DATA 609 HW 6

Daniel Moscoe

11/4/2021

Ex. 1. Use a data set such as `PlantGrowth` to calculate three different distance metrics and discuss the results.

Response. `PlantGrowth` contains data on the dried weight of 30 plant samples. Each sample is a member of the `ctrl` group, the `trt1` group, or the `trt2` group. I'll build distance matrices for three different distance metrics and discuss the results.

```r
df <- PlantGrowth
distmatrix_euc <- dist(df$weight, method = "euclidean")
distmatrix_max <- dist(df$weight, method = "maximum")
distmatrix_man <- dist(df$weight, method = "manhattan")

print(length(distmatrix_euc))
```

```
## [1] 435
```

```r
print(sum(distmatrix_euc == distmatrix_man))
```

```
## [1] 435
```

```r
print(sum(distmatrix_man == distmatrix_max))
```

```
## [1] 435
```

Because the sum of equal entries in each pair of distance matrices equals the total number of entries in each lower-triangular distance matrix, the distance matrices are identical. This is a consequence of the 1-dimensionality of the data in `PlantGrowth`: all these distance metrics are equivalent in their computation of distance along a single dimension.

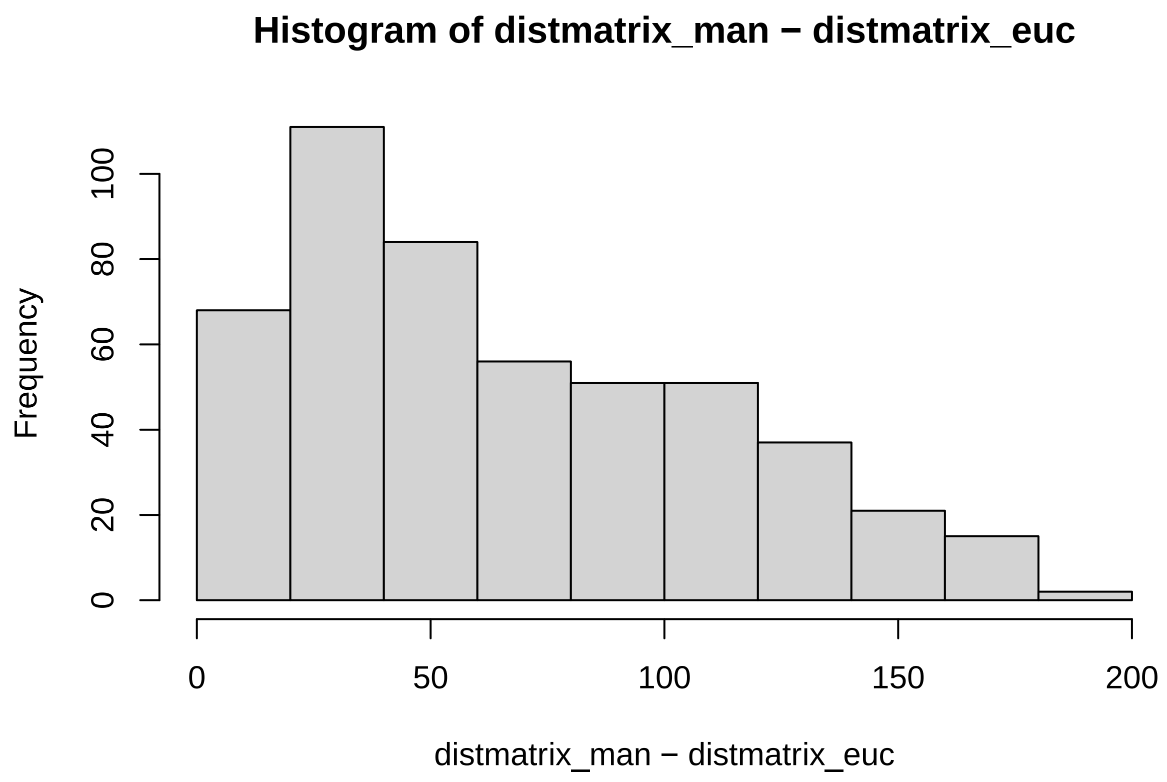
Ex. 2. Now use a higher dimensional data set, `mtcars`, try the same three distance metrics in the previous question and discuss the results.

Response.

