# HW6 - DATA 609

Thomas Hill

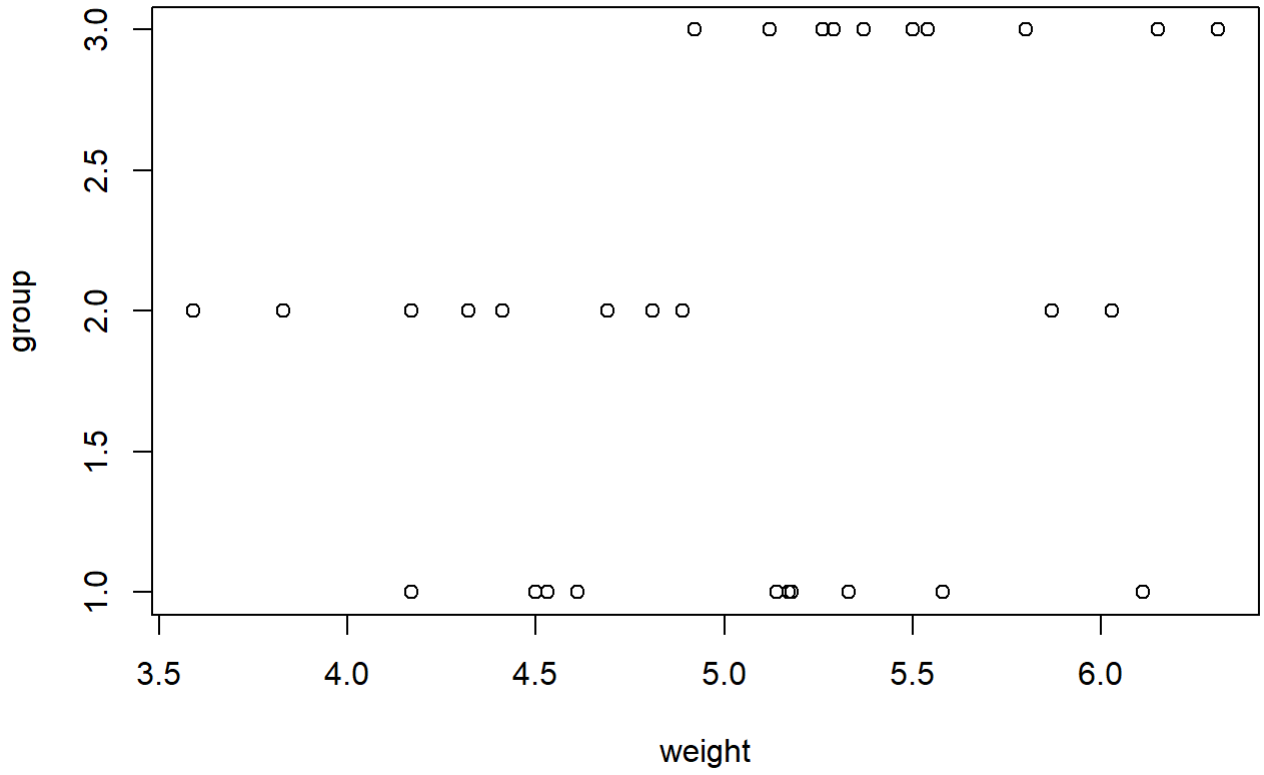
November 14, 2021

```
library(knitr)
library(class)
```

**Ex. 1** – Use a dataset such as the *PlantGrowth* in R to calculate three distance metrics and discuss the results.

```
pg <- PlantGrowth

plot(pg)
```




PlantGrowth is a dataframe comparing the yields of plants that had one of two treatments, or no treatment at all. Because these treatments only consider weight and another another dimension like treatment dose, I'll start by looking at the distance between points ranked by size. Because of this, the Euclidean and Manhattan distances are the same.

```
sum(abs(sort(pg[pg$group == 'trt1',]['weight']]) - sort(pg[pg$group == 'ctrl',]['weight']])))
```

```
## [1] 4.29
```

```
sum(abs(sort(pg[pg$group == 'trt2',][['weight']]) - sort(pg[pg$group == 'ctrl',][['weight']])))
```

```
## [1] 4.94
```

Using this metric, treatment 1 appears slightly closer to the control.

