lClass Project #2

Clustering and Classification

Rubric

Fall 2022

For the purposes of illustration, we will use the auto-mpg.data set from the UCI ML Repository:

auto-mpg.names is the description of the data set:

**Source:**

This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University. The dataset was used in the 1983 American Statistical Association Exposition.

**Data Set Information:**

This dataset is a slightly modified version of the dataset provided in the StatLib library. In line with the use by Ross Quinlan (1993) in predicting the attribute "mpg", 8 of the original instances were removed because they had unknown values for the "mpg" attribute. The original dataset is available in the file "auto-mpg.data-original".   
  
"The data concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of 3 multivalued discrete and 5 continuous attributes." (Quinlan, 1993)

**Attribute Information:**

1. mpg: continuous   
2. cylinders: multi-valued discrete   
3. displacement: continuous   
4. horsepower: continuous   
5. weight: continuous   
6. acceleration: continuous   
7. model year: multi-valued discrete   
8. origin: multi-valued discrete   
9. car name: string (unique for each instance)

**Relevant Papers:**

Quinlan, R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Conference of Machine Learning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.   
[[Web Link]](http://rexa.info/paper/9eac95c36cbec1ef4954d3491df1ea0a1d09c6f0)

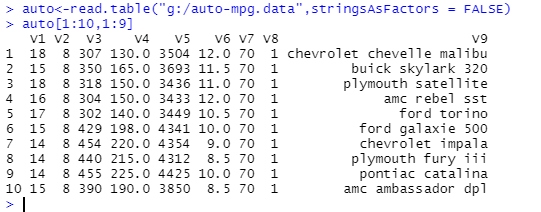
8. Missing Attribute Values: horsepower has 6 missing values.

…………….

Processing auto-mpg.data:

The data is a table of 9 fields by some number of rows.

Read.table reads the source file and creates a data frame

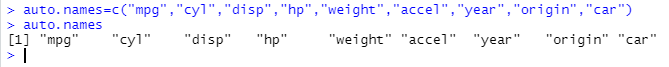


By using stringsAsFactors = FALSE, we ensure that no factors get inserted into the table. Factors cause problems with other functions.

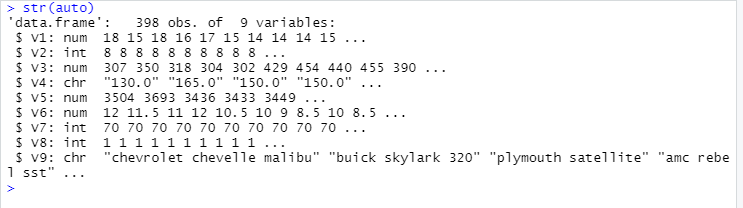
Note that read.table automatically assigns arbitrary variable names V1 through number of variables.

To assign variables names, we assign the names to variable x, check the length, then assign to names(auto).

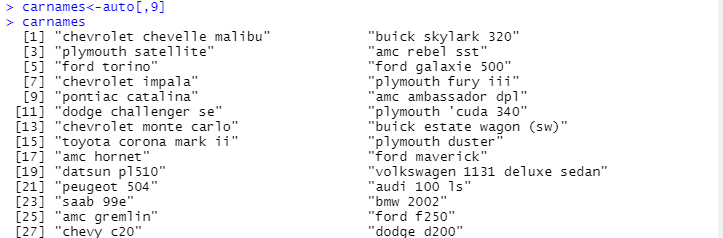
Assign car names to a vector:



Use “str” to view the structure of an object in R!! It is a data frame.

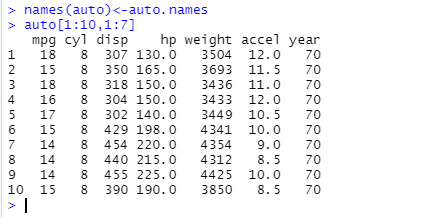


Extract carnames from the data frame:

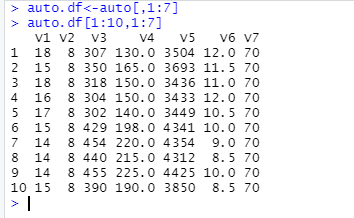


Notice V4 is of type “chr”.

Associate names of columns with the columns in the data frame:

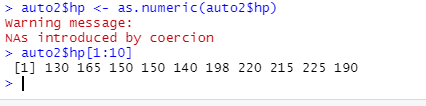


Strip off the last two columns because they do not contribute to analysis. Origin is sort of meaningless, year contributes nothing to mpg,and we have already saved car names in a separate vector.



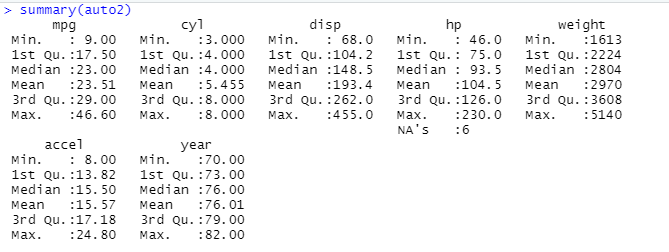
But, there are character values we need to convert to numeric. So, copy auto.df to auto2.

Use as.numeric to convert characters.



Now, all values are numbers.

Get a summary of the 7 variables for auto:



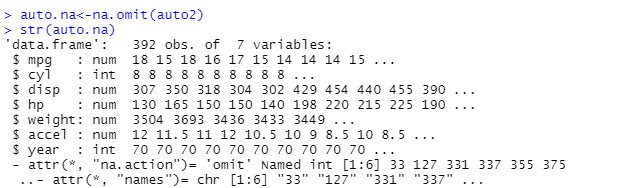
Note that six HP values are NAs. The NA rows do not participate in the computation.

We don’t have enough information replace those values so:

a. we can delete those rows. This might not be a good idea given the number of rows that have NAs, because the other columns may have useful information.

b. we can replace the NA values by the mean of the column, which probably skews the results for that column a bit. Again, if there are lots of rows with NAs, then this approach pushes the analysis towards the mean of the columns in which the NAs occur.

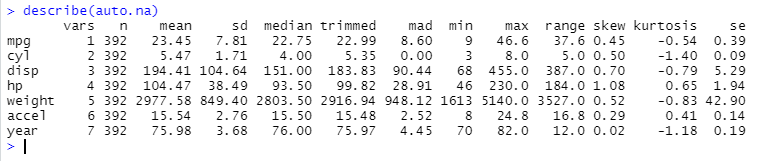
Use na.omit(<dataframe>) to remove rows that have NAs in them.



And, it tells us which rows were deleted. If we have other information, we could go back and fill in the values for the HP variable in those rows.

Install packages ‘sna’ and ‘psych’.

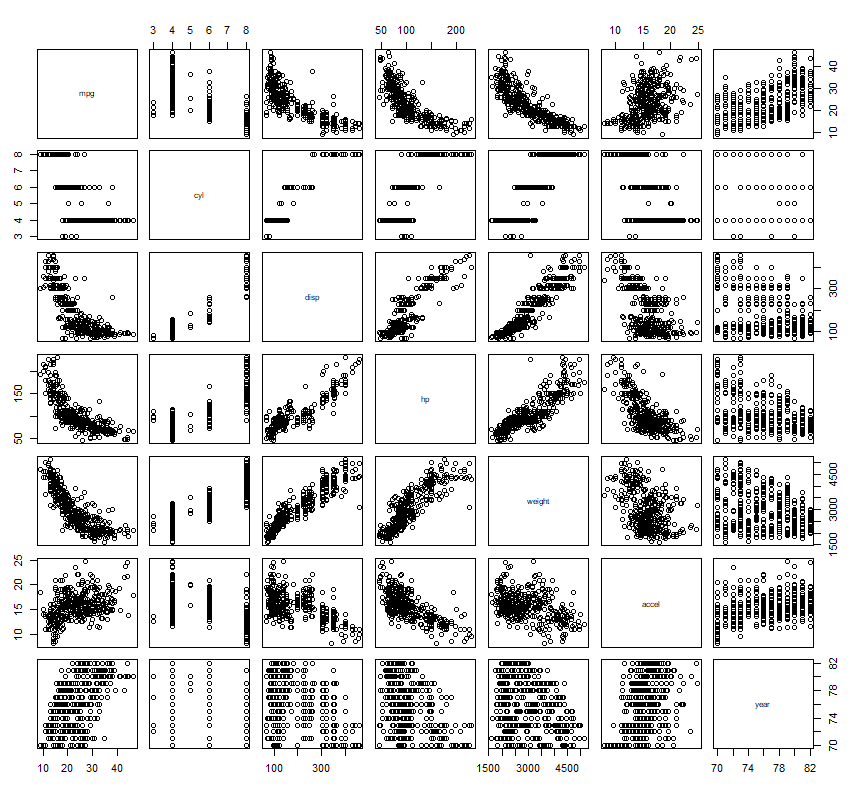
And, describe the statistics of the 7 variables:



Look up skew and kurtosis on Wikipedia.

Note that the “skew” computed by describe is for each variable, whereas the “skew and “kurtosis” functions in package “e1071” are multivariate.

Plotting the 7 variables in a pairwise manner:



Look for linear or near linear relationships.

So, mpg vs. disp, mpg vs. hp, and mpg vs. weight all have distinct relationships, e.g., as one variable changes, so does the other in a distinct way.

