# CMDA-3654

Homework 9

Eduardo Salvador

Due as a .pdf upload

## Problem 1: [50 pts] k-means clustering

Consider the Hotdog dataset shown in `hotdogs.csv`. Load in the data but ignore the first column as we'll pretend we know nothing other than the `Sodium` and `Calorie` content of the hotdogs.

a. Carry out K-means clustering with 2, 3, 4, 5 clusters. Don't forget to scaled the data first using:

```
#Loading libraries and reading csv file
library(cluster)
library(tidyverse)
hotdogs<-read.csv("/Users/eduardosalvador/Desktop/FINAL\ Spring\ Semester\ 2021/CMDA\ /Assignments/HW9/hotdogs
hotdogs.scaled <- scale(hotdogs [, 2:3])
#Using kmeans function to get kmean for 2,3,4,5 clusters
kmeans2<-kmeans(hotdogs.scaled,centers=2,nstart = 25)
kmeans2
kmeans3<-kmeans(hotdogs.scaled,centers=3,nstart = 25)
kmeans3
kmeans4<-kmeans (hotdogs.scaled,centers=4,nstart = 25)
kmeans4
kmeans5<-kmeans (hotdogs.scaled,centers=5,nstart = 25)
kmeans5
```

b. Plot the clusters from part (a) using `ggplot()` and assigning different colors and plot characters to the clusters found using `kmeans()`.

**Hint:** If `km.result <- kmeans(....)`, then the cluster assignments are in `km.result$cluster`, then simply add this to a new data.frame:

```
#Converting to scaled and dataframe
hotdogs2.scaled <- cbind(hotdogs.scaled, "cluster" = as.factor(kmeans2$cluster))
hotdogs2_df<-data.frame(hotdogs2.scaled)

hotdogs3.scaled <- cbind(hotdogs.scaled, "cluster" = as.factor(kmeans3$cluster))
hotdogs3_df<-data.frame(hotdogs3.scaled)

hotdogs4.scaled <- cbind(hotdogs.scaled, "cluster" = as.factor(kmeans4$cluster))
hotdogs4_df<-data.frame(hotdogs4.scaled)

hotdogs5.scaled <- cbind(hotdogs.scaled, "cluster" = as.factor(kmeans5$cluster))
hotdogs5_df<-data.frame(hotdogs5.scaled)
#Creating ggplot for kmean cluster 2
ggplot(hotdogs2_df,aes(x=Sodium,y=Calories,color=cluster))+theme_bw()+geom_point()

#Creating ggplot for kmean cluster 3
ggplot(hotdogs3_df,aes(x=Sodium,y=Calories,color=cluster))+theme_bw()+geom_point()

#Creating ggplot for kmean cluster 4
ggplot(hotdogs4_df,aes(x=Sodium,y=Calories,color=cluster))+theme_bw()+geom_point()

#Creating ggplot for kmean cluster 5
ggplot(hotdogs5_df,aes(x=Sodium,y=Calories,color=cluster))+theme_bw()+geom_point()
```

and then use `ggplot()` accordingly to make the plot.

c. Install and eanble the following R libraries: `cluster`, `NbClust`, `factoextra`. Then use the `fviz_cluster()` function to vizualize the clusters you made in part (a). Here is an example of how to use it.

```
library(ggplot2)
library(factoextra)
library(cluster)
library(NbClust)
fviz_cluster(kmeans2, data = hotdogs2.scaled)
fviz_cluster(kmeans3, data = hotdogs3.scaled)
fviz_cluster(kmeans4, data = hotdogs4.scaled)
fviz_cluster(kmeans5, data = hotdogs5.scaled)
```

**Determining the optimal number of clusters.**

Recall that the basic idea behind partitioning methods, such as k-means clustering, is to define clusters such that the total within-cluster variation or total within-cluster sum of squares is minimized: That is,

$$minimize\left(\sum_{i=1}^{k} W(C_k)\right)$$

There are a number of methods that we can use to determine the optimal number of clusters that should be used. One such method is called the Elbow Method which plots the total within-cluster sum of squares versus number of clusters.

We can obtain the total within-cluster sum of squares using `km.result$tot.withinss`.

The elbow method suggest that you find the "elbow" of this plot, where the total within-cluster sum of squares essentially stops reducing signficantly as the number of clusters grows.

Thankfully I can save you from this the long way by telling you about a function that can do this automatically.

d. Use the function `fviz_nbclust(hotdogs.scaled, kmeans, method = "wss")` to produce a plot using the Elbow method. Using this plot, determine how many clusters you think should be used.

```
#Using function to produce a plot using the Elbow method

#fviz_nbclust(hotdogs2.scaled, kmeans, method = "wss")

#For some reason it didn't let me knit using fviz_nbclust function but. plot did come out

#I believe I should use 4 clusters since the line after 4 becomes increasingly similar.
```

**Discussion:** There are other methods for determining the optimal number of clusters that are more sophisticated such as the average silhouette method and the gap statistic method, but these are beyond the scope of this course.

