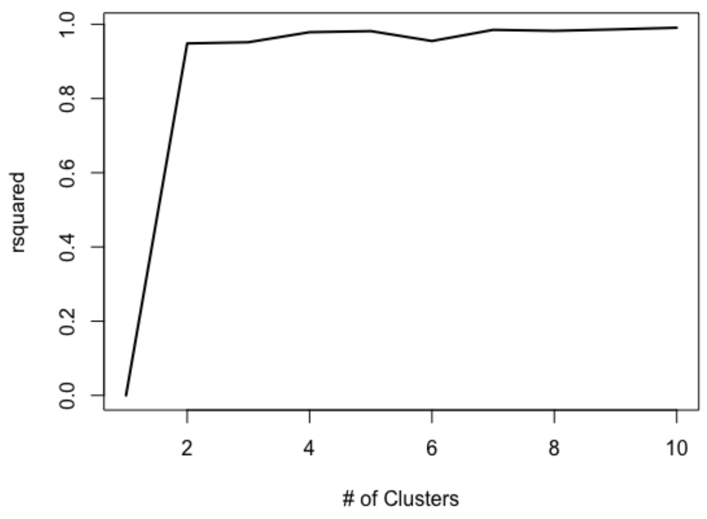
Exercise 1: Read the data in the appendix into R. You will examine the data in 3d:

```
library(car)
library(car)
scatter3d(x1~x2+x3, data=dat, neg.res.col="white",
pos.res.col="white", surface.alpha=0.0)
```

a. Use kmeans to determine the optimal number of clusters, and then plot the 3d plot.

Solution:
The kmeans algorithm simply finds the optimal partition of the data which reduces the intra-class variance (sum of squared distances from cluster centroids). With that in mind determining an optimal number of clusters is subjective, however we can make an informed decision on how many clusters we should use by examining the ratio between the sum of squares between clusters and the sum of squares total (like r^2). The greater this ratio, the more of the variance is explained by our clustering. Simulating kmeans for 1 to 10 clusters and computing the r^2 we get the following,

Figure 1: Plot of r^2 up to 10 Clusters.



Clearly we can see that after 2 clusters we are seeing marginal returns on the amount of variance explained. Comparing the 2 clustering and the data, it seems as though 2 clusters is satisfactory, and likely the true signal behind whatever system generated the data.

Figure 2: 3D Data Plot

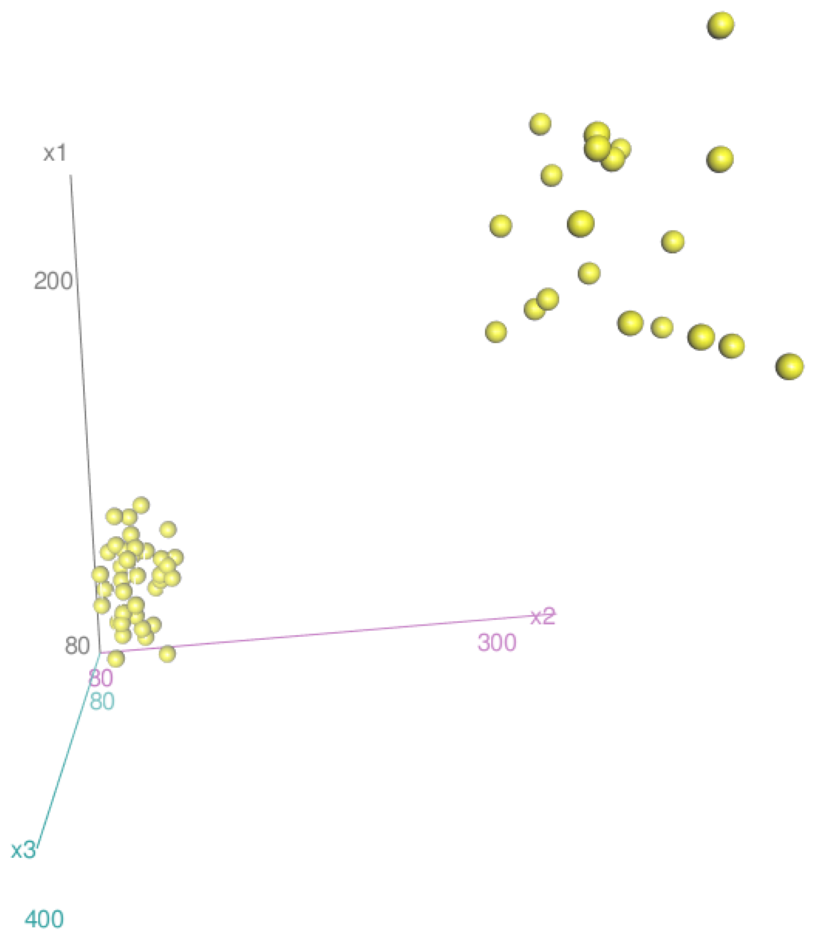
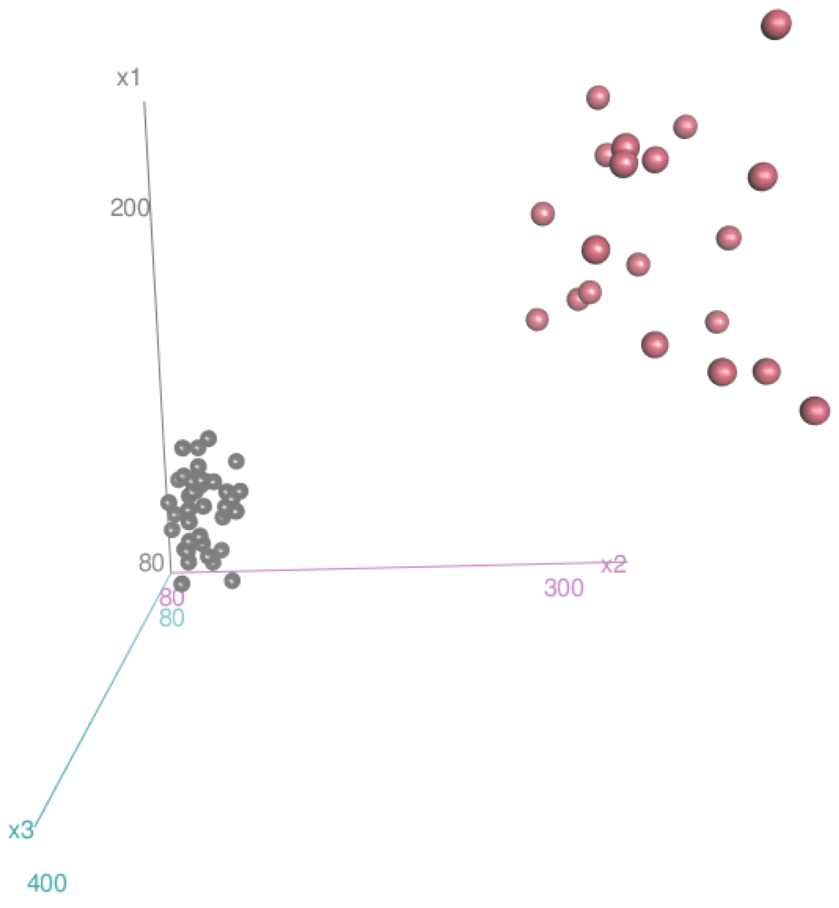


