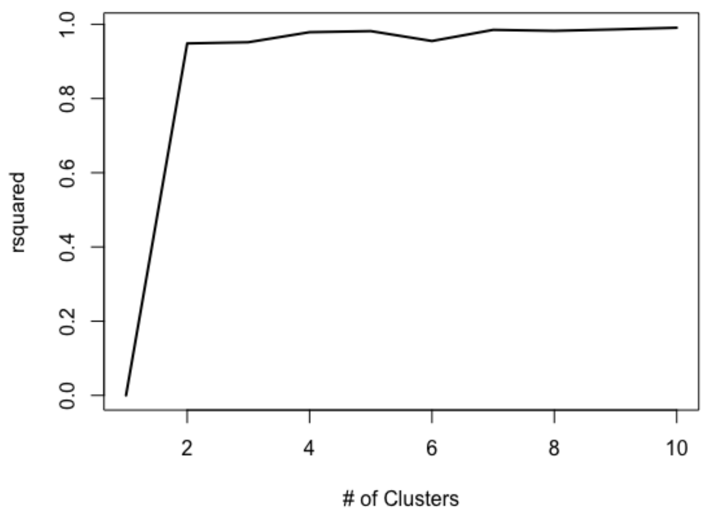
Exercise 1: Read the data in the appendix into R. You will examine the data in 3d:

```
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

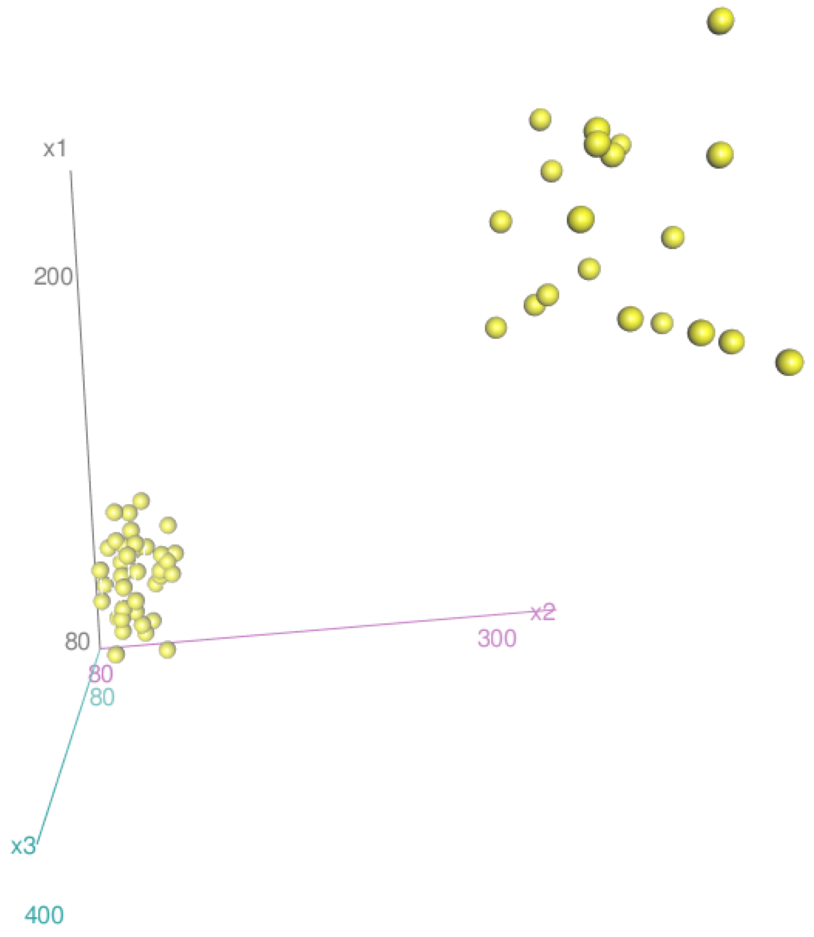
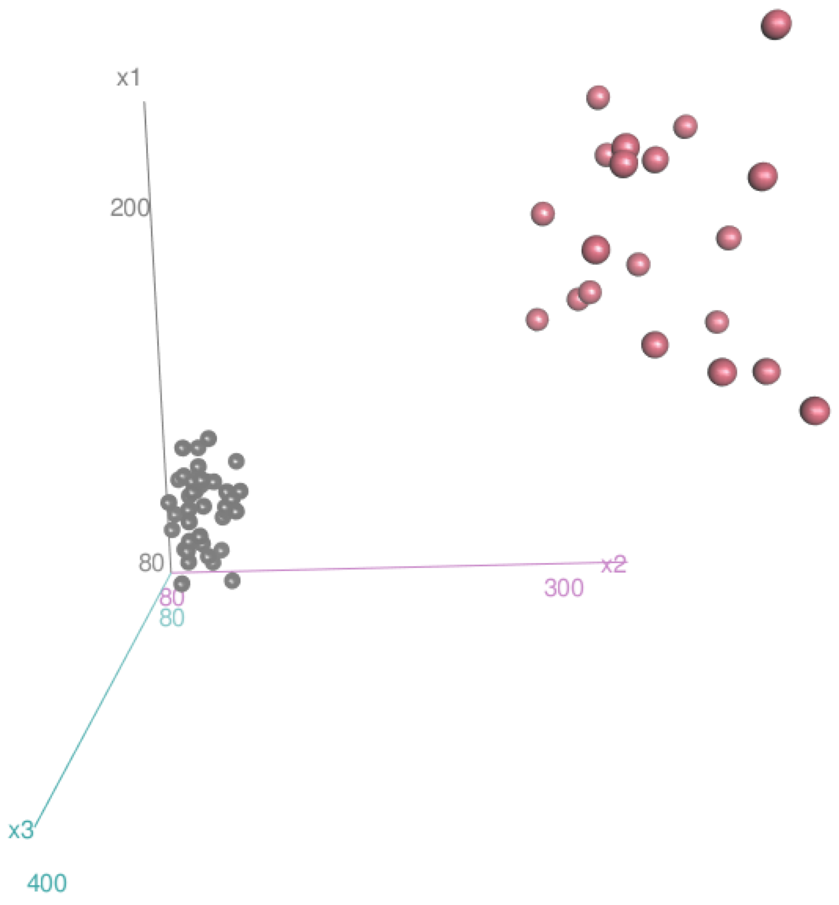


Figure 3: 3D Data Plot With 2 Clusters



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