## Problem 2: [50 pts] Hierarchical Clustering.

Consider the mtcars dataset.

a. Using Euclidean distance as the dissimilarity measure, perform hierarchical clustering on the data, with (i) Complete Linkage, (ii) Average Linkage, and (iii) Single Linkage. Don't forget that you need to scale the data and compute the distances before using the hclust() function.

```
#Scaling mt cars data
mtcars.scaled<-scale(mtcars)
#Get distance mtcars
dist.mtcars<-dist(mtcars.scaled)

#Performing hierarchical clustering with Complete Linkage
complete.mtcars<-hclust(dist.mtcars)

#Performing hierarchical clustering with Average Linkage
average.mtcars<-hclust(dist.mtcars,method = "average")

#Performing hierarchical clustering with Single Linkage
single.mtcars<-hclust(dist.mtcars,method = "single")
```

b. For all three methods in (a), cut the hierarchical clustering tree at 4 clusters and report the two-way table of the car name and the cluster it belongs to.

```
#Cutting the complete linkage tree at 4 clusters
complete.cut<-cutree(complete.mtcars,4)
#Reporting two-way table of the car name and the cluster
complete.table<-table(rownames(mtcars),complete.cut)

#Cutting the average linkage tree at 4 clusters
average.cut<-cutree(average.mtcars,4)
#Reporting two-way table of the car name and the cluster
average.table<-table(rownames(mtcars),average.cut)

#Cutting the single linkage tree at 4 clusters
single.cut<-cutree(single.mtcars,4)
#Reporting two-way table of the car name and the cluster
single.table<-table(rownames(mtcars),single.cut)

#Displaying tables
complete.table
```

```
                     complete.cut
                      1 2 3 4
  AMC Javelin         0 0 0 1
  Cadillac Fleetwood  0 0 0 1
  Camaro Z28          0 0 0 1
  Chrysler Imperial   0 0 0 1
  Datsun 710          0 1 0 0
  Dodge Challenger    0 0 0 1
  Duster 360          0 0 0 1
  Ferrari Dino        1 0 0 0
  Fiat 128            0 1 0 0
  Fiat X1-9           0 1 0 0
  Ford Pantera L      1 0 0 0
  Honda Civic         0 1 0 0
  Hornet 4 Drive      0 0 1 0
  Hornet Sportabout   0 0 0 1
  Lincoln Continental 0 0 0 1
  Lotus Europa        0 1 0 0
  Maserati Bora       1 0 0 0
  Mazda RX4           1 0 0 0
  Mazda RX4 Wag       1 0 0 0
  Merc 230            0 0 1 0
  Merc 240D           0 0 1 0
  Merc 280            0 0 1 0
  Merc 280C           0 0 1 0
  Merc 450SE          0 0 0 1
  Merc 450SL          0 0 0 1
  Merc 450SLC         0 0 0 1
  Pontiac Firebird    0 0 0 1
  Porsche 914-2       0 1 0 0
  Toyota Corolla      0 1 0 0
  Toyota Corona       0 0 1 0
  Valiant             0 0 1 0
  Volvo 142E          0 1 0 0
```

```
average.table
```

```
                     average.cut
                      1 2 3 4
  AMC Javelin         0 0 1 0
  Cadillac Fleetwood  0 0 1 0
  Camaro Z28          0 0 1 0
  Chrysler Imperial   0 0 1 0
  Datsun 710          1 0 0 0
  Dodge Challenger    0 0 1 0
  Duster 360          0 0 1 0
  Ferrari Dino        1 0 0 0
  Fiat 128            1 0 0 0
  Fiat X1-9           1 0 0 0
  Ford Pantera L      0 0 0 1
  Honda Civic         1 0 0 0
  Hornet 4 Drive      0 1 0 0
  Hornet Sportabout   0 0 1 0
  Lincoln Continental 0 0 1 0
  Lotus Europa        1 0 0 0
  Maserati Bora       0 0 0 1
  Mazda RX4           1 0 0 0
  Mazda RX4 Wag       1 0 0 0
  Merc 230            0 1 0 0
  Merc 240D           0 1 0 0
  Merc 280            0 1 0 0
  Merc 280C           0 1 0 0
  Merc 450SE          0 0 1 0
  Merc 450SL          0 0 1 0
  Merc 450SLC         0 0 1 0
  Pontiac Firebird    0 0 1 0
  Porsche 914-2       1 0 0 0
  Toyota Corolla      1 0 0 0
  Toyota Corona       0 1 0 0
  Valiant             0 1 0 0
  Volvo 142E          1 0 0 0
```

```
single.table
```

```
                    single.cut
                     1 2 3 4
  AMC Javelin        0 1 0 0
  Cadillac Fleetwood 0 1 0 0
  Camaro Z28         0 1 0 0
  Chrysler Imperial  0 1 0 0
  Datsun 710         1 0 0 0
  Dodge Challenger   0 1 0 0
  Duster 360         0 1 0 0
  Ferrari Dino       1 0 0 0
  Fiat 128           1 0 0 0
  Fiat X1-9          1 0 0 0
  Ford Pantera L     0 0 1 0
  Honda Civic        1 0 0 0
  Hornet 4 Drive     1 0 0 0
  Hornet Sportabout  0 1 0 0
  Lincoln Continental 0 1 0 0
  Lotus Europa       1 0 0 0
  Maserati Bora      0 0 0 1
  Mazda RX4          1 0 0 0
  Mazda RX4 Wag      1 0 0 0
  Merc 230           1 0 0 0
  Merc 240D          1 0 0 0
  Merc 280           1 0 0 0
  Merc 280C          1 0 0 0
  Merc 450SE         0 1 0 0
  Merc 450SL         0 1 0 0
  Merc 450SLC        0 1 0 0
  Pontiac Firebird   0 1 0 0
  Porsche 914-2      1 0 0 0
  Toyota Corolla     1 0 0 0
  Toyota Corona      1 0 0 0
  Valiant            1 0 0 0
  Volvo 142E         1 0 0 0
```

c. We can plot the dendrograms easily enough by doing the following:

```
plot(complete.mtcars, labels = rownames(mtcars),
     main = "Cluster Dendrogram (Complete Linkage)")
```