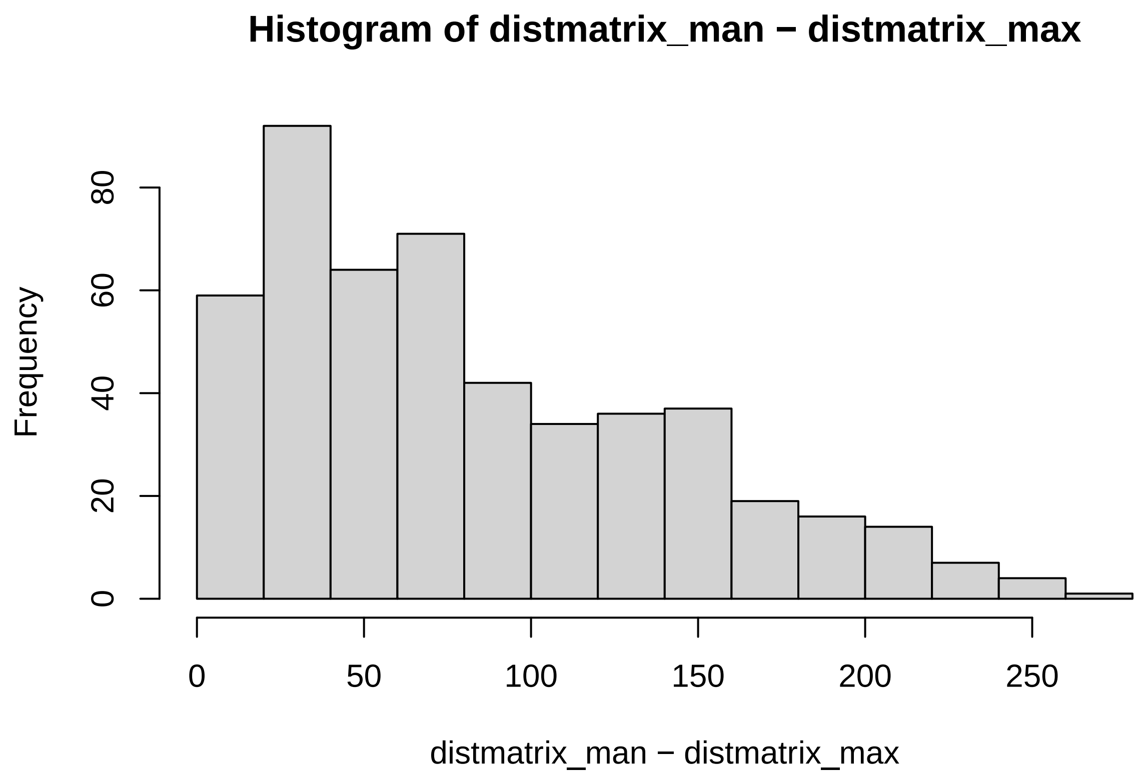
```r
df <- mtcars
distmatrix_euc <- dist(df, method = "euclidean")
distmatrix_max <- dist(df, method = "maximum")
distmatrix_man <- dist(df, method = "manhattan")
```

The histograms below show that the Manhattan distance between two points is always at least as great as the Euclidean distance and the maximum distance, where maximum distance is defined as the maximum distance between corresponding components of two observations. The Euclidean distance is always at least as great as the maximum distance.

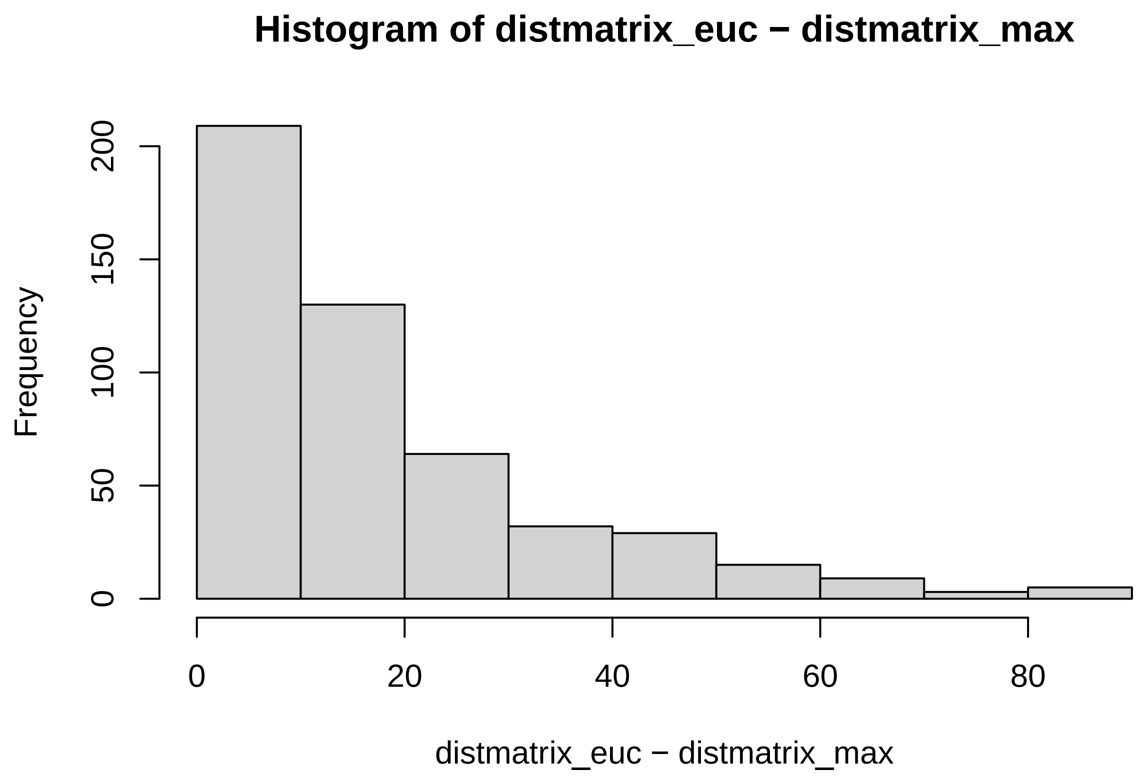
```
hist(distmatrix_man - distmatrix_euc)
```



```
hist(distmatrix_man - distmatrix_max)
```



```
hist(distmatrix_euc - distmatrix_max)
```