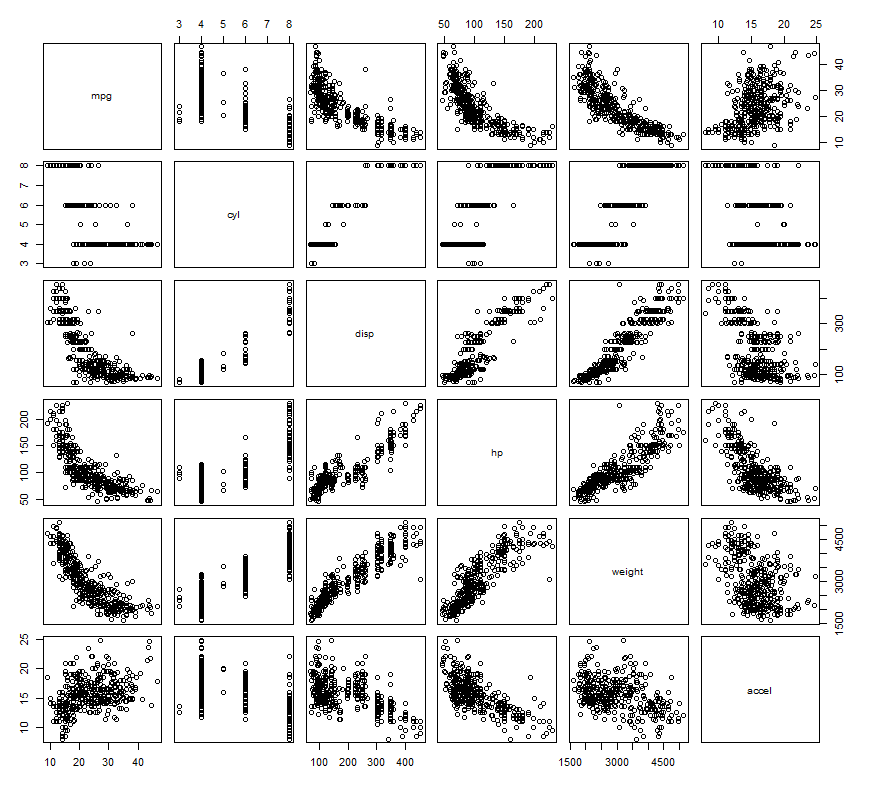
Note the curvature in the plots. This indicates that is probably an order-2 (e.g., a power of 2 factor) in the equation describing the relationship of the two variables.

Mpg vs. accel seems to be fuzzy. The relationship may be dependent on the efficiency of the car for which we have no data

Cyl vs. <other variables> doesn’t help a lot. As the number of cylinders goes up, the displacement goes up to accommodate them. Year vs. anything does not seem to have any relationship.

So, considering removing year from further consideration.





Note that disp vs. hp and disp vs. weight have very nice linear relationships.

Note that weight vs. acceleration seems to have a negative relationship, but it is not at all clear. It is a non-linear relationship, but other factors, which are not measured in this data set will come into play.

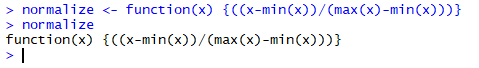
Keep these plots in mind because when we get to linear modeling we are going to look at these patterns.

Scaling: we normalize so that larger values for certain variables don’t overwhelm the smaller values of other variables.

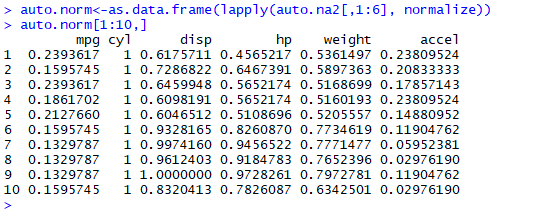
Consider two ways to normalize:

Min-Max Normalization:

* A normalization strategy which linearly transforms x to y= (x-min)/(max-min)
* Where min and max are the minimum and maximum values in X, where X is the set of observed values of x.
* It can be easily seen that when x=min, then y=0, and. When x=max, then y=1.

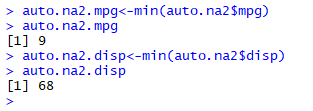


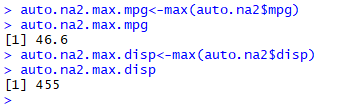
Scaling with normalize: The lapply() method applies to the entire data frame and all selected columns.

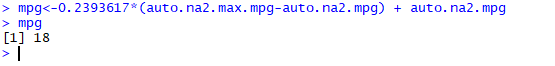


To get back to the original values, later:

x = norm(x) \* (max(x)-min(x) ) + min(x).



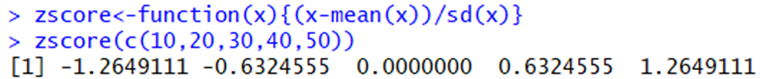


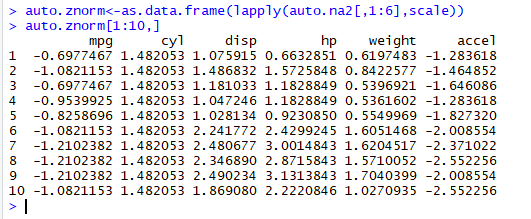


Voila!

**Z-score normalization**:

* All elements of the input vector are transformed into the output vector whose mean is approximately 0 while the standard deviation is in a range close to 1
* Scale() handles this for you





To get back to x, we use x = sd(x) + mean(x).

Try this with your data set.

Compare the normalization results of min-max vs. z-score normalization.

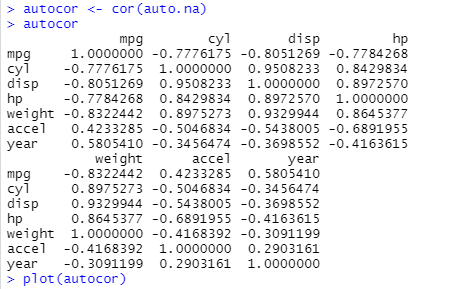
Which might be better for particular types of analysis?

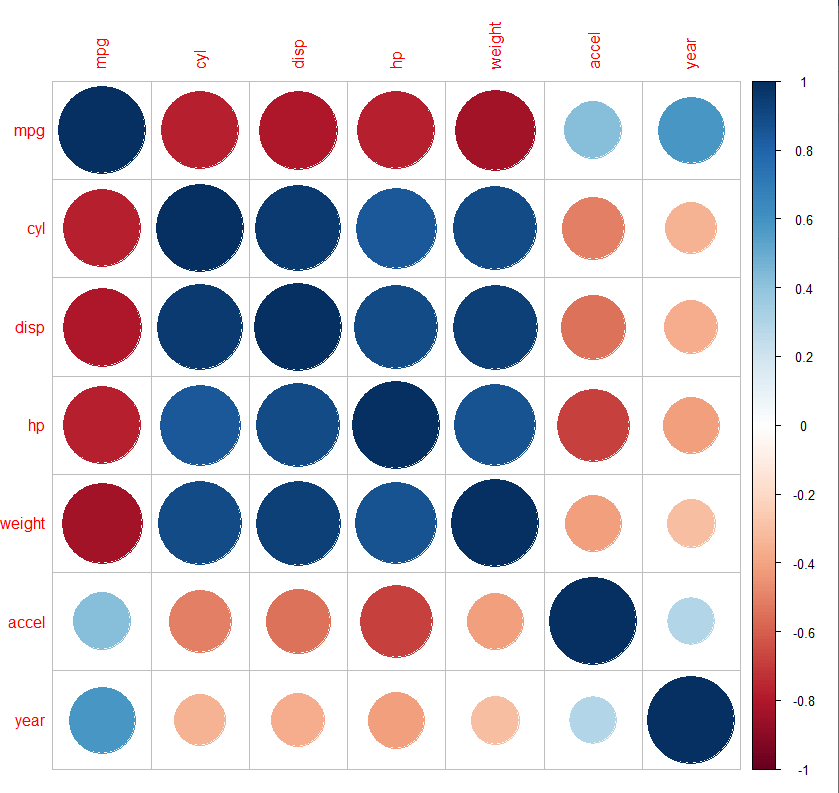
Install “corrplot”

A graphical display of a correlation matrix, confidence interval. The details are paid great attention to. It can also visualize a general matrix by setting is.corr = FALSE.

The areas of circles or squares show the absolute value of corresponding correlation coefficients.

See the documentation for additional parameters. Experiment with drawing a dew correlation plots.





**Kmeans Clustering**:

Read the description via RStudio!!

Here is the method invocation:

object = kmeans( x, // numeric matrix of data

centers, // the number of centers k = ‘n’

iter.max = 10, // maximum number of iterations to use

nstart = 1, // number of random sets

algorithm = c(Hartigan-Wong),

trace=FALSE) // only used with Hartigan-Wong

You may also choose: ", "Lloyd", "Forgy", "MacQueen" as algorithm choices.

fitted(object, method = c("centers", "classes"), ...)

where:

"centers" causes fitted to return cluster centers (one for each input point) and "classes" causes fitted to return a vector of class assignments.

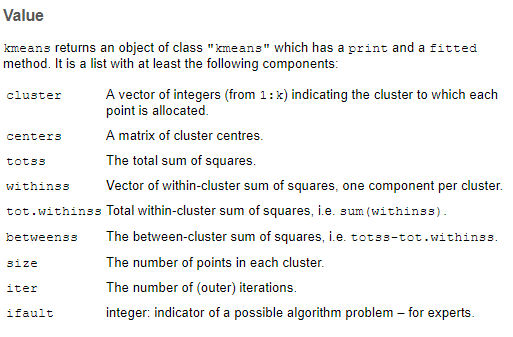
Some things to think about:

* Where does the distance d come from? For kmeans below, it is defined within the program.
* What if we used a different distance measure?
* How do we choose the best distance?
* How do we choose k?
* Will it converge?

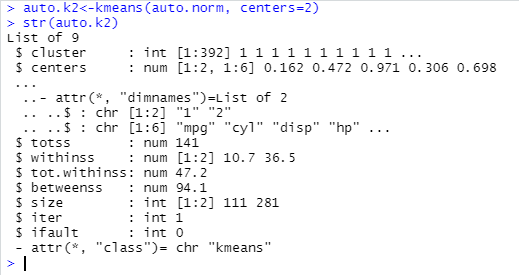
The default iter.max is 10, but can be overridden by the user.