Figure 3: 3D Data Plot With 2 Clusters



b. Now use a hierarchical method. Try both complete and single linkage. Do you get the same clustering in the same order for both linkages?

Solution:
Using an agglomerative hierarchical method with the hclust() function we got the following dendrograms for the given data, using both linkages and euclidean distances,

Figure 4: Single Linkage

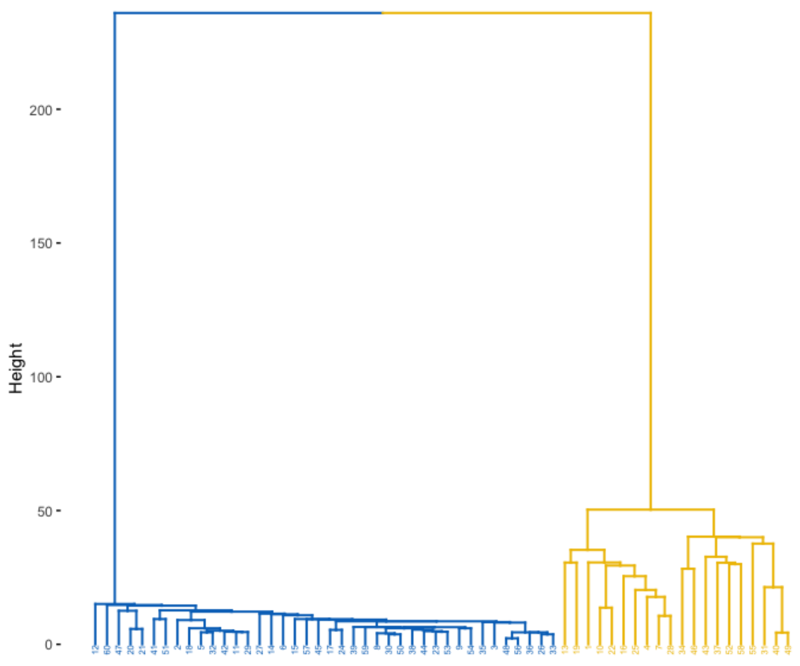
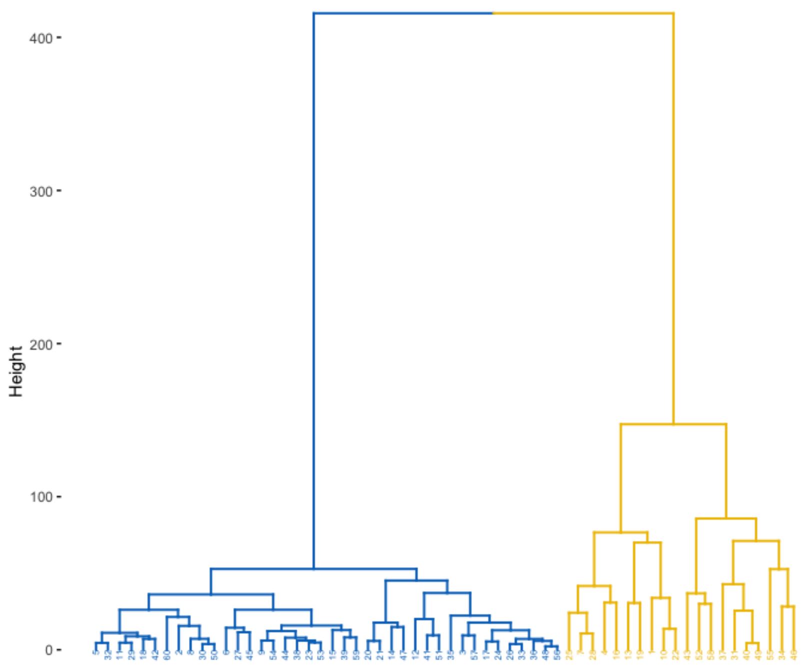
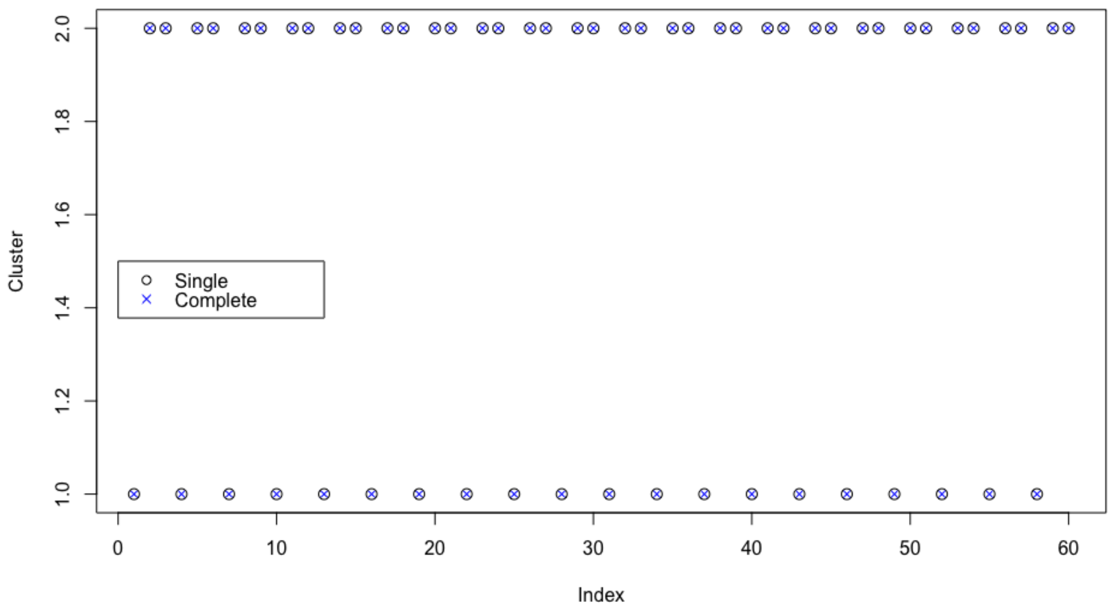