The histograms also show that the mean difference between Euclidean and maximum distance is much less than mean differences between either other pair of metrics.

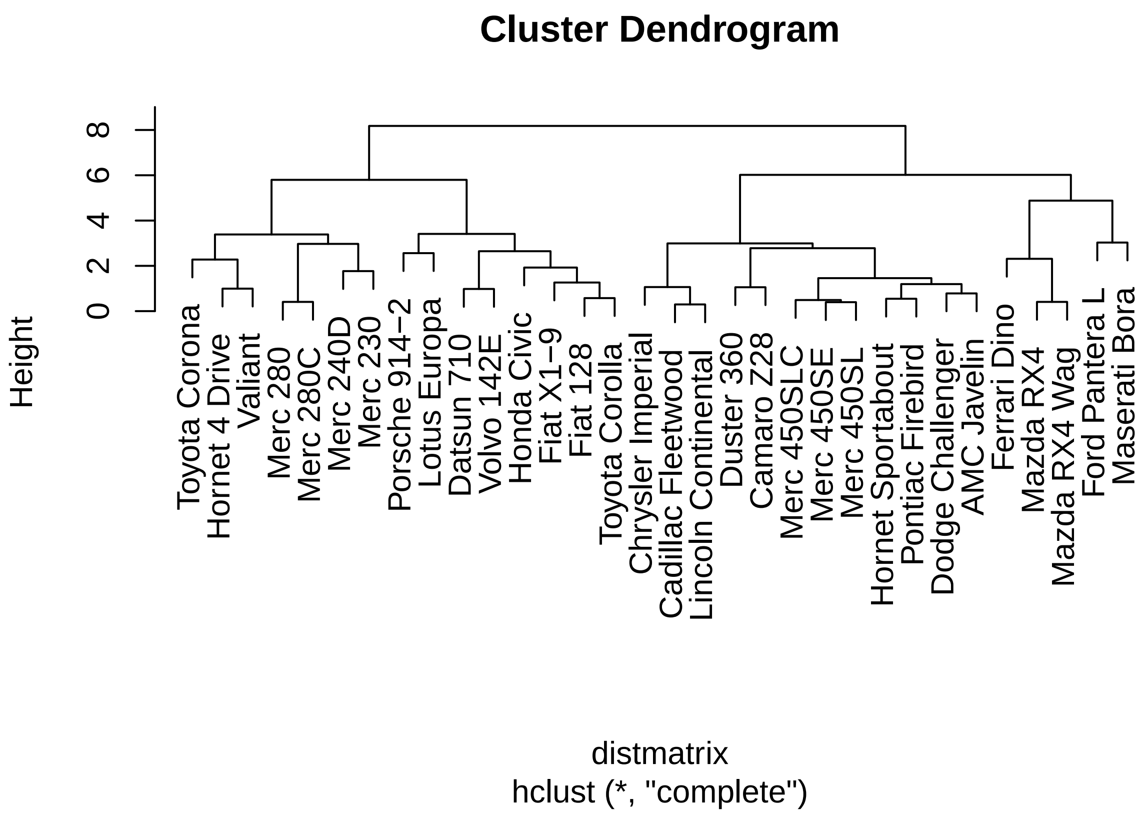
Ex. 3. Use the built-in data set `mtcars` to carry out hierarchy clustering using two different distance metrics and compare if they get the same results. Discuss the results.