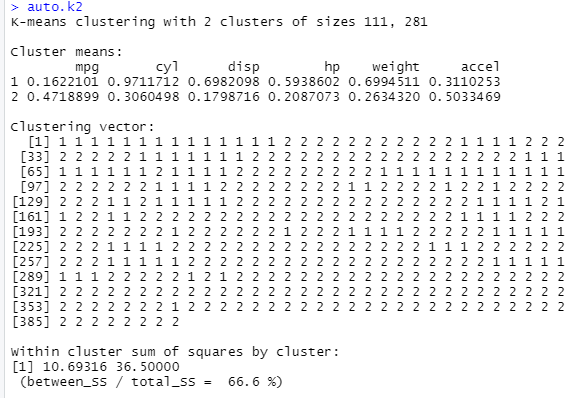
nstart specifies the number of random centers to be initially selected.

Usually, center is a number indicating the number of clusters we expect and nstart is equal to centers.



With k =2:



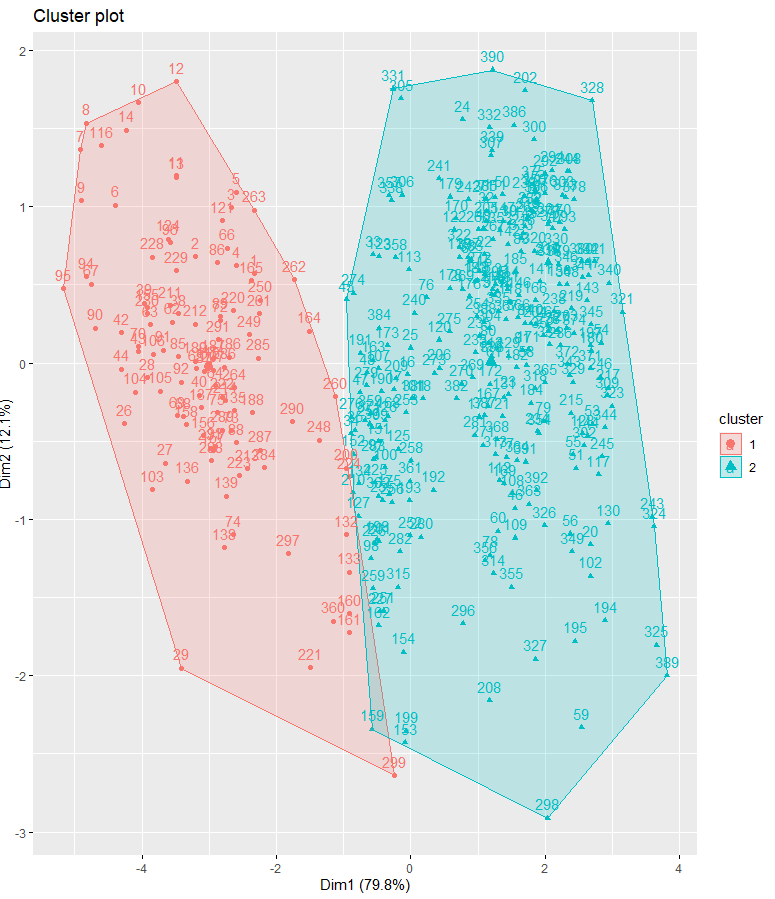


Using factoextra: factoextra::fviz\_cluster(auto.k2,auto.norm)

Well, lots of overlap using two clusters. But, some clear distinctions on left and right.

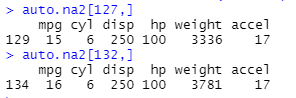
You need to go back to the original data to determine what characterizes the two clusters, particularly the non-overlapping parts.

So, write a small function to extract cluster 1 objects from auto.k2. Make it general so that it applies to any cluster number for a given execution of kmeans with different clusters.



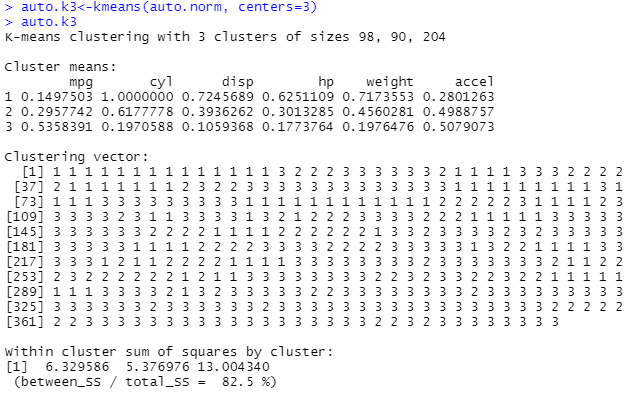
So, we see for two clusters some overlap for some vehicles. To determine why this is so, we would have to go back and examine the records near the overlap and see what the values of the attributes are.

So, for records 127 and 132, we see:



So, very similar except for disparity in weight.

With k = 3:



So, between\_ss/total\_ss has improved by about 24%.

Using factoextra::fviz\_cluster(auto.k3,auto.norm)

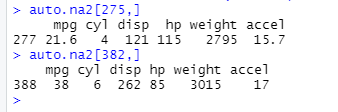


Well, still overlapping, but some clearer separation.

Better separation between first two clusters, but some overlap between

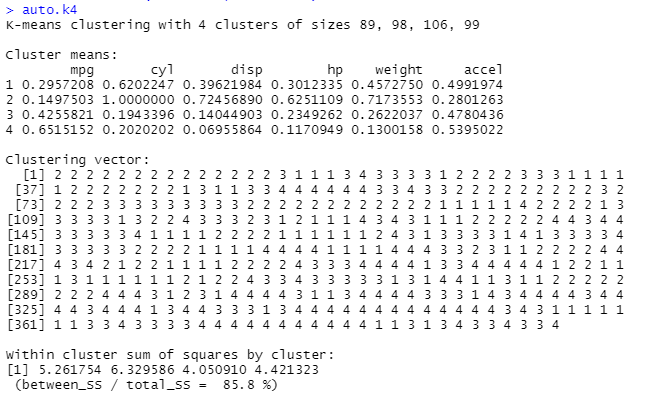
cluster 2 and 3. Records 127 and 132 are clearly within cluster 3.

So, let’s look at records 275 and 328 at the left edge of cluster 3:



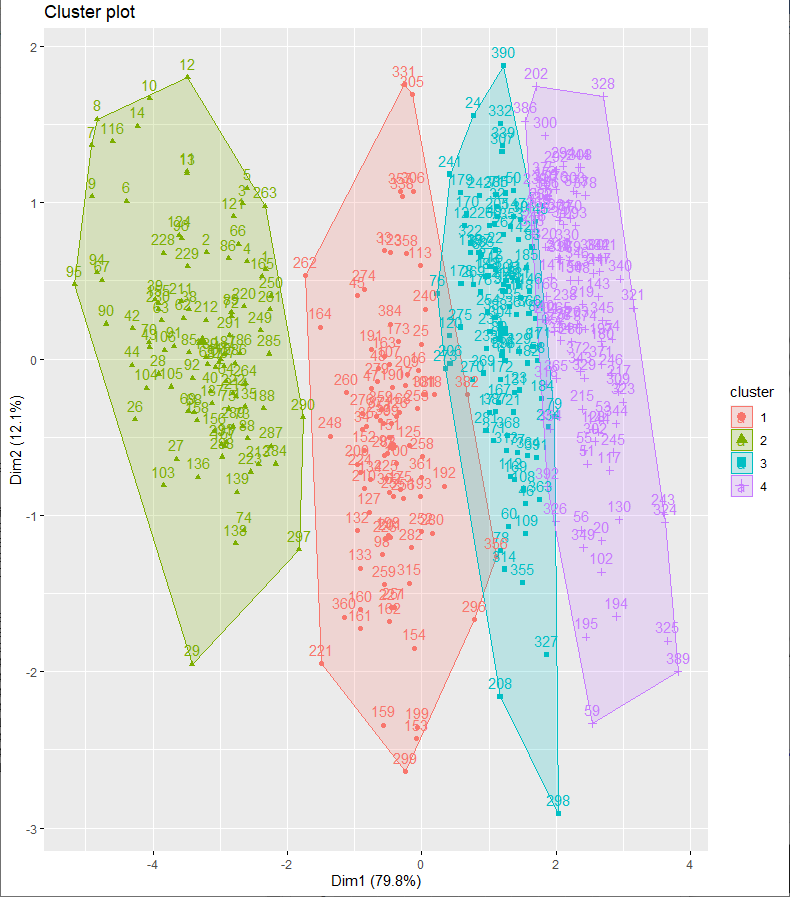
Record 275 clear in cluster 3, but record 382 looks right on the edge of cluster 2.

With k = 4:



So, this is only 4% improvement in between\_ss/total\_ss.

Using factoextra::fviz\_cluster(auto.k4,auto.norm)



So, what are seeing here?

We have distinct clusters, but the boundaries of the clusters 2,3, and 4, given the data we have, are not distinct.

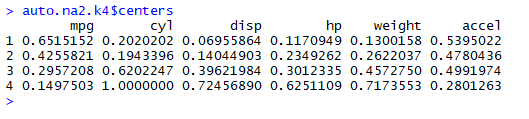
As I mentioned in class, the plotting algorithm is drawing the convex

hull around the data points comprising a cluster.

Examine the size of the clusters. Clusters that are too large or too small are not likely to be very useful because they lack descriptive power. So, for k =4, we see the cluster sizes are 99, 106, 89, and 88. Compared to k=2 or k=3, this is a homogenous distribution.

Now, examine the cluster centroids.

Look for the largest number in each column. Indicates that is most important cluster for that attribute.



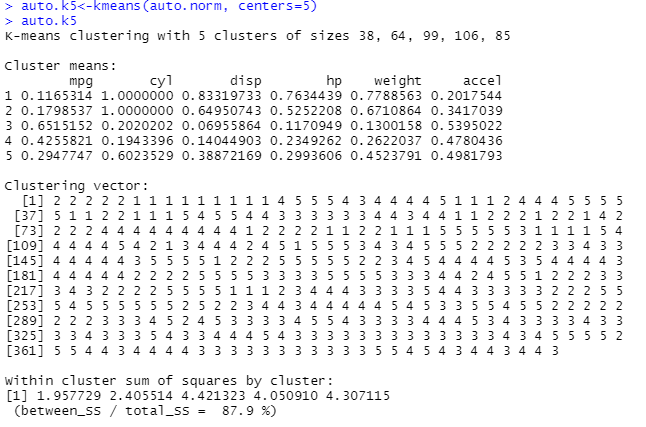
So, mpg-1, cyl-4, disp-4, hp-4, weigh-4, accel-1.