Figure 5: Complete Linkage



The following figure shows that each clustering method resulted in the same clusters, however using the order call on the dendrogram object we can see that the order is not the same,

Figure 6: Checking Clusters



Code:

```
library(ade4)

## Generating Distance
D <- dist(dat)
## Generating Dendrogram
tmp <- hclust(D, method="single")
tmp2 <- hclust(D, method="complete", )

## Colored Clustered Dendrogram
library(factoextra)
cluster1 <- fviz_dend(tmp, cex = 0.4,
          k = 2, # Cut in 2 groups
          palette = "jco", # Color palette
)
plot(cluster1)

cluster2 <- fviz_dend(tmp2, cex = 0.4,
                      k = 2, # Cut in 2 groups
                      palette = "jco", # Color palette
)
```

```
plot(cluster2)

## Cluster Plot to show same cluster
library(dendextend)
cut1 <- cutree(tmp, k = 2)
cut2 <- cutree(tmp2, k = 2)
plot(cut2, cex = 1.25, ylab = 'Cluster')
points(cut1, col = 'blue',pch = 4, cex = .75 )
legend(0,1.5, c('Single', 'Complete'), col = c('black', 'blue'), pch = c(1, 4))

## Checking if the order is the same
isTRUE(tmp$order == tmp2$order)
## [1] FALSE
```

c. What is complete linkage? What is single linkage?

Solution:
Linkage describes how we measure distance between clusters (and clusters and single observations). Suppose A and B are clusters such that $x \in A$ and $y \in B$ then under single linkage we get the following,

$$d(A, B) = min(x, y).$$

Under complete complete linkage we get the following,

$$d(A, B) = max(x, y).$$

Exercise 2: a. Import the Woody Species data set.

Solution:
Code:

```
## Reading in Data
dat <- read.csv('WoodySpecies.csv', header=TRUE)
dat_t <- t(dat[,-1])
colnames(dat_t) <- dat[,1]
```

b. Now use kmeans to determine a 'reasonable' number of clusters. You should justify your choice of cluster numbers.

Solution:
Generating the r^2 plot similarly to the first problem we get the following plots,

Figure 7: Plot of r^2 up to 10 Clusters.