Where the above would plot the dendrogram for the Complete Linkage case. Provide this plot and repeat the above for the other 2 cases from part (a). Alternatively we can use the following library that makes use of `ggplot2`, called `ggdendro`.

```
library(ggplot2)
library(ggdendro)
# Vertical
ggdendrogram(complete.mtcars) + labs(title = "Cluster Dendrogram (Complete Linkage)")
# or horizontal with a different theme
ggdendrogram(complete.mtcars, rotate = T, theme_dendro = F) +
  labs(title = "Cluster Dendrogram (Complete Linkage)")

# Vertical Average Linkage
ggdendrogram(average.mtcars) + labs(title = "Cluster Dendrogram (Average Linkage)")
# or horizontal with a different theme Average Linkage
ggdendrogram(average.mtcars, rotate = T, theme_dendro = F) +
  labs(title = "Cluster Dendrogram (Average Linkage)")
```

```
# Vertical Single Linkage
ggdendrogram(single.mtcars) + labs(title = "Cluster Dendrogram (Single Linkage)")
# or horizontal with a different theme Single Linkage
ggdendrogram(single.mtcars, rotate = T, theme_dendro = F) +
  labs(title = "Cluster Dendrogram (Single Linkage)")
```

Some alternative tools for plotting & customizing dendrograms can be found here: http://www.sthda.com/english/wiki/beautiful-dendrogram-visualizations-in-r-5-must-known-methods-unsupervised-machine-learning

Section 5 has some really cool examples of how to colorize clusters, etc.

Another awesome example can be found here: https://www.r-graph-gallery.com/340-custom-your-dendrogram-with-dendextend/

d. Use the elbow method to determine the optimal number of clusters that you should use. This works the same basic way as in problem 1, but the call is slightly different because it needs to use the `hcut()` function (named without the ()) as an option as seen below.

```
fviz_nbclust(mtcars.scaled, hcut, method = "wss")
```

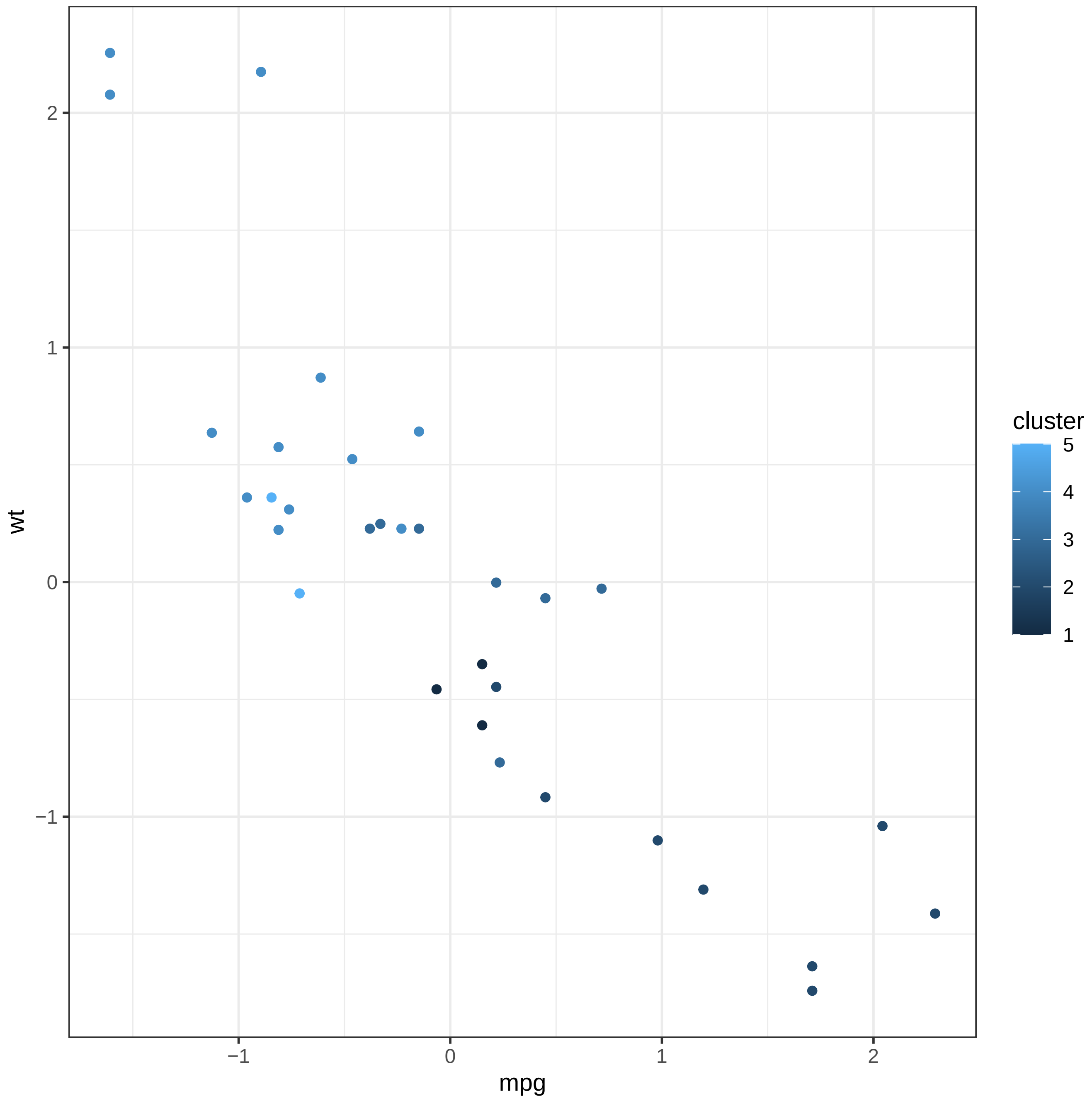
e. Add colored rectangles around the clusters you have determined in part (d) to the dendrogram in part (c). You can simply run the following line after the plot in (c) if you used the first method (other methods required other functions).

```
library(dendextend)

#Adding plot to dendrogramof complete linkage
plot(complete.mtcars,labels=rownames(mtcars),main="Cluster Dendrogram (Complete Linkage)")
rect.hclust(complete.mtcars,
            k = 4, # replace with whatever you decided based upon (d)
            border = c("red","blue","purple","magenta")
            )

#Adding plot to dendrogramof average linkage
plot(average.mtcars,labels=rownames(mtcars),main="Cluster Dendrogram (Average Linkage)")
rect.hclust(average.mtcars,
            k = 4, # replace with whatever you decided based upon (d)
            border = c("red","blue","purple","magenta")
            )

#Adding plot to dendrogramof complete linkage
plot(single.mtcars,labels=rownames(mtcars),main="Cluster Dendrogram (Single Linkage)")
rect.hclust(single.mtcars,
            k = 4, # replace with whatever you decided based upon (d)
            border = c("red","blue","purple","magenta")
            )
```

