plot_list_m <- list(abscor_plot_m, eu_plot_m, man_plot_m, sup_plot_m, gower_plot_m)

# create plots and arrange them
do.call("grid.arrange", c(plot_list_m, nrow = 3, ncol =2))
```