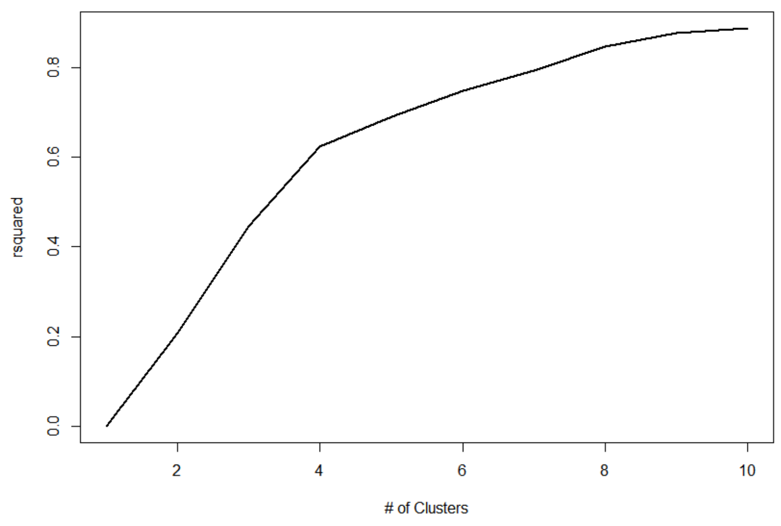
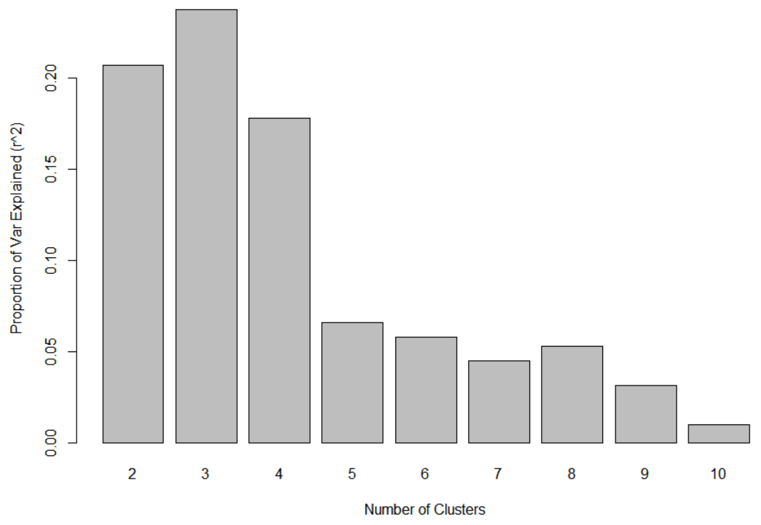


Figure 8: Proportion of r^2 for Each Cluster.(Scree-ish plot)



From here we can see that adding the fifth cluster contributes, marginally to the proportion of variance explained, and therefore I would explore clustering into 4 or

less groups.
Code:

```
## Simulating Clusters and Computing rSquared.
rsquared <- rep(NA, 10)
for (i in 1:10){
  tmp <- kmeans(dat_t, centers = i)
  rsquared[i] <- tmp$betweenss/tmp$totss
}
plot(rsquared, type="l", lwd=2, xlab = '# of Clusters')
barplot((rsquared[2:10] - rsquared[1:9]), names.arg = seq(2,10),
        ylab = 'Proportion of Var Explained (r^2)',
        xlab = 'Number of Clusters' )
```

c. Briefly, how does kmeans() work?

Solution:
The kmeans algorithm begins by randomly assigning the data to n(hyperparameter) different clusters. Then the centroid of each cluster is computed. The data are re-assigned to the same cluster as the nearest centroid. New centroids are computed and the data is reassigned again recursively. Eventually this algorithm converges, and sometimes it shows sensitivity to initial conditions which is why kmeans() runs several randomized instances of this algorithm.

d. Now cluster the data using any hierarchical clustering method you wish. Don't forget to think about a reasonable distance measure. Do you get similar clusters to when you used kmeans?

Solution:
Note that the data are stated to be plant densities from each 10 m^2 plot. With this mind I think an average linkage and normal euclidean distance might be the most appropriate, since are data are already an average in a sense, as opposed to actual plant counts. Generating the dendrogram we get the following,

Figure 9: Dendrogram for Woody Species Data from hclust() and 'average' Linkage

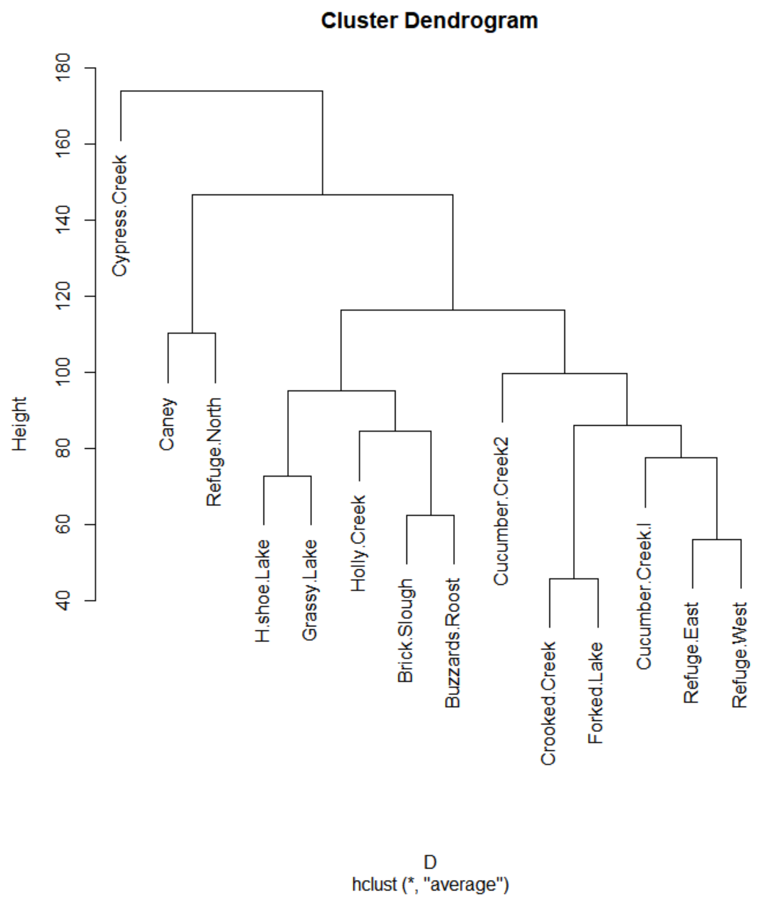


Figure 10: Cut with 3 Clusters (what was done in the paper.)