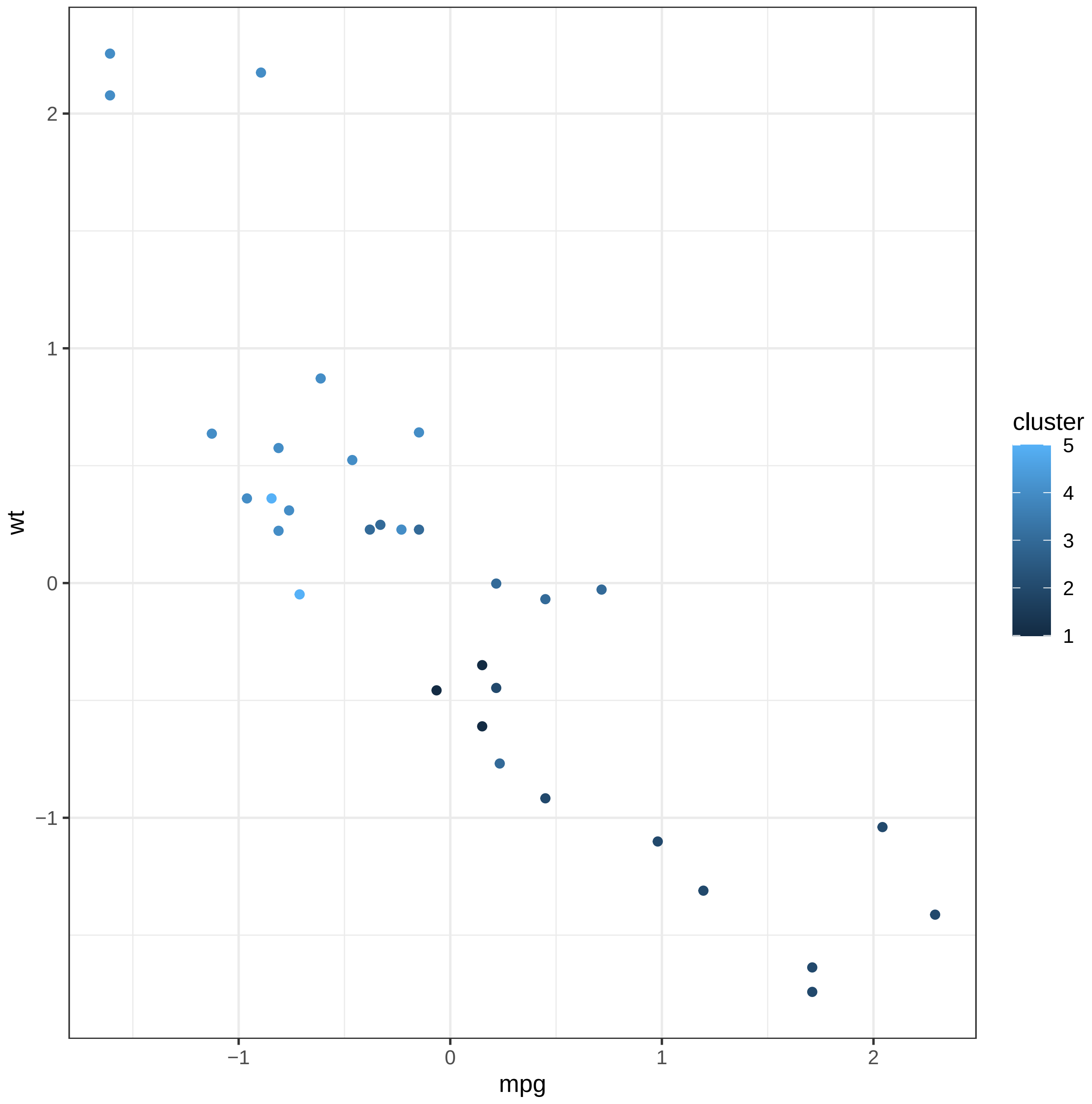
f. Use `cutree()` to obtain the cluster assignments using your decision in (d). Recode this as a factor object. Plot the `mpg` versus `wt` and use the different colors according to your cluster assignment.

```
library(ggplot2)
#Using cutree() to obtain the cluster assignments for every linkage and converting to dataframe
comp.cluster.mtcars <- cbind(mtcars.scaled, "cluster" = as.factor(cutree(complete.mtcars,h=4)))
comp.cluster.mtcars_df<-data.frame(comp.cluster.mtcars)
```