If you are a car afiocionado, then cyl, disp, hp, and weight are all related.

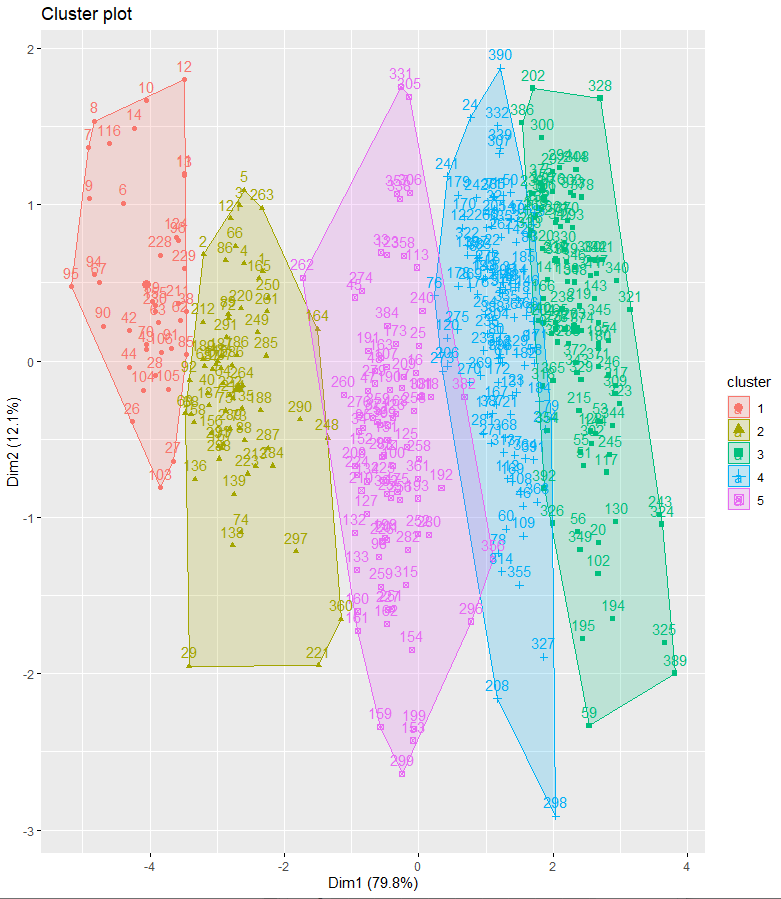
You then need to go back and investigate the cars in each cluster to draw some conclusions. To do this, find the cars in each cluster and examine their characteristics.

Try this w/ scale (aka zscore). Then, you should get both positive and negative numbers. Positive numbers are above the mean while negative numbers are below the mean. Can you see any patterns when you highlight the maximum values in each column?

With k=5:



So, only a slight improvement of 2.1 in between\_ss/total\_ss.



We see that clusters 3,4, and 5 stay the same.

And, so we could try higher k, but it seems unlikely we will get much better separation in the clusters.

With k=6 (not shown). No improvement in between\_ss/total\_ss.

…………..

In kmeans, the algorithm stops when:

1. Maximum number of iterations reached; or
2. The change to “within cluster sum-of-squares” in two successive iterations is less than the threshold value.

If the assignments of data points to clusters does not change during an iteration, then the algorithm has converged to at least a local minimum.

The within-cluster sum of squares is a measure of the variability of the observations within each cluster.

In general, a cluster that has a small sum of squares is more compact than a cluster that has a large sum of squares.

Clusters that have higher values exhibit greater variability of the observations within the cluster.

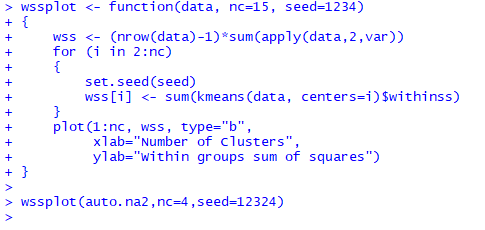
But, as the number of observations increases, the sum of squares becomes larger.

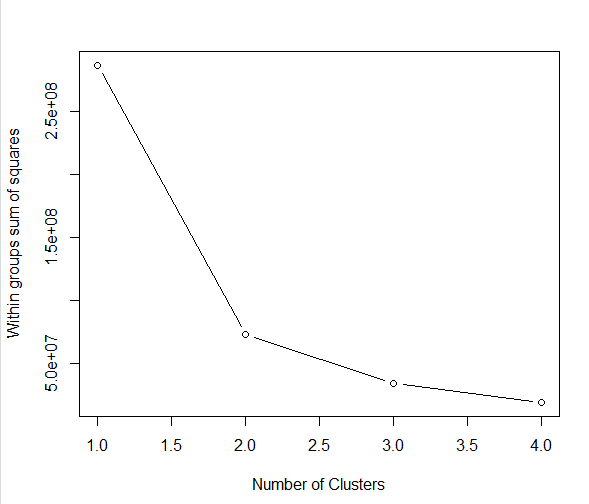
Whenever an assignment is changed, e.g., a data point moves from one cluster to another, the sum squared distances of the data points in the cluster is reduced.

The ratio (between\_ss/total\_ss) is a measure of the variance in the data set.

k-means minimizes the within group dispersion and maximizes the between-group dispersion.

Finding a Good Number of Clusters:

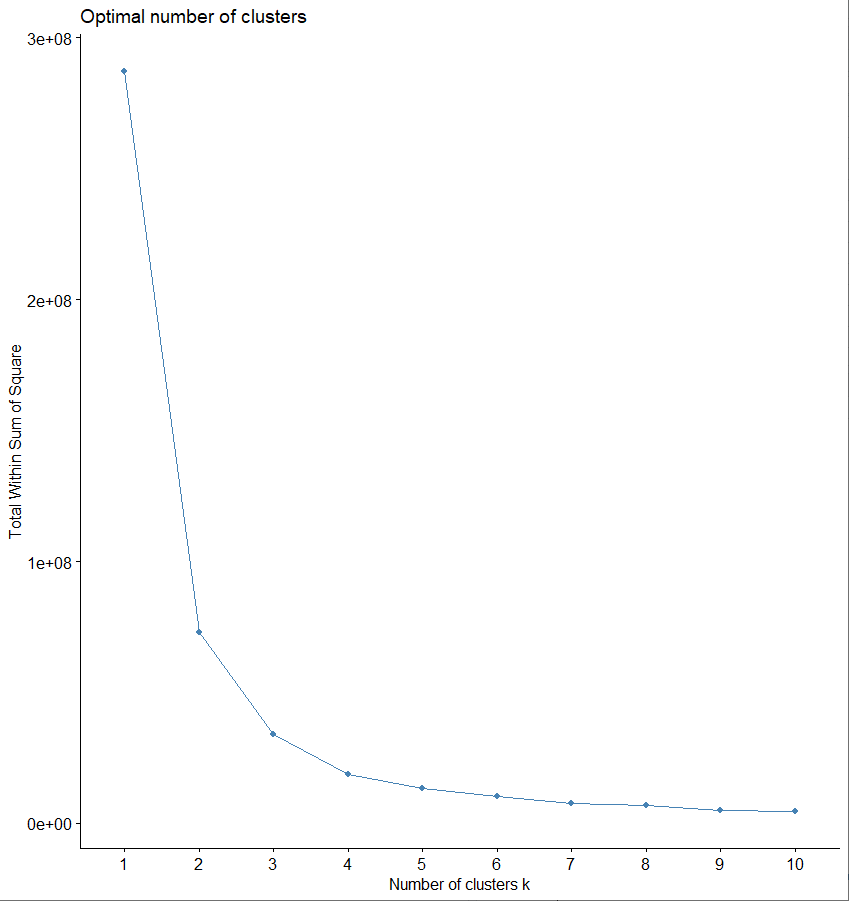




fviz\_nbclust(): Dertemines and visualize the optimal number of clusters using different methods: within cluster sums of squares, average silhouette and gap statistics.

Use the fviz\_nbclust() method in factoextra to get an estimate of the optimal number of clusters.





So, fviz\_nbclust suggests that the optimal number of clusters is around 4 or 5.

Remember that you can choose different algorithms for kmeans.

You do not have to choose different algorithms for the project, but you might consider doing it on your own just to what effect different distance metrics have on the clustering.

**K-NN**:

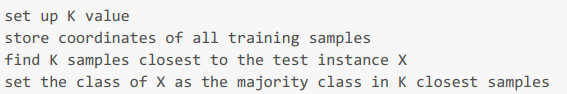
K-NN is a classification method, where k is the number of samples closest to the test instance.

Compute **k-nearest neighbour** classifications for test set data points based on a model constructed from the data points of the training set.

It is also used as a supervised machine learning algorithm, e.g., one which has a labeled data set.

For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the kth nearest vector, all candidates are included in the vote.

Here is the basic algorithm:



Note: There is no actual training process.

Larger k may lead to better performance/accuracy, but too large a k may end up looking samples that are not neighbors.

There are several knn algorithms in the CRAN Repository. This description is from class::knn.

knn(train, test, k = 1, prob = FALSE, use.all = TRUE)

where:

train matrix or data frame of training set cases.

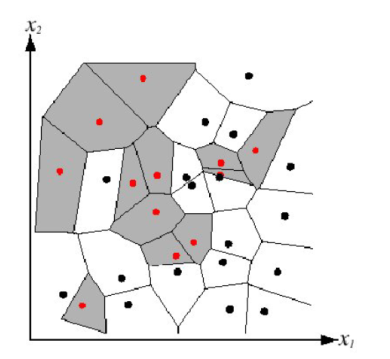
test matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case.

k number of neighbours considered.