Response.

```
cars <- mtcars

#Remove response variable
cars.cyl <- cars$cyl
cars$cyl <- NULL

#Scale data
cars_sc <- scale(cars)

#Construct Euclidean distance matrix
distmatrix <- dist(cars_sc, method = "euclidean")
?hclust
```

```
## starting httpd help server ... done
```

```
clusters <- hclust(distmatrix)
plot(clusters)
```



Since there are only three different values for `cyl`, cut the tree with `k = 3`:

```
cut_tree <- cutree(clusters, k = 3)
cars_cl <- cars %>%
  mutate(cluster = cut_tree)
table(cars_cl$cluster, cars.cyl)
```

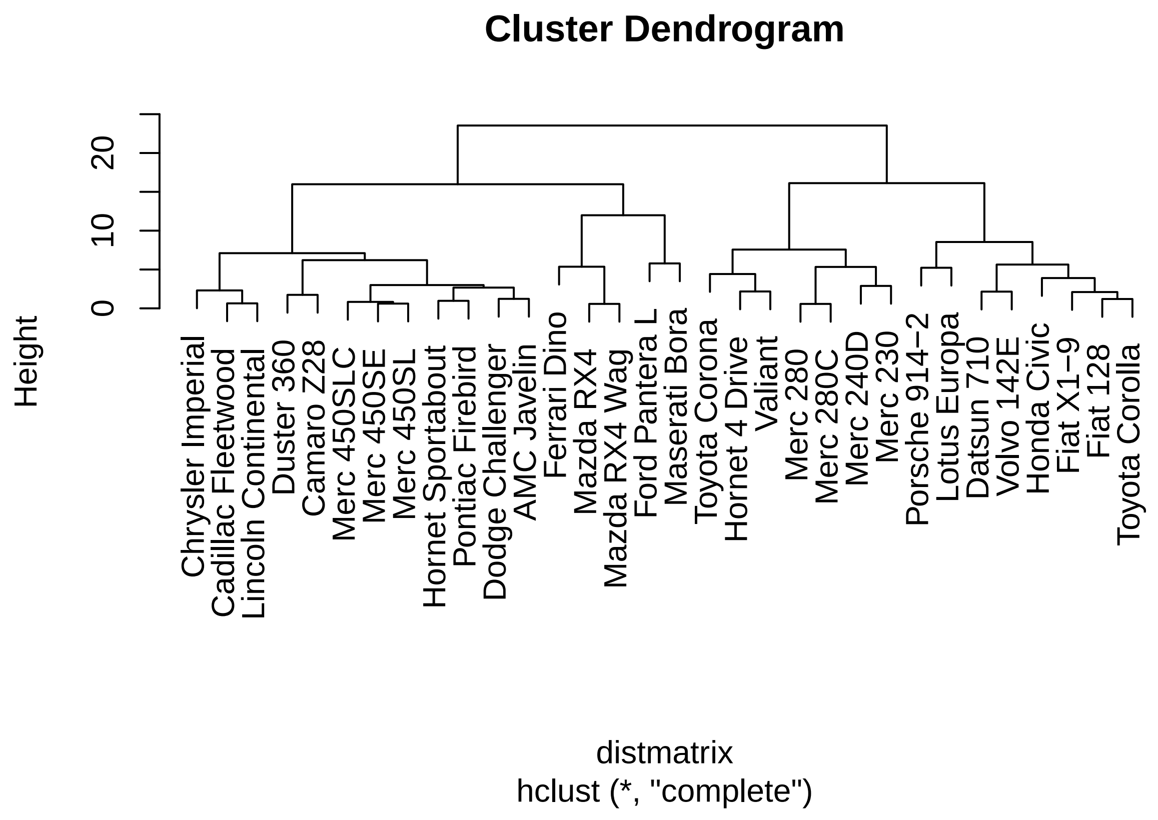
```
##    cars.cyl
##      4  6  8
##   1  0  3  2
##   2 11  4  0
##   3  0  0 12
```

Overall accuracy with the Euclidean metric is about 81%, assuming that cluster 2 represents 4-cylinder cars, cluster 1 represents 6-cylinder cars, and cluster 3 represents 8-cylinder cars.