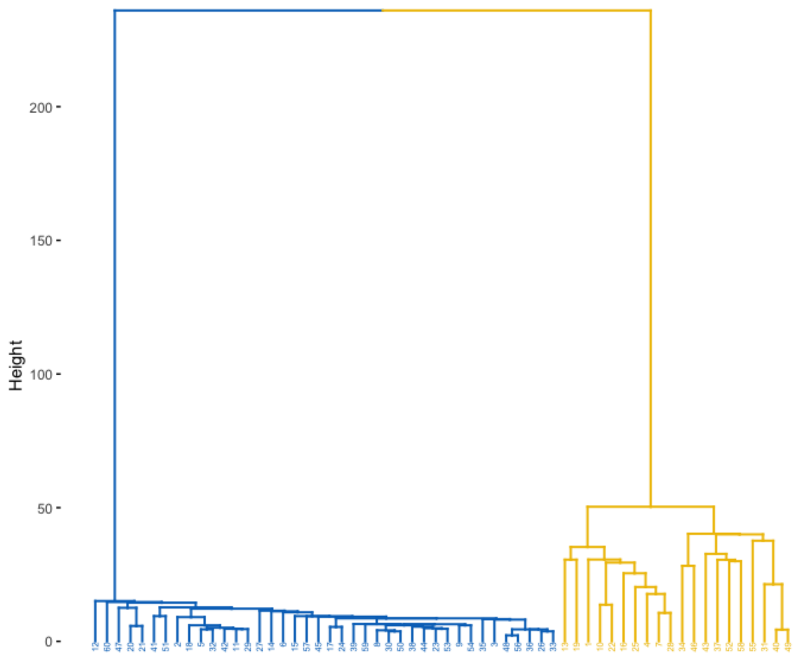
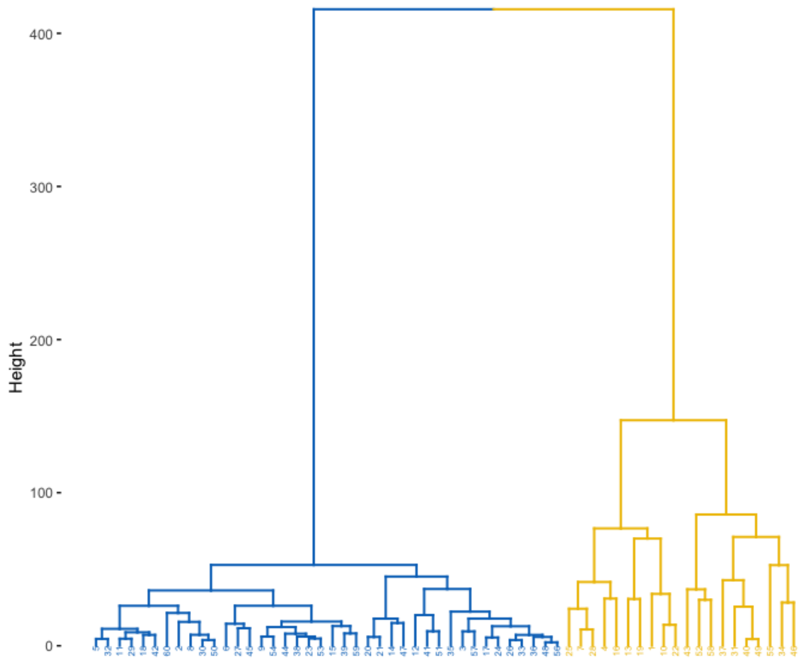
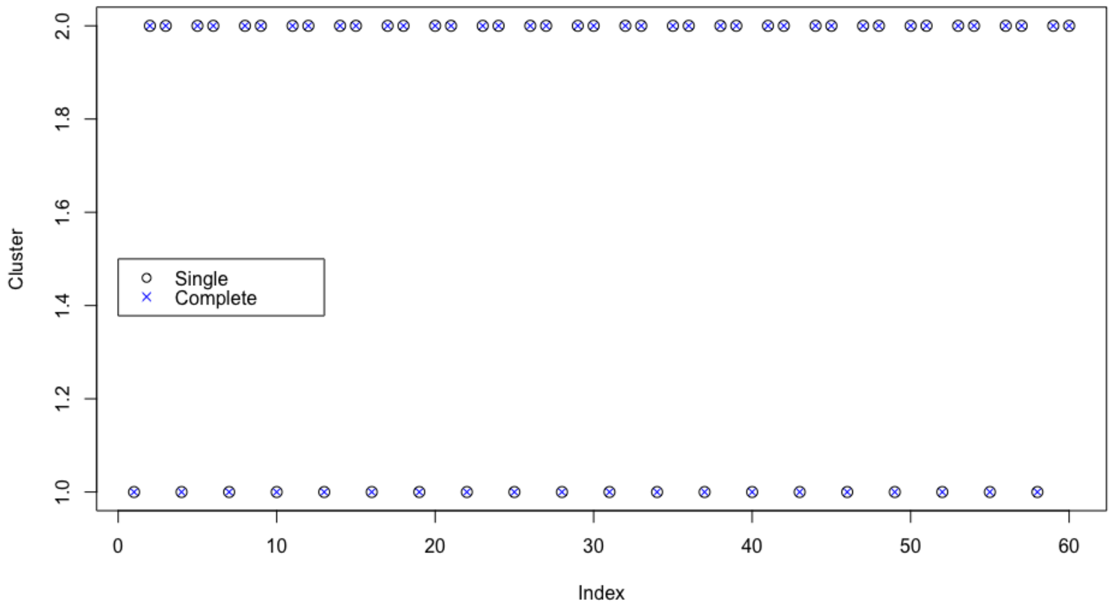


Figure 5: Complete Linkage



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 four groups
           palette = "jco", # Color palette
)
plot(cluster1)

cluster2 <- fviz_dend(tmp2, cex = 0.4,