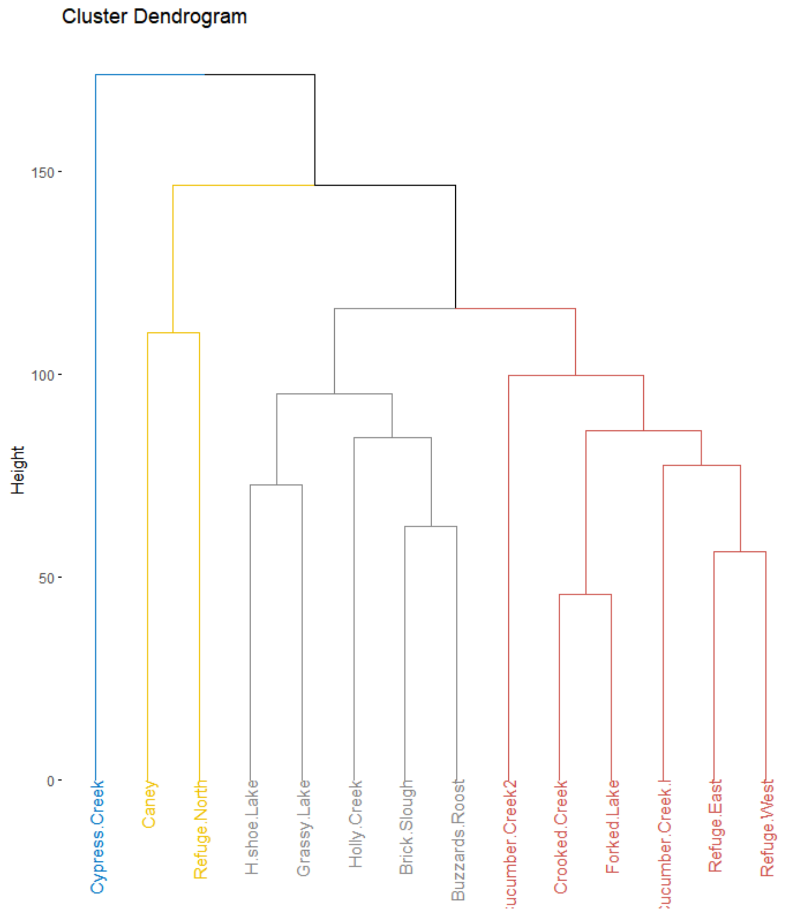
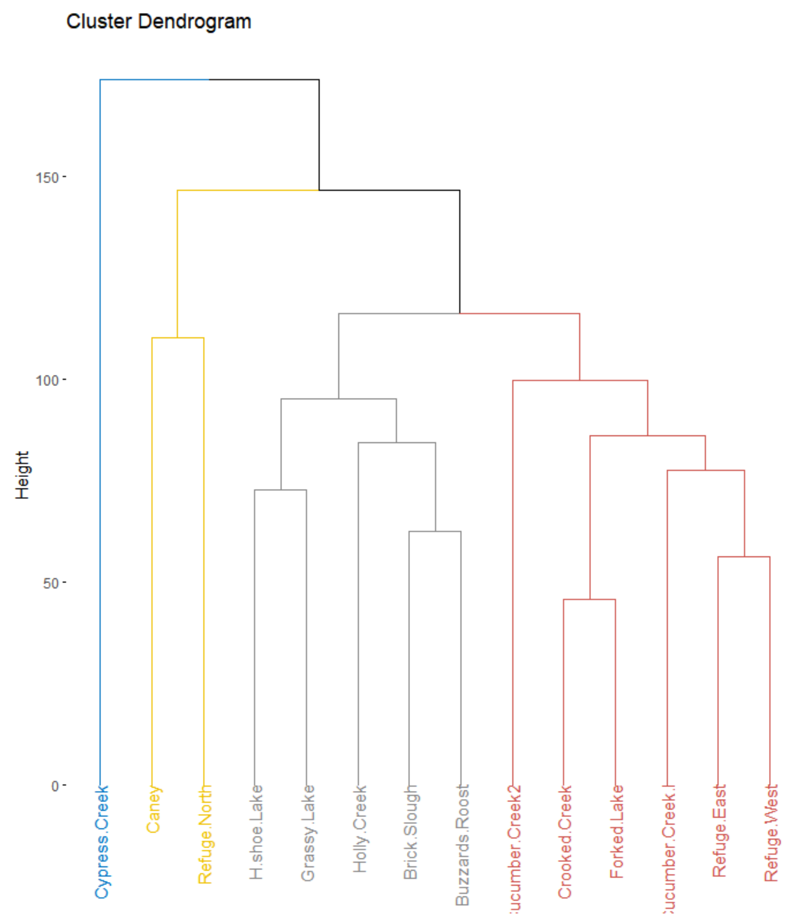


Figure 11: Cut with 4 Clusters



Code:

```
## Generating Distance
D <- dist(dat_t)

HclustWoodySpecies <- hclust(D, method = 'average')

## Generating Dendogram
plot(HclustWoodySpecies)
cluster <- fviz_dend(HclustWoodySpecies,
                     k = 3, # Cut in three groups
                     palette = "jco", # Color palette
)

plot(cluster)
cluster <- fviz_dend(HclustWoodySpecies,
                    k = 4, # Cut in four groups
                    palette = "jco", # Color palette
)
plot(cluster)
```

Comparing the two clustering methods, kmeans and the average linkage euclidean distance agglomerative hierarchical clustering produced the same clusters. This can be seen by reading the print out and noting that the observations are grouped in the same way (had a hard time visualizing this in a systematic way since the methods produced different labels for the clusters).