Using a Manhattan distance metric:

```
cars <- mtcars

#Remove response variable
cars.cyl <- cars$cyl
cars$cyl <- NULL

#Scale data
cars_sc <- scale(cars)
```

```r
#Construct Euclidean distance matrix
distmatrix <- dist(cars_sc, method = "manhattan")

clusters <- hclust(distmatrix)
plot(clusters)
```



```r
cut_tree <- cutree(clusters, k = 3)
cars_cl_man <- cars %>%
  mutate(cluster = cut_tree)
table(cars_cl_man$cluster, cars.cyl)
```

```
##    cars.cyl
##      4  6  8
##   1  0  3 14
##   2  8  0  0
##   3  3  4  0
```

Using Manhattan distance, overall accuracy is unchanged. Cluster 1 represents 8-cylinder cars, cluster 2 represents 4-cylinder cars, and cluster 3 represents 6-cylinder cars. Comparing the classifications from each model:

```r
res <- data.frame(
  "name" = rownames(mtcars),
  "actual" = mtcars$cyl,
  "euc" = cars_cl$cluster,
  "man" = cars_cl_man$cluster)

res$euc[res$euc == 1] <- 6
res$euc[res$euc == 2] <- 4
res$euc[res$euc == 3] <- 8

res$man[res$man == 1] <- 8
res$man[res$man == 2] <- 4
res$man[res$man == 3] <- 6
head(res)
```

```
##                                  name actual euc man
## Mazda RX4                   Mazda RX4      6   6   8
## Mazda RX4 Wag           Mazda RX4 Wag      6   6   8
## Datsun 710                 Datsun 710      4   4   4
## Hornet 4 Drive         Hornet 4 Drive      6   4   6
## Hornet Sportabout Hornet Sportabout      8   8   8
## Valiant                       Valiant      6   4   6
```

Looking more closely at cars that were misclassified by at least one model:

```r
misclass <- res %>%
  filter(actual != euc | euc != man)
head(misclass)
```

```
##                            name actual euc man
## Mazda RX4             Mazda RX4      6   6   8
## Mazda RX4 Wag     Mazda RX4 Wag      6   6   8
## Hornet 4 Drive Hornet 4 Drive      6   4   6
## Valiant                 Valiant      6   4   6
## Merc 240D             Merc 240D      4   4   6
## Merc 230               Merc 230      4   4   6
```

```r
print(sum(misclass$actual == 4)/sum(mtcars$cyl == 4))
```

```
## [1] 0.2727273
```

```r
print(sum(misclass$actual == 6)/sum(mtcars$cyl == 6))
```

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

```r
print(sum(misclass$actual == 8)/sum(mtcars$cyl == 8))
```

```
## [1] 0.1428571
```

About a quarter of 4-cylinder cars were misclassified by at least one model. All of the 6-cylinder cars were misclassified by at least 1 model. And 14% of the 8-cylinder cars were misclassified at least once.

Ex. 4. Load the well-known Fisher's `iris` data set that consists of 150 samples for three species (50 samples for each species).... Use kNN clustering to analyze this iris data set by selecting 120 samples for training and 30 samples for testing.

Response.

```
set.seed(346)
df <- iris
training_rows <- sample(nrow(df), 120, replace = FALSE)
df.train.X <- df[training_rows, c(1:4)]
df.train.y <- df[training_rows, 5]
df.test.X <- df[-training_rows, c(1:4)]
df.test.y <- df[-training_rows, 5]
knnmod <- class::knn(train = df.train.X, test = df.test.X, cl = df.train.y, k = 5)

cm <- table(knnmod, df.test.y)
cm
```

```
##             df.test.y
## knnmod        setosa versicolor virginica
##   setosa           9          0         0
##   versicolor       0          9         0
##   virginica        0          1        11
```

The overall accuracy of the kNN algorithm on the test set with $k = 5$ is about 97%.

Ex. 5. Use the `iris` data set to carry out k-means clustering. Compare the result to the actual classes and estimate the clustering accuracy.

Response.

```
df <- iris

df.X <- df[,c(1:4)]
df.y <- df[,5]

clusters <- kmeans(df.X, centers = 3, nstart = 4)
```

in `clusters`, `cluster` is a vector with the predicted class. `centers` is the location of each center. Comparing actual versus predicted clusters:

```
res <- data.frame("pred" = clusters$cluster, "actual" = df.y)
table(res$pred, res$actual)
```

```
##
##     setosa versicolor virginica
##   1      0          2        36
##   2      0         48        14
##   3     50          0         0
```

Cluster 1 represents setosa, cluster 2 represents virginical, and cluster 3 represents veresicolor. Overall accuracy is approximately 89%.