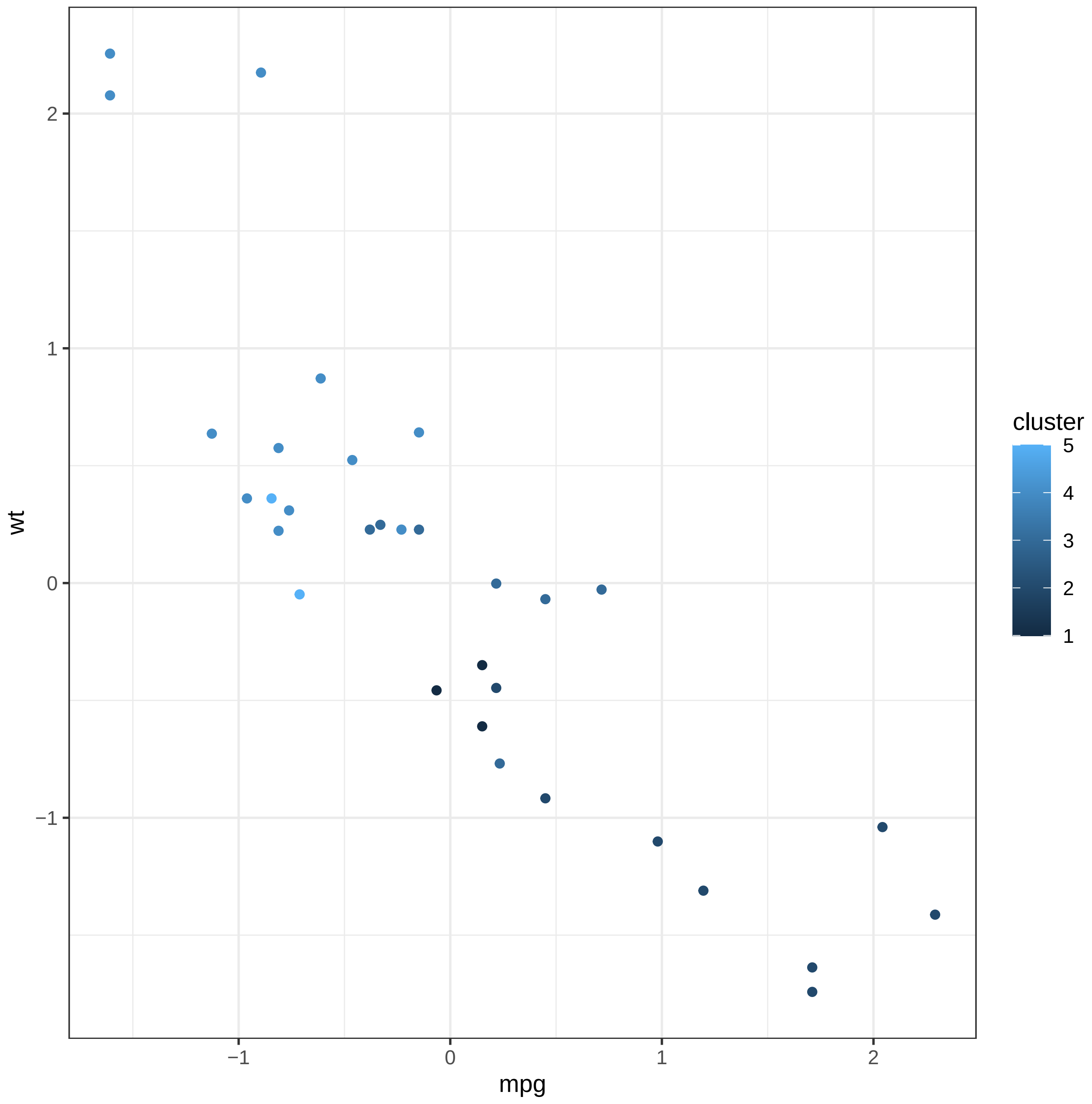
```
av.cluster.mtcars <- cbind(mtcars.scaled, "cluster" = as.factor(cutree(average.mtcars,h=4)))
av.cluster.mtcars_df<-data.frame(comp.cluster.mtcars)

sing.cluster.mtcars <- cbind(mtcars.scaled, "cluster" = as.factor(cutree(single.mtcars,h=4)))
sing.cluster.mtcars_df<-data.frame(comp.cluster.mtcars)

#Plotting mpg vs wt using different colors for complete linkage
ggplot(comp.cluster.mtcars_df,aes(x=mpg,y=wt,color=cluster))+theme_bw()+geom_point()
```



```
#Plotting mpg vs wt using different colors for average linkage
ggplot(av.cluster.mtcars_df,aes(x=mpg,y=wt,color=cluster))+theme_bw()+geom_point()
```



```
#Plotting mpg vs wt using different colors for single linkage
ggplot(sing.cluster.mtcars_df,aes(x=mpg,y=wt,color=cluster))+theme_bw()+geom_point()
```