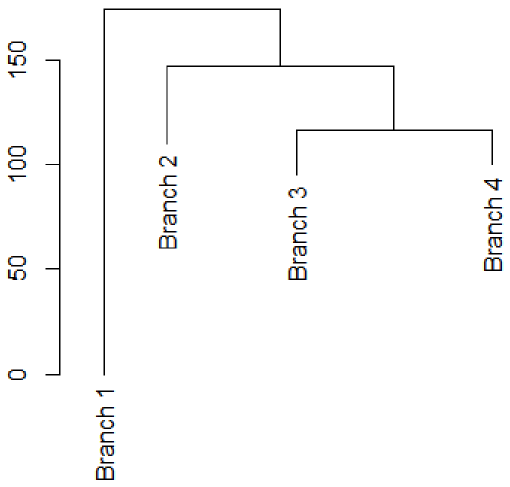
```
kmeansWoodySpecies <- kmeans(dat_t, centers = 4)
  > kmeansWoodySpecies$cluster
  Cypress.Creek    Crooked.Creek      Forked.Lake Cucumber.Creek.1
  3                4                4                4
  Cucumber.Creek2      Refuge.East      Refuge.West            Caney
  4                4                4                2
  Refuge.North      H.shoe.Lake     Brick.Slough      Grassy.Lake
  2                1                1                1
  Buzzards.Roost      Holly.Creek
  1                1
```

```
cutCompare <- cutree(HclustWoodySpecies, k = 4)
  > cutCompare
  Cypress.Creek    Crooked.Creek      Forked.Lake Cucumber.Creek.1
  1                2                2                2
  Cucumber.Creek2      Refuge.East      Refuge.West            Caney
  2                2                2                3
  Refuge.North      H.shoe.Lake     Brick.Slough      Grassy.Lake
  3                4                4                4
  Buzzards.Roost      Holly.Creek
  4                4
```

e. Use the cut() function to cut the tree at some level and plot the part of the dendrogram above that cut.

Solution:

Figure 12: Cut() with 4 Clusters



Code:

```
PrunedTree <- cut(as.dendrogram(HclustWoodySpecies), h = 115)
plot(PrunedTree$upper)
```

Exercise 3: Continue the analysis from the last problem (WoodySpecies data). Here you will look at cophenetic correlation and tanglegrams.

a. What is cophenetic correlation? What is it used for?

Solution:
A cophenetic distance, is the distance matrix for a set of data generated from the hierarchical cluster. Cophenetic correlation is what we use to compare the structure of the original distance matrix of the data, and the clustered distance of our data, similar to how a mantel test would be used.

b. Using the dendrogram from part (d), problem two, find the correlation between the cophenetic distance matrix and the regular distance matrix.

Solution:

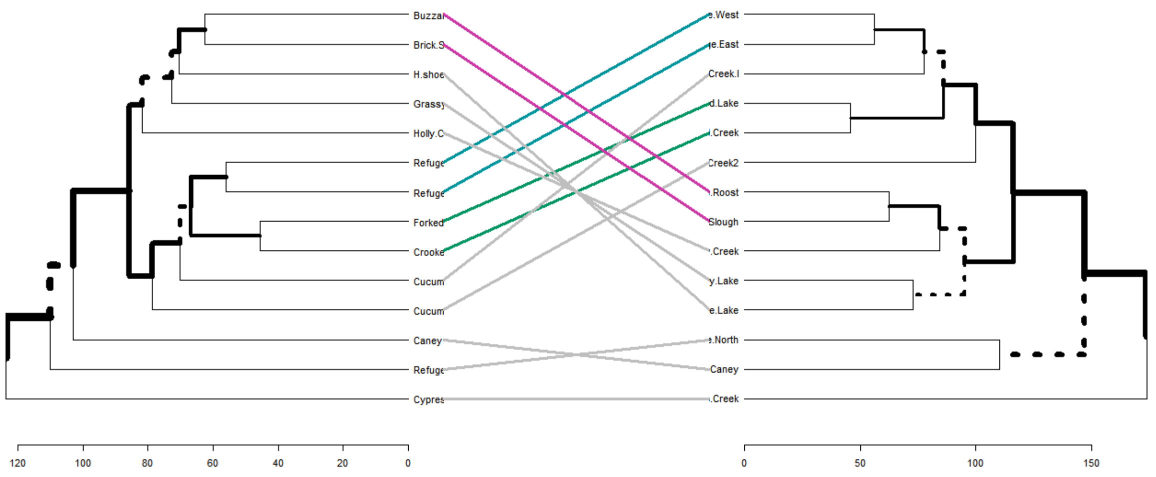
Code:

```
D <- dist(dat_t)
ClustD <- cophenetic(HclustWoodySpecies)
cor(ClustD, D)
## [1] 0.8223099
```

c. You should now compute another hierarchical clustering using a different method from what you used in the last problem and save the dendrogram. Using the tanglegram() function in dendextend package, examine whether this new dendrogram give similar clusters to the dendrogram from the last problem.