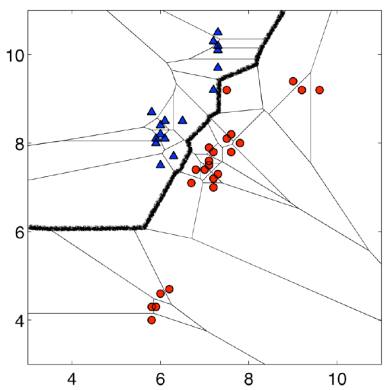
prob If this is true, the proportion of the votes for the winning class are returned as attribute prob.

use.all controls handling of ties. If true, all distances equal to the kth largest are included. If false, a random selection of distances equal to the kth is chosen to use exactly k neighbours.

Knn does not compute decision boundaries, e.g., how the data points are separated in the data space, but these can be inferred. One way is to use Voronoi diagrams, where the line segments represent the boundaries between classes and are equidistant between two points of opposite classes. (YOU DO NOT HAVE TO DO THIS!)



Then, a decision boundary between different classes might look like this:



One thing to be careful of is whether attributes are irrelevant to the problem to be solved. This requires domain knowledge, since such attributes may be considered noise. Thus, you should consider removing them from the data set under consideration, but save them off to the side.

Choosing the right value of K:

Run the knn algorithm several times. Choose the k that reduces the numbers of errors. But:

As we decrease the value of k to 1, our predictions become less stable.

As we increase the value of k, our predictions become more stable and accurate up to a point, but the number of errors is also likely to increase.

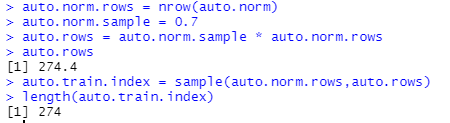
Where majority voting is used, prefer an odd k in order to have a tie breaker.

Creating training and test sets:

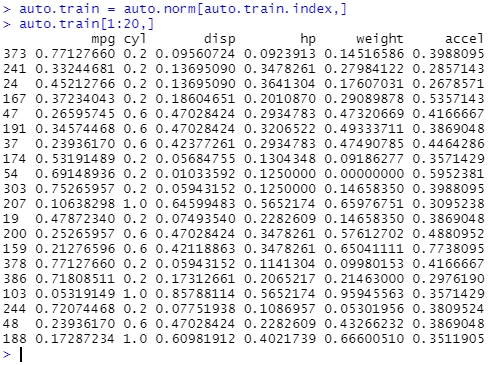
To create a training set, we sample the data set for a percentage of the rows using random selection.

In many problems, at least initially, the training set is larger than the test set.

In this exercise, we set the sample size to 70%.

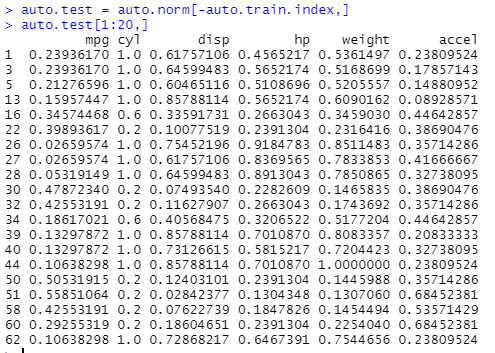


We get the actual training records by using the index to select the records from the data set:



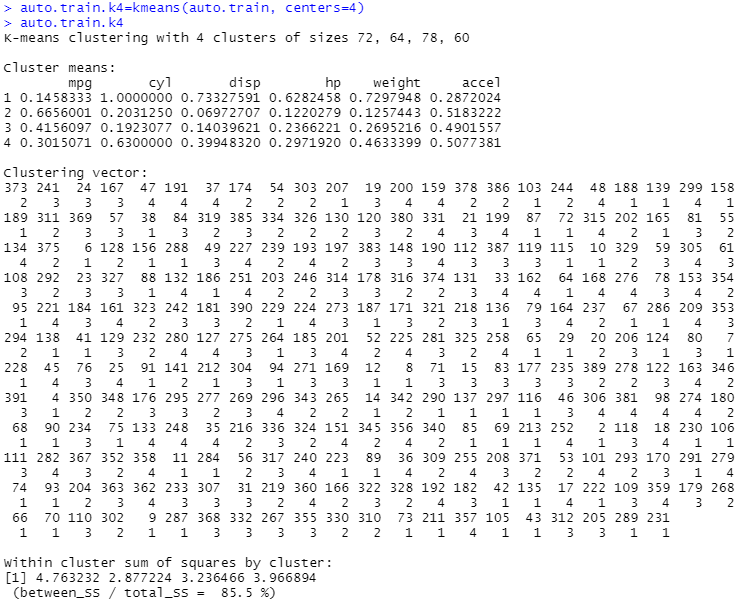
Note that records are not in order because sample uses a random number generator to select the records from the data set.

Now, we can construct the test data set as follows:



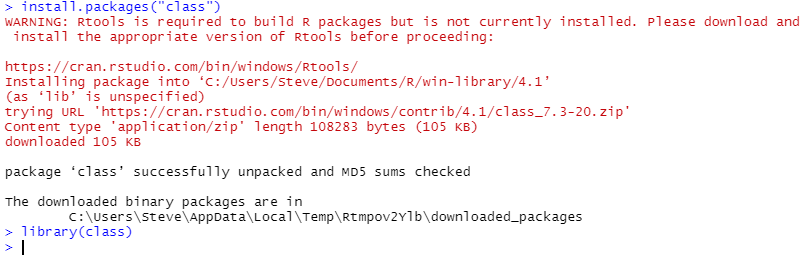
The only labels we have for the data points are the cluster ids after performing kmeans clustering.

So, to create training labels for kNN , we will do kmeans clustering on the training data set using k=4.

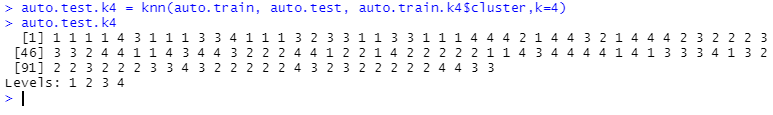


The clustering vector represents the labels for each of the data records in auto.train.

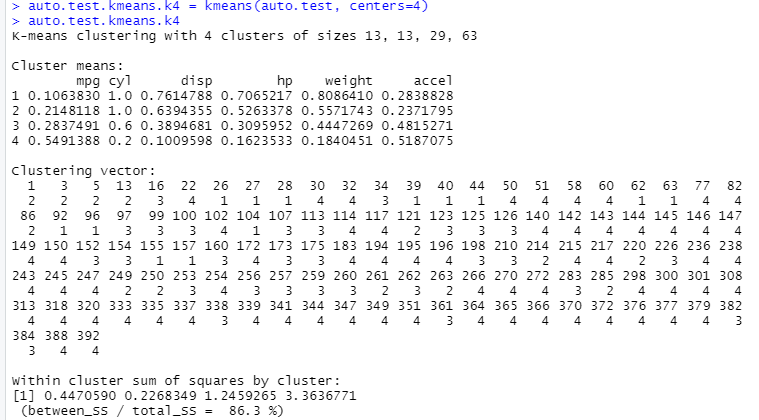
We need to load class which contains knn:



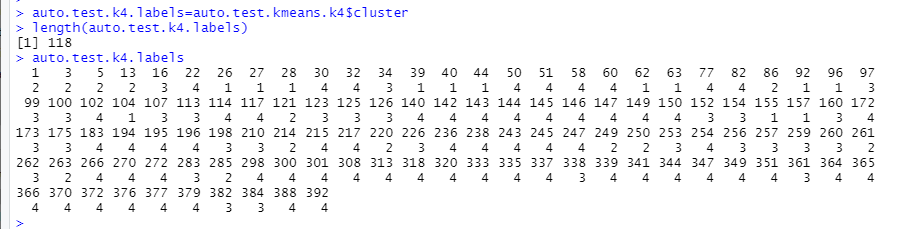
kNN returns a feature vector with the predicted values for each of the test values.



Now, apply kmeans to auto.test to generate labels for the test records



Now, we need to get the labels for the test records from applying kmeans to auto.test:



Compare the values of auto.test.k4.labels from kmeans to auto.test.k4 from kNN.

Why do they differ in terms of the clusters assigned to each car?

There are at least two answers.

1. They use different distance metrics (Hartigan-Wong vs. Euclidean). Choosing different distance metrics will give different values. Cluster number is NOT invariant with distance metric.

2. As noted in lecture, there may be some relationships between the attributes. Thus, the independence of different attributes is not guaranteed and the distribution of values may not be normal.

This latter point means that we require considerable domain knowledge to interpret the results.

Now, we want to use the kNN results to predict what the labels for the test set records should be.,

**Linear Modeling**:

Linear modeling tries to fit a line to the data that we have.

Remember that we said we make the assumption of a linear relationship because it makes the problem analysis tractable.

So, let’s use regression to see if we can detect where the dependencies might lie.

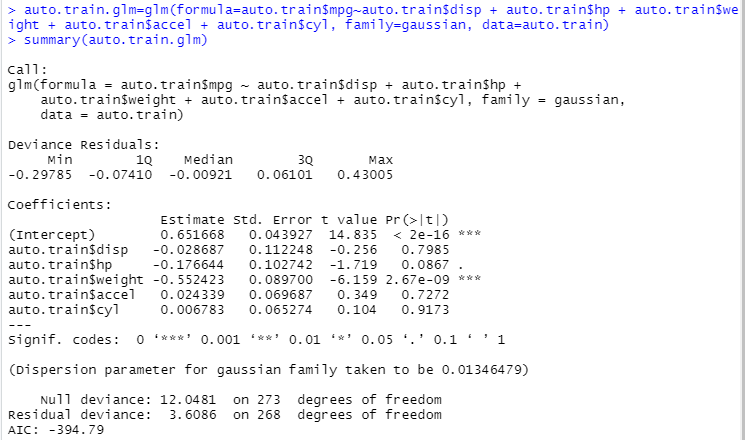
Clearly, mpg is dependent on cyl, disp, hp, weight, and accel for most cars then (when the data were collected).

We will use the glm() function in the stats package which is automatically loaded by R when you initiate RStudio.

Models for glm are specified symbolically. A typical model has the form response ~ terms where response is the (numeric) response vector and terms is a series of terms which specifies a linear predictor for response.