**Discussion:** While you can see some clustering that makes sense, some of the clusters seem to intermixed with each other. Remember, the clustering was determined not using only the `mpg` and `wt` variables alone, but using all of the information from all of the variables in the dataset. So the clusters are formed in multidimensional space. This can be difficult to visualize when the number of dimensions is bigger than 3.

One solution is to rotate the multidimensional variable coordinate system into the principal component coordinate system as we will show in the next problem. This means that if most of the variation is in the first 2 principal components, it might be possible to more easily see the clusters in higher dimensional space using the lower dimensional representation.

## Problem 3: [15 pts Extra Credit] PCA + Hierarchical Clustering.

Consider the `mtcars` dataset once again.

a. Use PCA to rotate the observations from the `mtcars` dataset into a new coordinate system using the principal components. Remember that you have to do either `scale. = TRUE` if using `prcomp()` or `cor = TRUE` if using `princomp()`. We want the component scores which will either be `pca.result$x` if you used `prcomp()` or `pca.result$scores` if you use `princomp()`.

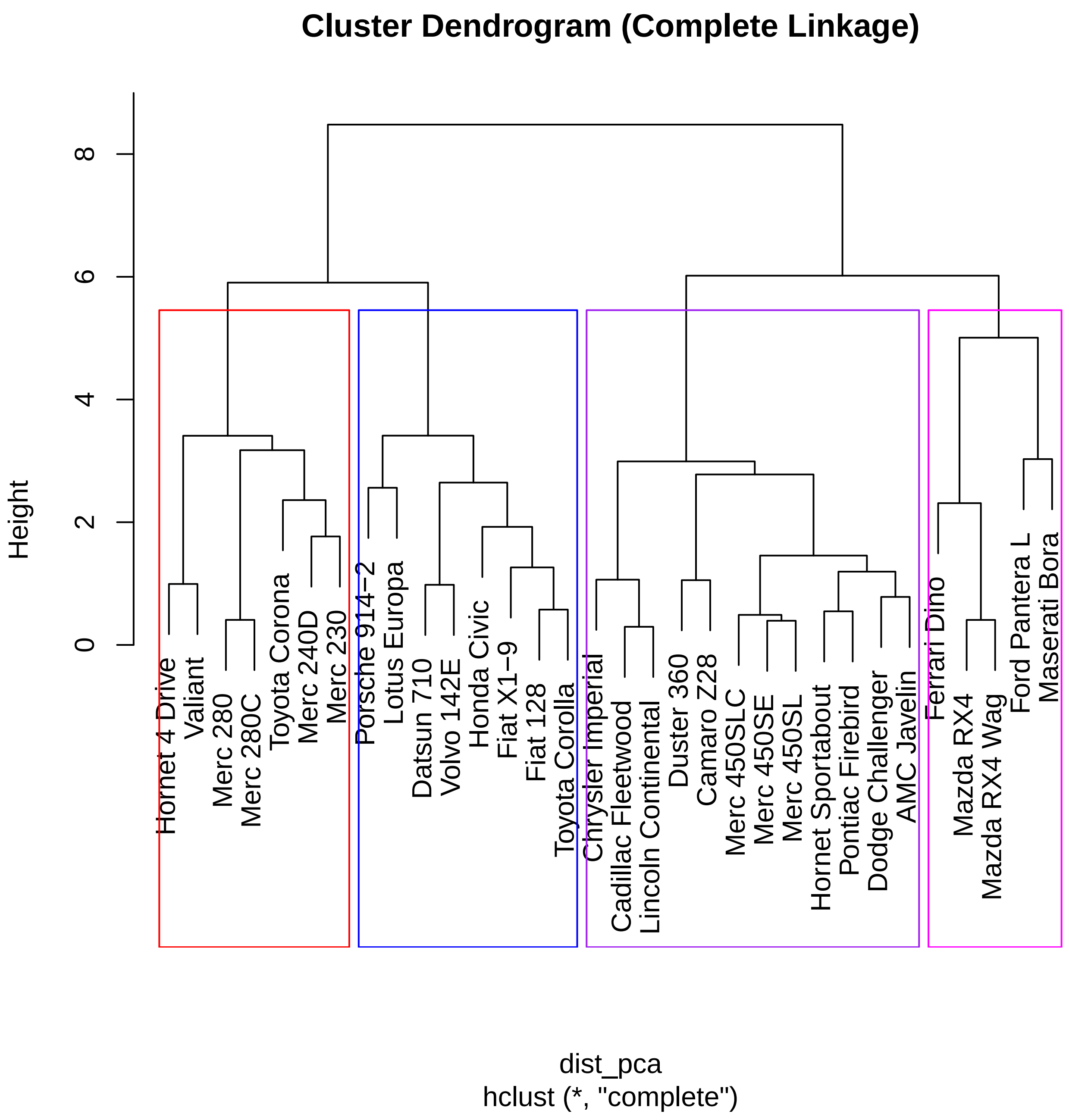
```
#PCA+Hierarchical Clustering using prcomp
pca.result<-prcomp(mtcars,scale. = T)
pca.result$x
```

```
                              PC1        PC2        PC3          PC4
Mazda RX4           -0.6468627420  1.7081142 -0.5917309  0.113702214
Mazda RX4 Wag       -0.6194831460  1.5256219 -0.3763013  0.199121210
Datsun 710          -2.7356242748 -0.1441501 -0.2374391 -0.245215450
Hornet 4 Drive      -0.3068606268 -2.3258038 -0.1336213 -0.503800355
Hornet Sportabout    1.9433926844 -0.7425211 -1.1165366  0.074461963
Valiant             -0.0552534228 -2.7421229  0.1612456 -0.975167425
Duster 360           2.9553851233  0.3296133 -0.3570461 -0.051529216
Merc 240D           -2.0229593244 -1.4421056  0.9290295 -0.142129082
Merc 230            -2.2513839535 -1.9522879  1.7689364  0.287210957
Merc 280            -0.5180912217 -0.1594610  1.4692603  0.066263362
Merc 280C           -0.5011860079 -0.3187934  1.6570701  0.094357222
Merc 450SE           2.2124096339 -0.6727099 -0.3694707 -0.129797905
Merc 450SL           2.0155715693 -0.6724606 -0.4768341 -0.210991001
Merc 450SLC          2.1147047372 -0.7891129 -0.2904620 -0.175332868
Cadillac Fleetwood   3.8383725118 -0.8149087  0.6370972  0.290505877
Lincoln Continental  3.8918495626 -0.7218314  0.7092612  0.405336898
Chrysler Imperial    3.5363862158 -0.4145024  0.5402468  0.665665306
Fiat 128            -3.7955510831 -0.2920783 -0.4161681  0.055191058
Honda Civic         -4.1870356784  0.6775721 -0.2035831  1.167526096
Toyota Corolla      -4.1675359344 -0.2748890 -0.4589124  0.183313028
Toyota Corona       -1.8741790870 -2.0864529  0.1543265  0.050514126
Dodge Challenger     2.1504414942 -0.9982442 -1.1503639 -0.584982249
AMC Javelin          1.8340369797 -0.8921886 -0.9472872  0.005694071
Camaro Z28           2.8434957523  0.6701037 -0.1605593  0.814340105
Pontiac Firebird     2.2105479148 -0.8600504 -1.0279577  0.146420497
Fiat X1-9           -3.5176818134 -0.1192950 -0.4464716 -0.013427353
Porsche 914-2       -2.6095003965  2.0141425 -0.8172519  0.568564789
Lotus Europa        -3.3323844512  1.3568877 -0.4467167 -1.153197531
Ford Pantera L       1.3513346957  3.4448780 -0.1343943  0.590098358
Ferrari Dino        -0.0009743305  3.1683750  0.3957610 -0.938933017
Maserati Bora        2.6270897605  4.3107016  1.3315940 -0.877332804
Volvo 142E          -2.3824711412  0.2299603  0.4052798  0.223549117
                             PC5           PC6         PC7          PC8
Mazda RX4           -0.945523363 -0.0169873733 -0.42648652 -0.009631217
Mazda RX4 Wag       -1.016680740 -0.2417246434 -0.41620046 -0.084520213
Datsun 710           0.398762288 -0.3487678138 -0.60884146  0.585255765