Solution:
Computing a new clustering using single linkage, and generating a tanglegram we get the following,

Figure 13: Tanglegram: Average Linkage vs. Single Linkage



Code:

```
D <- dist(dat_t)
HclustAverage <- hclust(D, method = 'average')
HclustSingle <- hclust(D, method = 'single')
tanglegram(HclustSingle, HclustAverage)
cor_cophenetic(HclustAverage, HclustSingle)
## [1] 0.9706721
```

From the tanglegram we can see that really the only observations that really separated between the two clustering paradigms where Cucumber.Creek1 and Cucumber.Creek2. Computing the cophenetic correlation we get a value of .97 which makes sense given the tanglegram, shows us generally the same clusters.

Exercise 4: a. What is a divisive hierarchical technique, as opposed to an agglomerative hierarchical technique?

Solution:
Divisive clustering begins by partitioning the data into one partition, then uses some rule to split the partition into two. This is then repeated until all the data is divided into their own partitions(like decision trees). Agglomerative clustering goes in the other direction by starting with each data in their own partition and using some rule to join partitions together until we form a single partition.

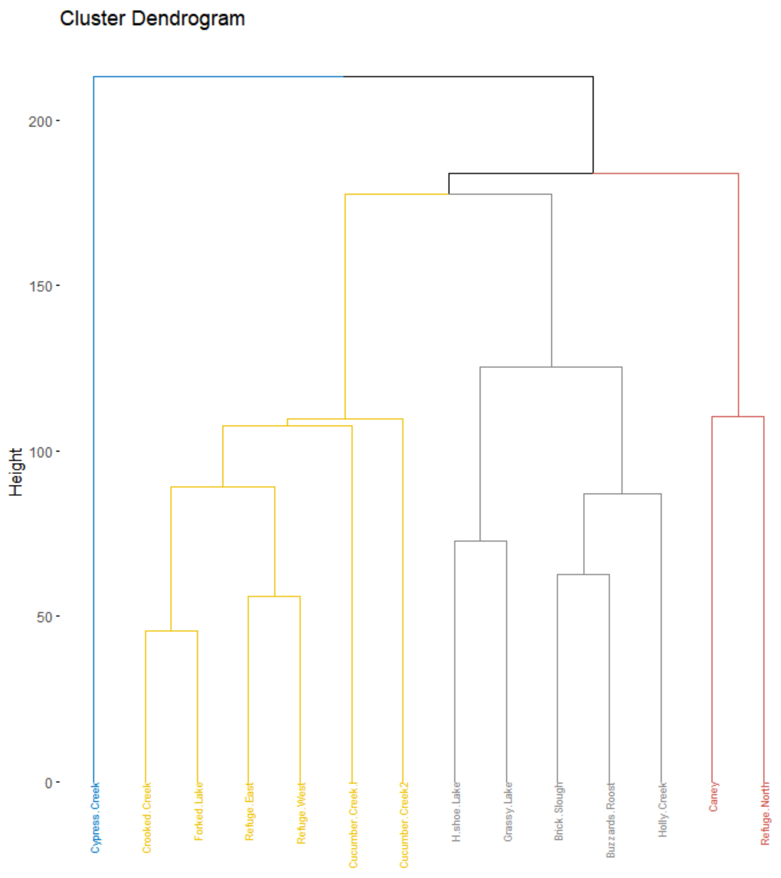
b. Use code similar to the code below to run DIANA on the Woody Species data.

```
library(cluster)
res.diana <- diana(distmat, stand = TRUE)

# Plot the dendrogram
library(factoextra)
fviz_dend(res.diana, cex = 0.5,
    k = 4, # Cut in four groups
    palette = "jco" # Color palette
    )
```

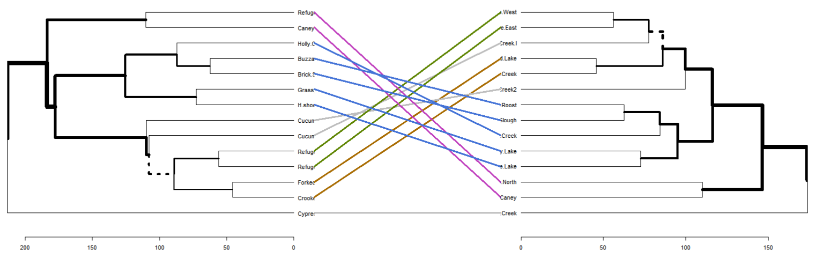
Solution:
Running the code to generate the plot for the divisive clustering paradigm,

Figure 14: Divisive Clustering of WoodySpecies Data.



Generating a tanglegram, and computing the cophenetic correlation to compare the clustering to the average linkage agglomerative clustering we found very similar clustering.

Figure 15: Divisive vs Agglomerative Clustering.



Code:

```
D <- dist(dat_t)
library(cluster)
res.diana <- diana(D, stand = TRUE)

# Plot the dendrogram
library(factoextra)
fviz_dend(res.diana, cex = 0.5,
          k = 4, # Cut in four groups
          palette = "jco" # Color palette
)

tanglegram(as.dendrogram(res.diana), HclustAverage)
cor_cophenetic(as.dendrogram(res.diana), HclustAverage)
## [1] 0.8989625
```

Exercise 5: Somewhere, find a dataset that you are interested in that compares multiple sites, countries, etc. Choose your favorite clustering method, then use it to cluster the dataset. How many clusters did you get? why did you pick that number of clusters? Do you think your approach did a good job of clustering?