Here’s the auto.train model based on the training data set:





Note the squiggle (~) after auto.norm$mpg.

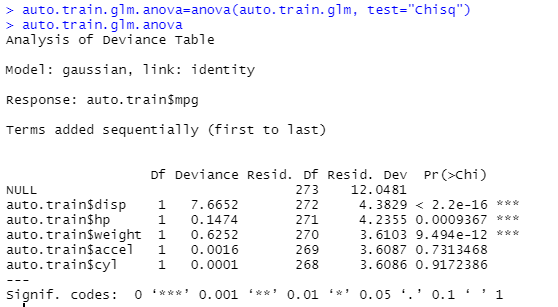
The Estimate gives an estimate of the regression coefficients.

The Std. Error gives the standard error for those estimates.

The t value = coefficient/Std. Error

The Pr(>|t|) converts the t value into a probability of the coefficient being 0.

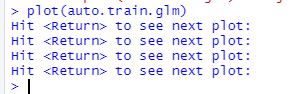
Let’s run the anova – Analysis of Variance on the model result.

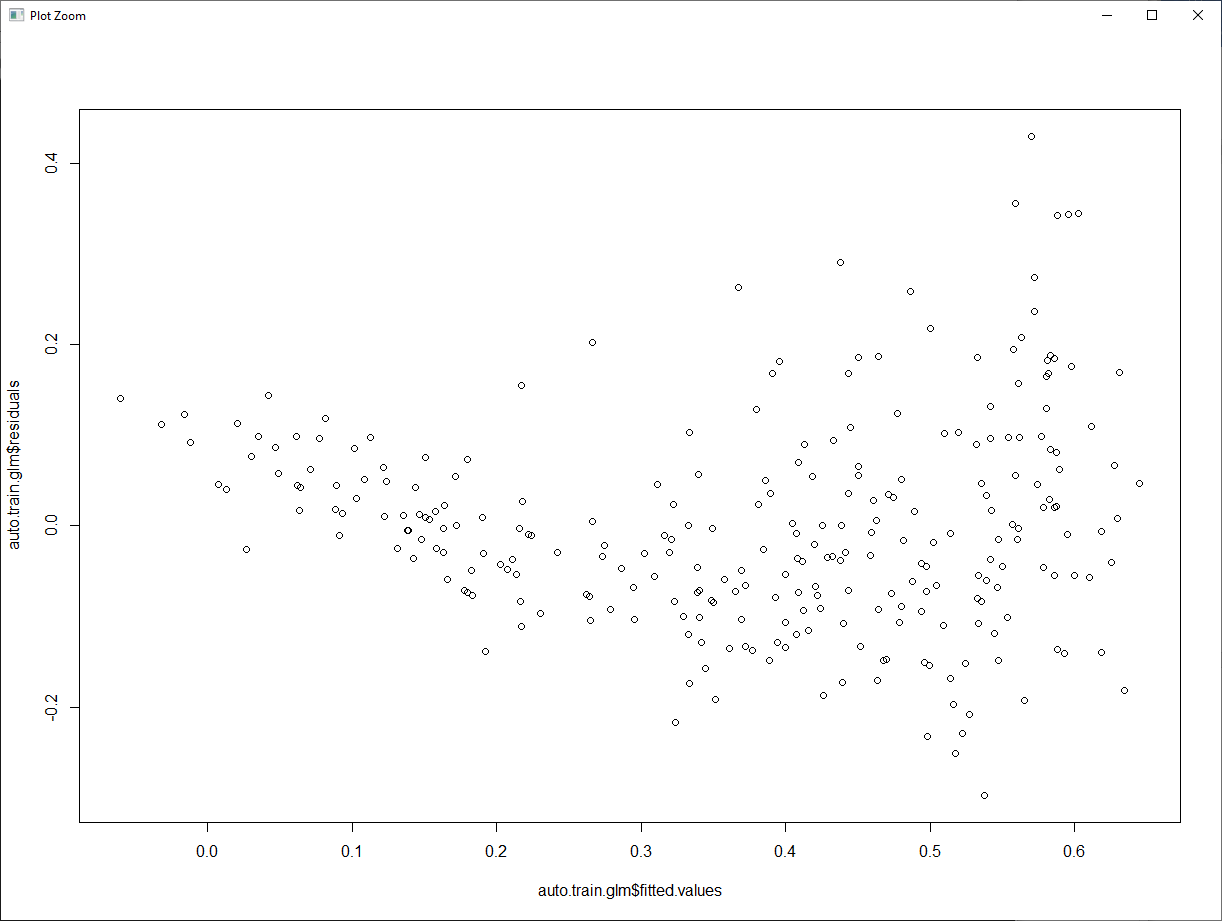


The difference between the null deviance and the residual deviance shows how our model is doing against the null model (a model with only the intercept).

The wider this gap, the better. Analyzing the table we can see the drop in deviance when adding each variable one at a time.

Now, lets plot the residuals versus the fitted.values





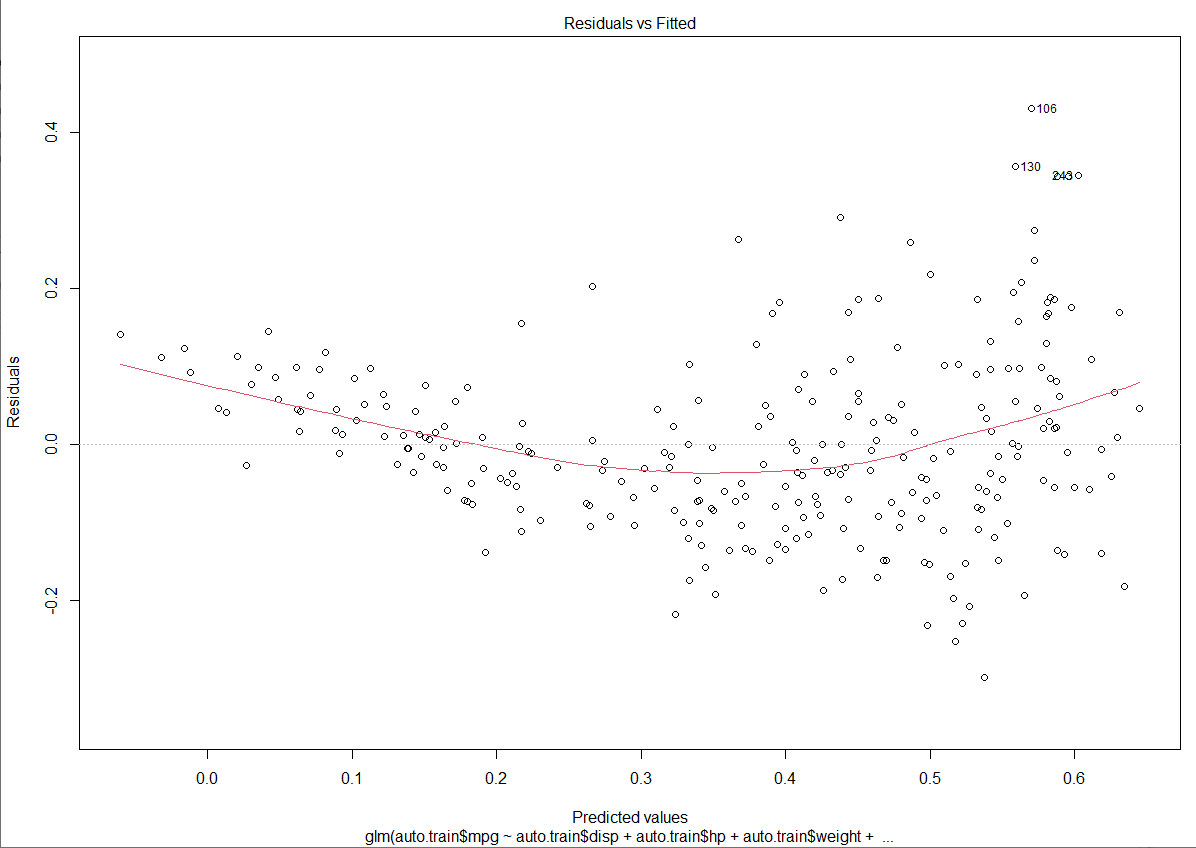
If residuals have a non-linear pattern, there could be a non-linear relationship between predictor variables and an outcome variable. The residuals are not equally distributed around the plot.

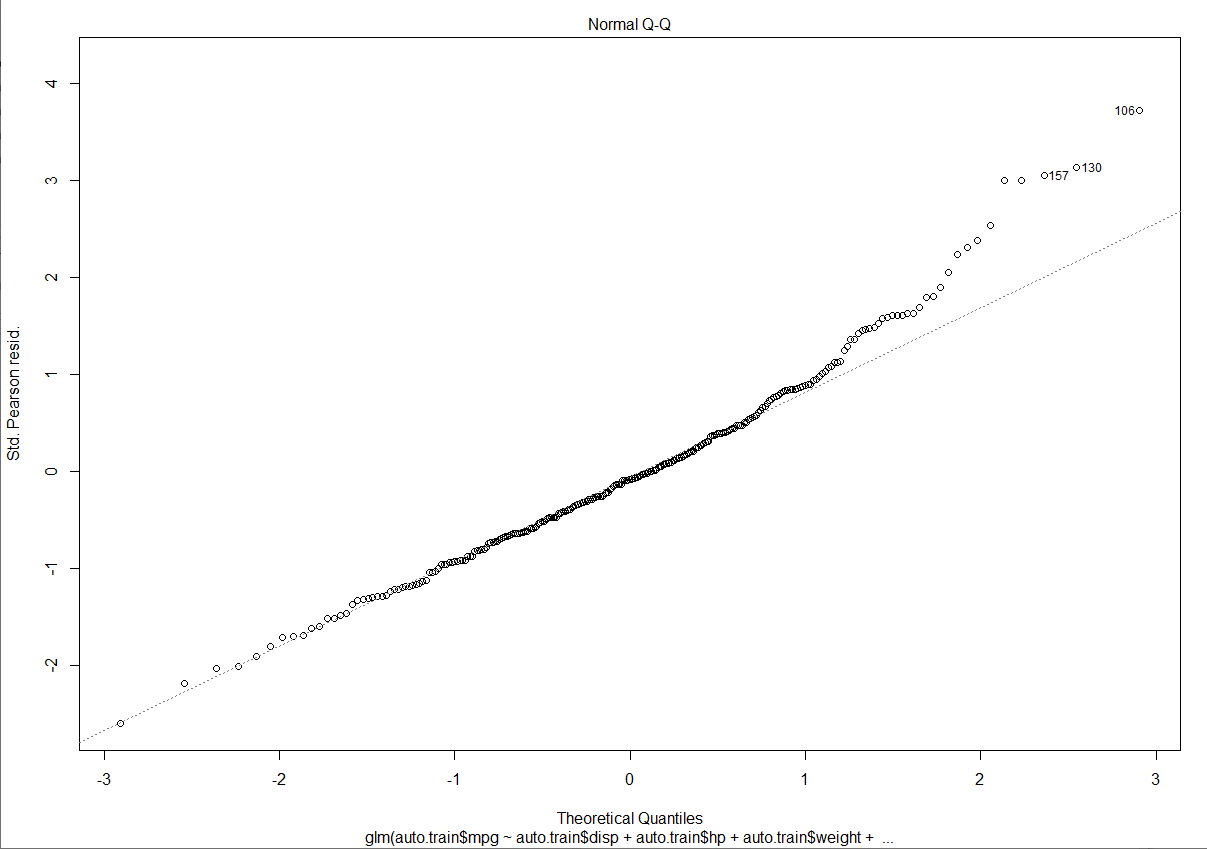
What do we see?