Next, lets consider the distance of each treatment to its nearest control neighbor. I did this by subtracting each control value from a single treatment, found the absolute value of each difference, and located the minimum value of the vector. I did this for all 10 samples and summed the total distances as calculated by Euclidean distance.

```
pg_ctrl <- pg[pg$group == 'ctrl',][['weight']]
pg_t1 <- pg[pg$group == 'trt1',][['weight']]
pg_t2 <- pg[pg$group == 'trt2',][['weight']]


sum(min(abs(pg_t1[1] - pg_ctrl)) + min(abs(pg_t1[2] - pg_ctrl)) + min(abs(pg_t1[3] - pg_ctrl)) +
min(abs(pg_t1[4] - pg_ctrl)) + min(abs(pg_t1[5] - pg_ctrl)) + min(abs(pg_t1[6] - pg_ctrl)) + min
(abs(pg_t1[7] - pg_ctrl)) + min(abs(pg_t1[8] - pg_ctrl)) + min(abs(pg_t1[9] - pg_ctrl)) + min(ab
s(pg_t1[10] - pg_ctrl)))
```

```
## [1] 2.01
```

```
sum(min(abs(pg_t2[1] - pg_ctrl)) + min(abs(pg_t2[2] - pg_ctrl)) + min(abs(pg_t2[3] - pg_ctrl)) +
min(abs(pg_t2[4] - pg_ctrl)) + min(abs(pg_t2[5] - pg_ctrl)) + min(abs(pg_t2[6] - pg_ctrl)) + min
(abs(pg_t2[7] - pg_ctrl)) + min(abs(pg_t2[8] - pg_ctrl)) + min(abs(pg_t2[9] - pg_ctrl)) + min(ab
s(pg_t2[10] - pg_ctrl)))
```

```
## [1] 0.97
```

Using this metric, treatment 2 is closer to control.

Finally, I'll consider the Jaccard similarity index of each treatment group versus control. To apply this to one-dimensional values, I will consider the range of each treatment as $|U \cap V|$, while the range of control and treatment will be $|U \cup V|$. Subtracting this value from 1 will give the Jaccard distance.

```
1- (max(pg_t1) - min(pg_t1)) / (max(cbind(pg_ctrl, pg_t1)) - min(cbind(pg_ctrl, pg_t1)))
```

```
## [1] 0.03174603
```

```
1- (max(pg_t2) - min(pg_t2)) / (max(cbind(pg_ctrl, pg_t2)) - min(cbind(pg_ctrl, pg_t2)))
```

```
## [1] 0.3504673
```

The Jaccard distance for treatment 1 is closer to control.

**Ex. 2** – Now use a higher-dimensional set *mtcars*, try the same distance metrics in the previous question and discuss the results.

For a larger dimension dataset, I'll be using the built-in *dist* function in R to calculate the distance. While *mtcars* contains all numerical values for each variable, there is one dummy variable (transmission is 0 or 1), and several categorical variables (number of forward gears, carburetors, cylinders). I will consider the distance of each value

```
sum(dist(as.matrix(mtcars), method = 'euclidian'))
```

```
## [1] 83966.83
```

Without summing the values, the *dist* function would provide a matrix giving the distance of each car from each other car. I summed the resulting matrix for all of its values for easier comparison between distance metrics.

```
sum(dist(as.matrix(mtcars), method = 'manhattan'))
```

```
## [1] 116831.2
```

For Jaccard distance, I used the *distance* function as part of the 'ecodist' package.