Solution:
The data I decided to use comes from Kaggle and it contains information about water quality from 3276 different bodies of water. The goal of the data is to be able to predict the potability of water form metrics like pH, Hardness, Conductivity, Sulfates, and several other predictors. The data also contains labels for each of the observations so we do have a true clustering to compare our methods to.
A kmeans clustering showed that the data contains mainly 2 clusters. The following plots of cumulative and accumulative rsquared show that most of the variance is explained by 2 groups. The two cluster correctly classified 1062 of the 2011 usable observations, which is about as good as randomly assigning labels.

Figure 16: Cumulative RSquared from Kmeans Analysis on WaterData

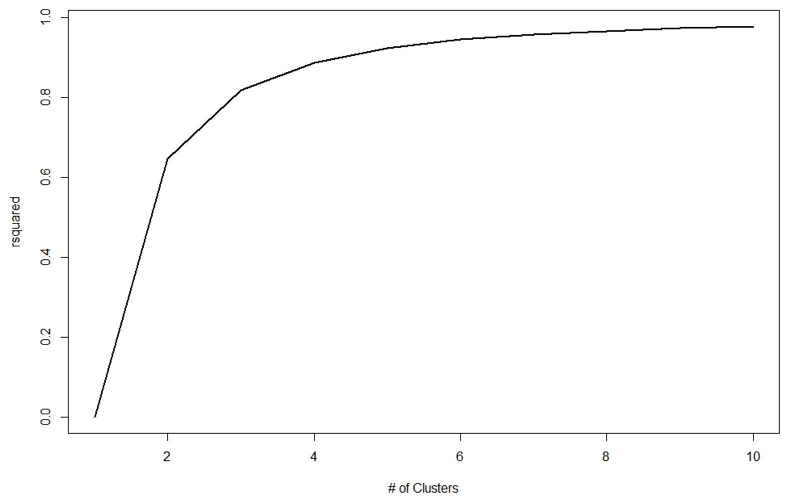
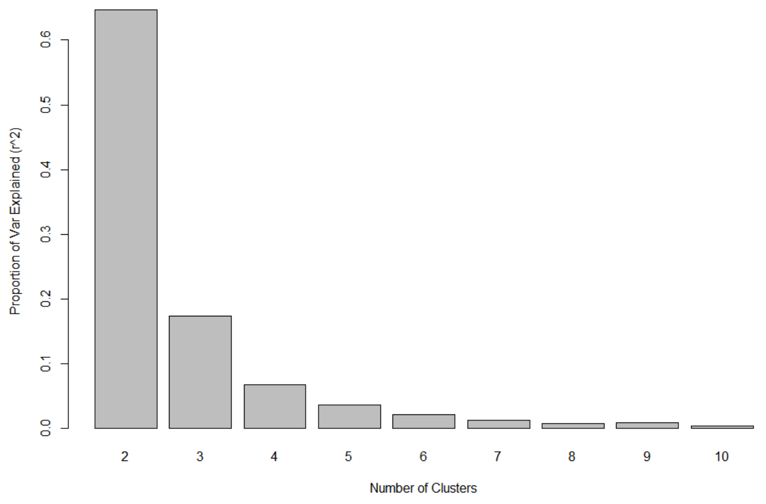


Figure 17: Accumulative RSquared from Kmeans Analysis on WaterData



Code:

```
## Reading the data in
WaterDat <- read.csv('water_potability.csv', header=TRUE)
WaterDat <- na.omit(WaterDat)
## Pulling true clustering
TrueClustering <- WaterDat[, length(WaterDat)]
## Removing Labels from the data/
```

```
WaterDat <- WaterDat[, -length(WaterDat)]

##Generating Plots
rsquared <- rep(NA, 10)
for (i in 1:10){
  tmp <- kmeans(WaterDat, centers = i)
  rsquared[i] <- tmp$betweenss/tmp$totss
}
plot(rsquared, type="l", lwd=2, xlab = '# of Clusters')
barplot((rsquared[2:10] - rsquared[1:9]), names.arg = seq(2,10),
        ylab = 'Proportion of Var Explained (r^2)',
        xlab = 'Number of Clusters' )

## Computing Classification rate
WaterKmeansClust <- kmeans(WaterDat, centers = 2)
WaterKmeansClust <- WaterKmeansClust$cluster - 1

kClassification <- WaterKmeansClust == TrueClustering
length(kClassification[kClassification == TRUE])
length(kClassification)
## 1062/2011 classified correctly.
```

Things don't get much better when we try the agglomerative hierarchical clustering. Generating the dendrogram for this data is unhelpful (too much data), it shows the same thing as the kmeans plots; that two clusters explain a majority of the variance. With the hierarchical clustering we correctly classified 1188 of the 2011 observations, again about as good as random.

Code:

```
D <- dist(WaterDat)
AgglomerativeClust <- hclust(D, method = 'average')
AgglomerativeClust <- cutree(AgglomerativeClust, k = 2)
AgglomerativeClust <- AgglomerativeClust - 1

aClassification <- AgglomerativeClust == TrueClustering
length(aClassification[aClassification == TRUE])
length(aClassification)
## 1188/2011 classified correctly
```