a. There are some outliers that we might want to remove that seem to be dragging the curve upward.

b. Since a curve is fitted, tells us there are some dependencies.

Source: <https://data.library.virginia.edu/diagnostic-plots/>



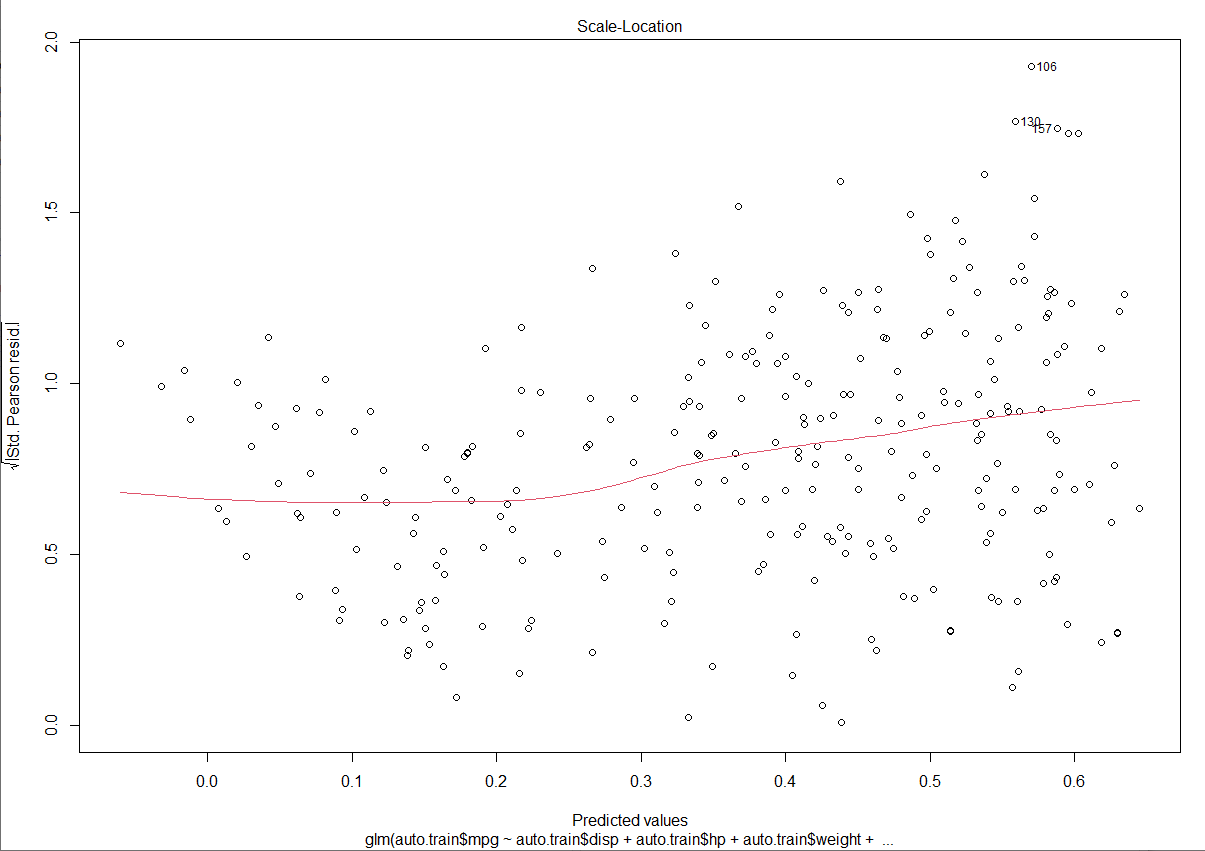


In statistics, a **Q–Q (quantile-quantile) plot** is a probability plot, which is a graphical method for comparing two probability distributions by plotting their quantiles against each other.

If the distributions are linearly related, the points in the Q–Q plot will approximately lie on a line, but not necessarily on the line *y* = *x*.

This plot indicates the data are almost normally distributed, except for the high end. Points 321,382,389, are problematics. We should go back and review their values and maybe remove them from the data set.

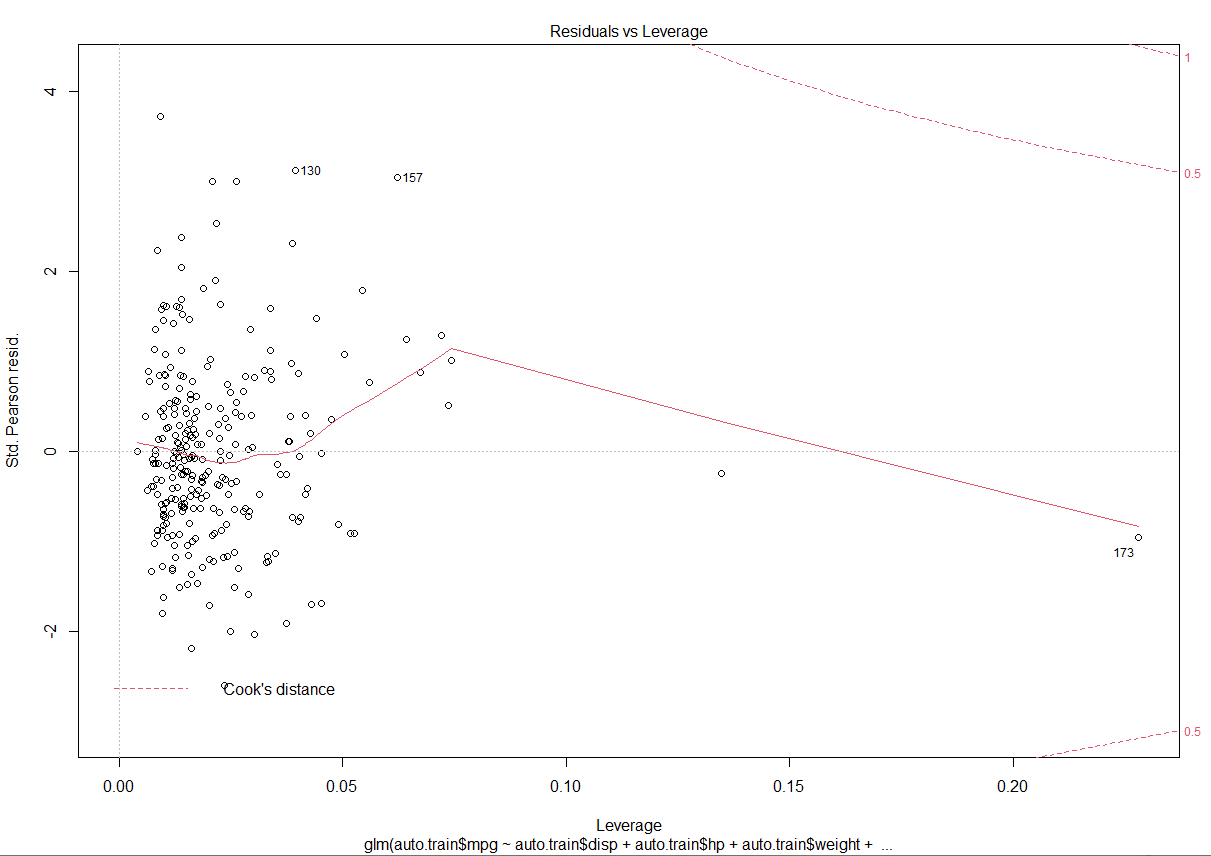
The deviation from the line at the high end is due to the outliers.



Also called a **Spread-Location plot**.

This plot shows if residuals are spread equally along the ranges of predictors. This is how you can check the assumption of equal variance (homoscedasticity). It’s good if you see a horizontal line with equally (randomly) spread points.

Difficult to visualize, but it seems more points are plotted above the line as we move to the right.



This plot helps us to find influential cases (i.e., subjects) if any. Not all outliers are influential in linear regression analysis (whatever outliers mean). Even though data have extreme values, they might not be influential to determine a regression line. That means, the results wouldn’t be much different if we either include or exclude them from analysis. They follow the trend in the majority of cases and they don’t really matter; they are not influential. On the other hand, some cases could be very influential even if they look to be within a reasonable range of the values. They could be extreme cases against a regression line and can alter the results if we exclude them from analysis. Another way to put it is that they don’t get along with the trend in the majority of the cases.

We look in the upper right or lower right for outlying values.

When points are outside of Cook’s distance, they are influential in affecting the regression.

We don’t have that case here. But, point 14 is problematic as it is an outlier.