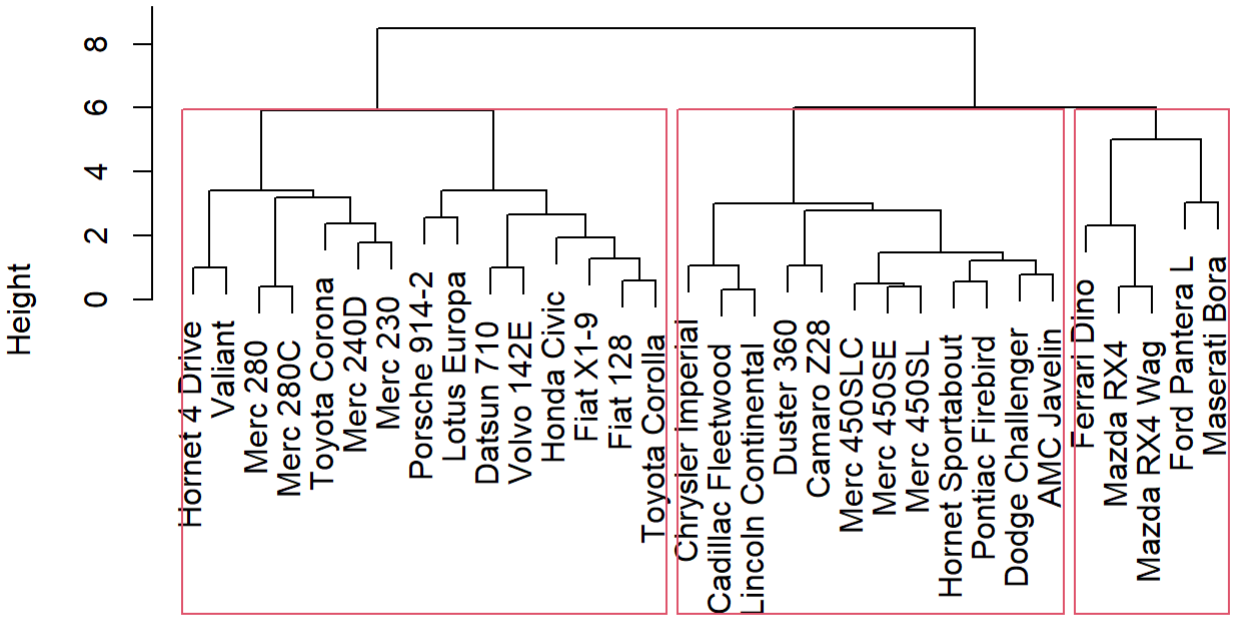
```
sum(ecodist :: distance(as.matrix(mtcars), method = 'jaccard'))
```

```
## [1] 46.78182
```

Compared to the 1-dimensional example, there appears to be much more variation between distance metrics. This comparison may underscore the importance of scaling initial values for effective use of distance metrics.

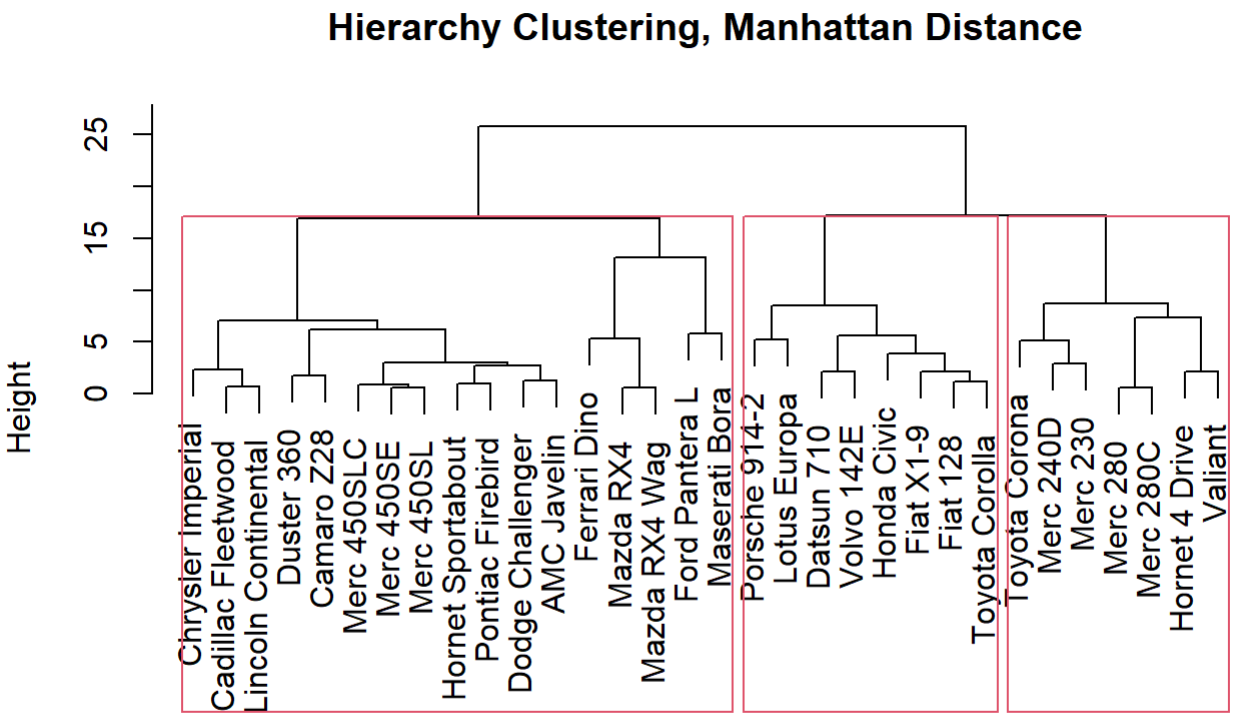
**Ex. 3** – Use the built-in dataset *mtcars* to carry out hierarchy clustering using two different distance metrics and compare if they get the same results. Discuss the results.

```
plot(hclust(dist(scale(mtcars), method = 'euclidian')), main = 'Hierarchy Clustering, Euclidian
 Distance')
rect.hclust(hclust(dist(scale(mtcars), method = 'euclidian')), k = 3)
```