                      k = 2, # Cut in four 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.

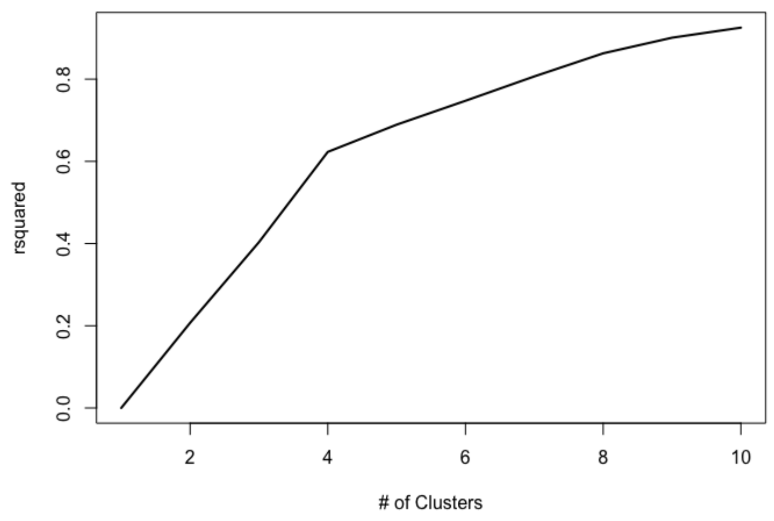
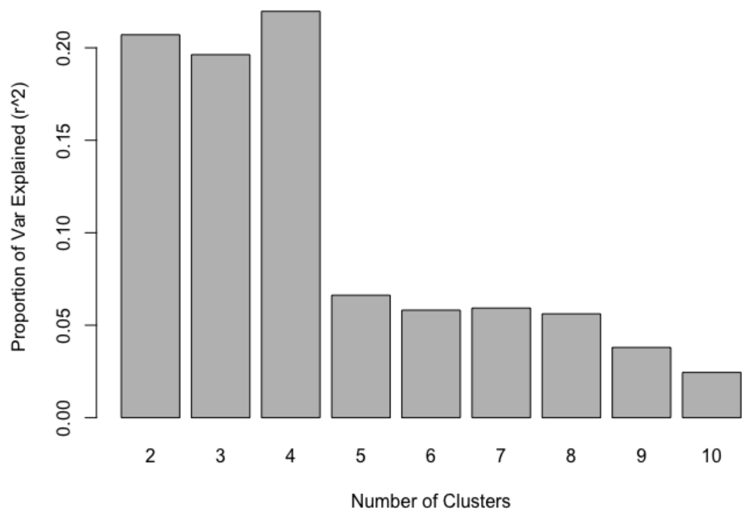


Figure 8: Proportion of r^2 for Each Cluster.(Scree-ish plot)



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, 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 centoid of each cluster is computed.