What does having patterns in residuals mean to your investigation?

It tells you about your model and data. Your current model might not be the best way to understand your data if there’s so much good stuff left in the data.

You may want to include a quadratic term in your model or you may have left out an important variable or you may want to remove outliers or problematic data.

**Predict**

Predict obtains predictions and optionally estimates standard errors of those predictions from a fitted generalized linear model object.

predict(object, newdata = NULL,

type = c("link", "response", "terms"),

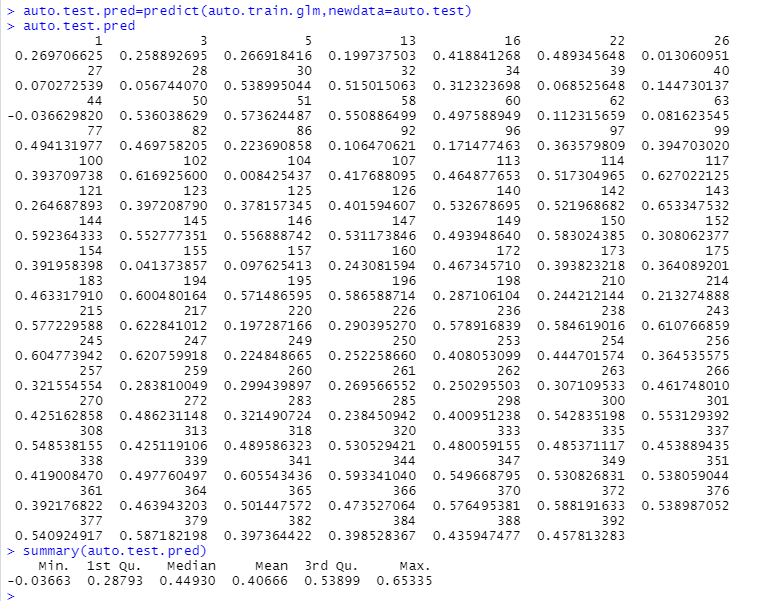
se.fit = FALSE,

dispersion = NULL,

terms = NULL,

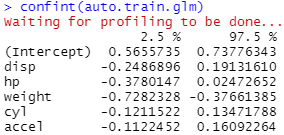
na.action = na.pass, ...)

Predict returns a vector of predicted values for the dependent variable (e.g., mpg) based on the training set.



**Confidence Intervals**:

A **confidence interval** (**CI**) is a range of estimates for an unknown [parameter](https://en.wikipedia.org/wiki/Statistical_parameter), defined as an interval with a lower bound and an upper bound. The 95% confidence level is most common. The factors affecting the width of the CI include the confidence level, the [sample size](https://en.wikipedia.org/wiki/Sample_size_determination), and the [variability](https://en.wikipedia.org/wiki/Statistical_dispersion) in the sample



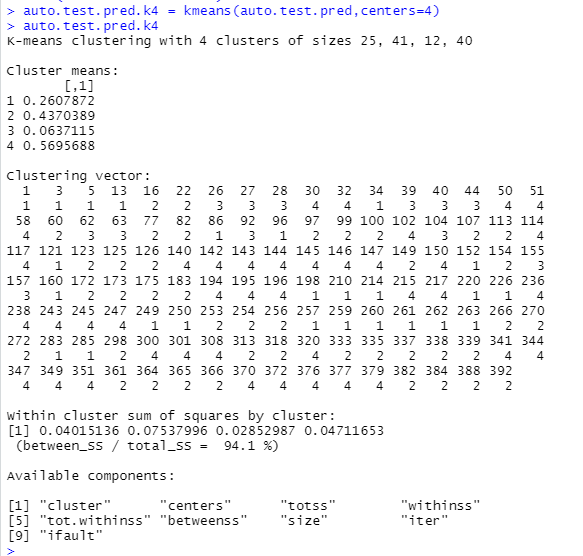
We want to ficus on the variability of the Intercept since that is the range of the dependent variable.

Of course, you need to rescale it to get back to the values associated with the data set.

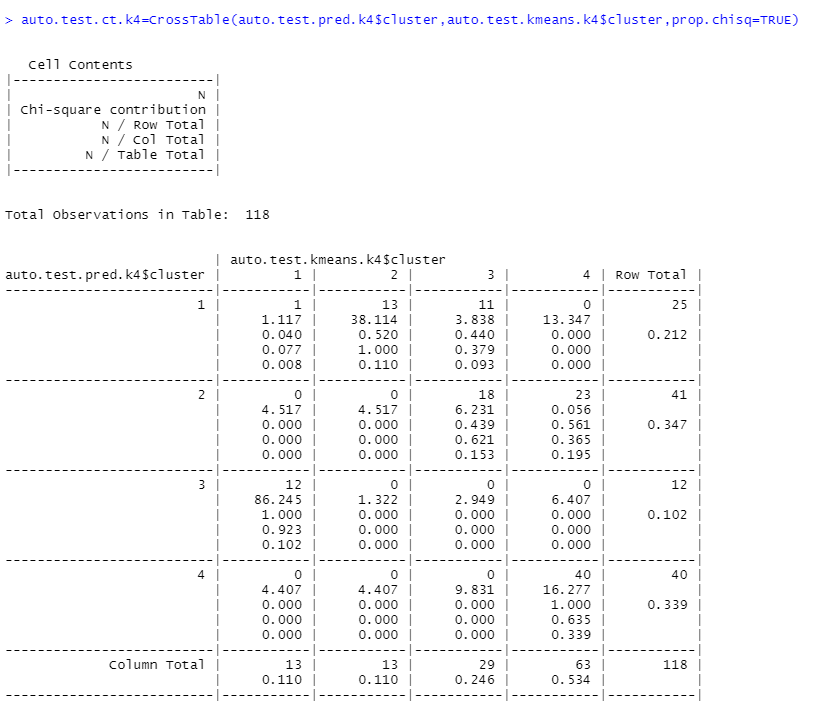
**Comparing Actual vs. Prediction**:

So, we have auto.test.kmeans.k4 for the original data.

We need to get the kmeans data for the predicted values.



Now, we will compare the results from the two clusterings using CrossTable.in gmodels package.



Well, looks like our prediction isn’t very good.

There could be several reasons for this :

1. There are interdependencies among the five variables used to predict mpg ;
2. We may need to remove one or more variables, which do not really contribute and may be perturbing the results.

So, compare the cluster counts :

Cluster auto.test.kmeans.k4 auto.test.pred.k4

1 1 1

2 13 41

3 29 12

4 63 40

Compute the performance measures:

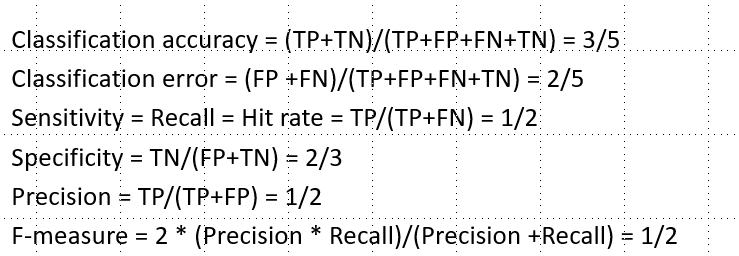
Performance Measures for Classification Models:

TP = True Positives

FP = False Positives

TN = True Negatives

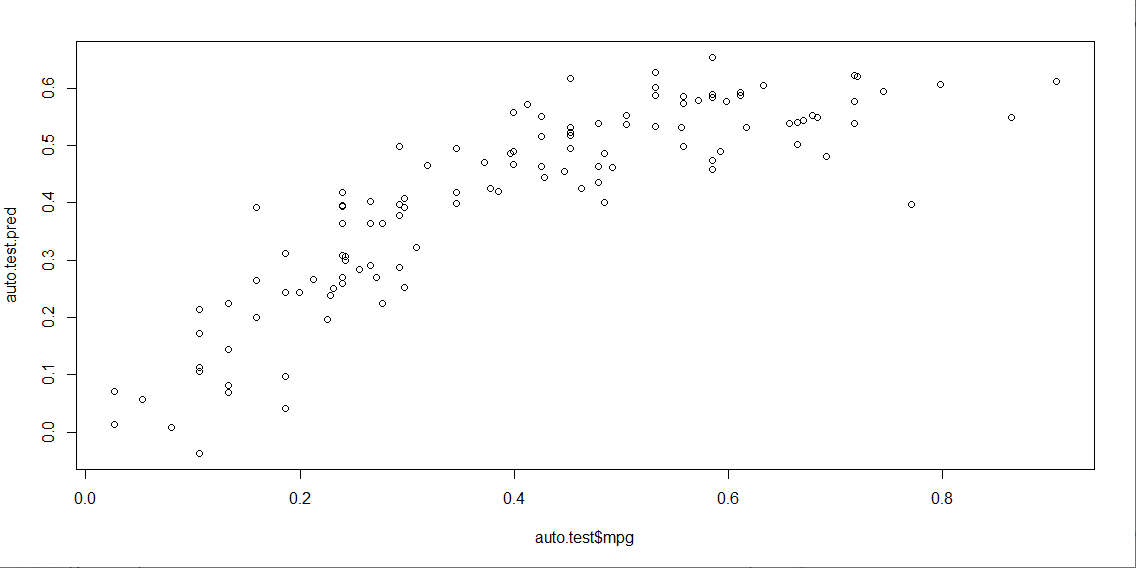
FN = False Negatives



So, 41 samples were predicted correctly and 69 were predicted incorrectly.

Since we only have TP =41 and FP = 77, we have precision = TP/(TP+FP) = 41/118 = 34%

Plotting auto.test.$mpg versus auto.test.pred provides the following plot :



We see here not quite a linear correspondance, but the need for a non-linear term, but probably not of the form x2.

You would need to identify what variable(s) have the greatest interdependency.

Remember : Data science is an ***experimental*** science. There are many aspects of the methods we have shown that have optional arguments associated with them. We do not have space here to explore them, but reading the method documentation will give you an idea of what alternatives that you have.

NOTE: This document is subject to revision!!

Appendix : Different Formula Expressions