Hornet 4 Drive       0.549208936  0.0192969984 -0.04036075 -0.049583029
Hornet Sportabout    0.207515698  0.1491927606  0.38350816 -0.160297757
Valiant              0.211665375 -0.2438358546 -0.29464160  0.256612420
Duster 360           0.343847875  0.7126920868 -0.13607792 -0.171103449
Merc 240D           -0.316651386 -0.0009889391  0.63946214  0.163156195
Merc 230            -0.333682355 -0.3338703384  0.62201034 -0.105779936
Merc 280            -0.069624161  0.8165308365  0.16117090  0.099983313
Merc 280C           -0.148803650  0.7308383757  0.09254430  0.197306566
Merc 450SE          -0.378611141  0.1317014762 -0.01645498 -0.194092435
Merc 450SL          -0.355611763  0.2400263805  0.05123623 -0.329669990
Merc 450SLC         -0.432140303  0.1801997325 -0.06675316 -0.119252582
Cadillac Fleetwood  -0.048245223 -0.8844735483 -0.16615296  0.138398783
Lincoln Continental  0.003899176 -0.8625868981 -0.19250873  0.129305868
Chrysler Imperial    0.208027112 -0.6536447300  0.03449804 -0.391104141
Fiat 128             0.219981109 -0.4675796343 -0.03749941 -0.625278746
Honda Civic          0.097674091  0.5180554279 -0.25316291 -0.395045565
Toyota Corolla       0.222152228 -0.3171521124  0.06617540 -0.853947085
Toyota Corona        0.039299002  0.7236992559 -0.28027808  0.207237627
Dodge Challenger    -0.226237802  0.1062181942  0.09489585  0.316055390
AMC Javelin         -0.252565496  0.2888101997  0.08161916  0.321900593
Camaro Z28           0.389118986  0.9468795171 -0.21157976  0.038657331
Pontiac Firebird     0.299261925 -0.1983310387  0.47269865 -0.234144182
Fiat X1-9            0.206753365 -0.1449905641 -0.35850305  0.089109764
Porsche 914-2       -0.597313744 -0.3394265065  0.82032965  0.634987241
Lotus Europa         0.694667640  0.0165037718  0.51018011  0.004140777
Ford Pantera L       1.101648091 -0.1746156635  0.41358868  0.609167214
Ferrari Dino        -0.848833976 -0.0097569921  0.02967883  0.014187801
Maserati Bora        0.455265189 -0.0156094416 -0.18813730 -0.558646792
Volvo 142E           0.321777017 -0.3263029217 -0.77995741  0.476634473
                            PC9        PC10         PC11
Mazda RX4            0.14642303 -0.06670350  0.179693570
Mazda RX4 Wag        0.07452829 -0.12692766  0.088644265
Datsun 710          -0.13122859  0.04573787 -0.094632914
Hornet 4 Drive       0.22021812 -0.06039981  0.147611269
Hornet Sportabout   -0.02117623 -0.05983003  0.146406899
Valiant             -0.03222907 -0.20165466  0.019545064
Duster 360          -0.17844547  0.36086641  0.171863162
Merc 240D            0.37698418  0.29086529 -0.019090358
Merc 230            -0.86455356 -0.11597058  0.159688512
Merc 280             0.54092449 -0.22093750 -0.124486227
Merc 280C            0.30876072 -0.34417564 -0.034578568
Merc 450SE          -0.05614966 -0.06531727 -0.396445135
Merc 450SL          -0.20501055 -0.10761308 -0.197616838
Merc 450SLC         -0.38704169 -0.21191036 -0.142498830
Cadillac Fleetwood   0.19333387  0.06184979  0.262886205
Lincoln Continental  0.19523562  0.12094849  0.039191100
Chrysler Imperial    0.27447514  0.27588169 -0.224420191
Fiat 128             0.10550311 -0.02717077 -0.208865888
Honda Civic          0.23711675 -0.15433928  0.246835364
Toyota Corolla      -0.11313627 -0.12606845 -0.031747839
Toyota Corona       -0.44646972  0.51147635  0.063679725
Dodge Challenger     0.10435633 -0.13641143  0.049594456
AMC Javelin         -0.12237636 -0.29628634  0.045293027
Camaro Z28          -0.05282991  0.32624525 -0.099386307
Pontiac Firebird     0.20849043  0.01547674  0.122593248
Fiat X1-9           -0.02228967 -0.08414018 -0.005746448
Porsche 914-2       -0.12999660  0.34968156 -0.111596656
Lotus Europa         0.29680350  0.23980308  0.030015592
Ford Pantera L      -0.23280792 -0.50262890 -0.042242570
Ferrari Dino         0.09813571  0.14491815  0.043006835
Maserati Bora       -0.34081133  0.04706368  0.062135486
Volvo 142E          -0.04473670  0.11767108 -0.145329008
```