dist(scale(mtcars), method = "euclidian")
hclust (*, "complete")

```
plot(hclust(dist(scale(mtcars), method = 'manhattan')), main = 'Hierarchy Clustering, Manhattan
 Distance')
rect.hclust(hclust(dist(scale(mtcars), method = 'manhattan')), k = 3)
```




After scaling data, the two distance metrics offered seemingly different clusters. I added calls to *rect* to better visualize the differences. It appears that some of the clusters have similarities but there are discernible differences between the distance metrics.

**Ex. 4** – Load the well-known Fisher's *iris* flower dataset that consists of 150 samples for three species (50 samples per species). The four measures or features are the lengths and widths of sepals and petals. Use the kNN clustering to analyze this iris dataset by selecting 120 samples for training and 3 samples for testing.

```
set.seed(1114)

train_index <- sample(seq_len(150), size =120) #generate random index separating iris into speci
fied samples
train <- iris[train_index, -5] #training data
test <- iris[-train_index, -5] #test data

train_cat <- iris[train_index, 5]
test_cat <- iris[-train_index, 5]  #true designations for test

iris_predict <- (knn(train = train, test = test, cl = train_cat, k = 3)) #find predicted categor
ies

table(iris_predict, test_cat) #contingency table
```

```
##             test_cat
## iris_predict setosa versicolor virginica
##   setosa          6          0         0
##   versicolor      0         13         0
##   virginica       0          1        10
```

Using kNN only missed a single categorization at k = 3.

**Ex. 5** – Use the *iris* dataset to carry out k-means clustering. Compare the results to the actual classes to estimate the clustering accuracy.

```
set.seed(1114)

iris_kmeans <-  kmeans(iris[,-5], centers = 3)$cluster #obtain cluster designations

round(table(iris$Species, iris_kmeans)*100/150,1) #create contingency table for cluster numbers
 versus species
```

```
##             iris_kmeans
##                 1    2    3
##   setosa      0.0 33.3  0.0
##   versicolor  1.3  0.0 32.0
##   virginica  24.0  0.0  9.3
```

After dividing by n = 150 to obtain a percentage, it appears that clustering accuracy for k-means is not as high as knn. The *setosa* species is present in clusters 2 and 3 at a 2:1 ratio, while *versicolor* and *virginica* are lumped together in cluster 1. Let's see if increasing number of centroids to 5 will improve accuracy

```
set.seed(1114)

iris_kmeans <-  kmeans(iris[,-5], centers = 5)$cluster #obtain cluster designations

round(table(iris$Species, iris_kmeans)*100/150,1) #create contingency table for cluster numbers
 versus species
```

```
##             iris_kmeans
##                 1    2    3    4    5
##   setosa      0.0 14.7  0.0 18.7  0.0
##   versicolor 15.3  0.0 18.0  0.0  0.0
##   virginica  11.3  0.0  0.7  0.0 21.3
```

While the result is a little difficult ot understand, it still appears there are accuracy problems with k-means compared to knn.