b. For the principal component "variables" for the cars (which are the component scores), use the hierarchical clustering techniques that you learned in Problem 2. Use this to determine the optimal number of clusters, show the dendrogram and put a box around these clusters. **Use Complete Linkage only.** How does this compare with your answer result from the previous problem when you used complete linkage on the original variables?

```
#Using hierarchical clustering techniques from problem 2 fviz_eig() function
```

```r
#fviz_eig(pca.result)
#It won't knit using fviz_eig function but graph looks good

#Getting distance for every pca variable score
dist_pca<-dist(pca.result$x)
#Clustering based on distance
pca.cluster.comp<-hclust(dist_pca)

#Plotting dendrogram
plot(pca.cluster.comp,labels=rownames(mtcars),main="Cluster Dendrogram (Complete Linkage)")
rect.hclust(pca.cluster.comp,
            k = 4, # replace with whatever you decided based upon (d)
            border = c("red","blue","purple","magenta")
            )
```



```r
#It gives me the same answer as the previous problem when used complete linkage
```

c. Using `k = 4` for the number of clusters, plot `PC2` versus `PC1` and use different colors to show the clusters. You will need to use `cutree()` to get the cluster assignments. Can you see the clusters a bit better than you did compared to plotting `mpg` versus `wt` in the previous problem?

```
#Getting cluster assignments using cutree()
as_cluster<-as.factor(cutree(pca.cluster.comp,h=4))
```

---