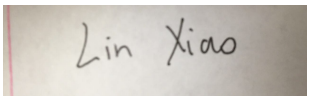


XIAO_Lin_Solutions_hw4

Lin Xiao

September 23, 2015

Problem 1
Download the file "hw1prob1.Rdata" from the course website, and load it into your R session using load("hw1prob1.Rdata"). Now you will have a document- term matrix dtm loaded into memory, of dimension 8 documents 6820 words. These are the 8 wikipedia documents that were used in lecture 2. 4 of them on the teenage mutant ninja turtles and 4 of them on the artists of the same name. If you're curious, type rownames(dtm) and colnames(dtm) (don't worry if the second command gives you a warning).

(a).
First normalize the documents by their total word count, and then apply IDF weighting to the words. Save this matrix as dtm1. Now reverse the order: first apply IDF weighting to the words, then normalize the documents by their total word count, and save this matrix as dtm2. Are dtm1 and dtm2 different? Explain why you think they should or shouldn't be different. (Note: your normalization and IDF weighting must be computed manually, i.e., you can't use functions in the tm package.)

```
load("/Users/LinXIAO/Downloads/hw1prob1.Rdata")
D<-8  # number of documents
nw<-colSums(dtm!=0) #returns a vector with the number of documents which contains the word
idf<-log(D/nw)
#normalizing and IDF weighting
dtm1<-t(t(dtm/rowSums(dtm))*idf)
#transposing here is for the multiplying calculation
#IDF weighting and normalizing
dtm2<-t(t(dtm)*idf)/rowSums(t(t(dtm)*idf))
identical(dtm1, dtm2)
```

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

They are different because in dtm2 we normalize after IDF weighting, which leads to the change of row sums. The row sums are the word counts with IDF weighting, not word counts any more.

(b).
We're going to forgo IDF weighting with such a small collection (8 documents). Normalize the documents by their total word count, and call this matrix dtm3. According to this document-term matrix, which document is closest (measured in Euclidean distance) to the document named "tmnt mike"?

```
dtm3<-dtm/rowSums(dtm)
which(rownames(dtm)=="tmnt mike") #return the row index
```

```
## [1] 3
```

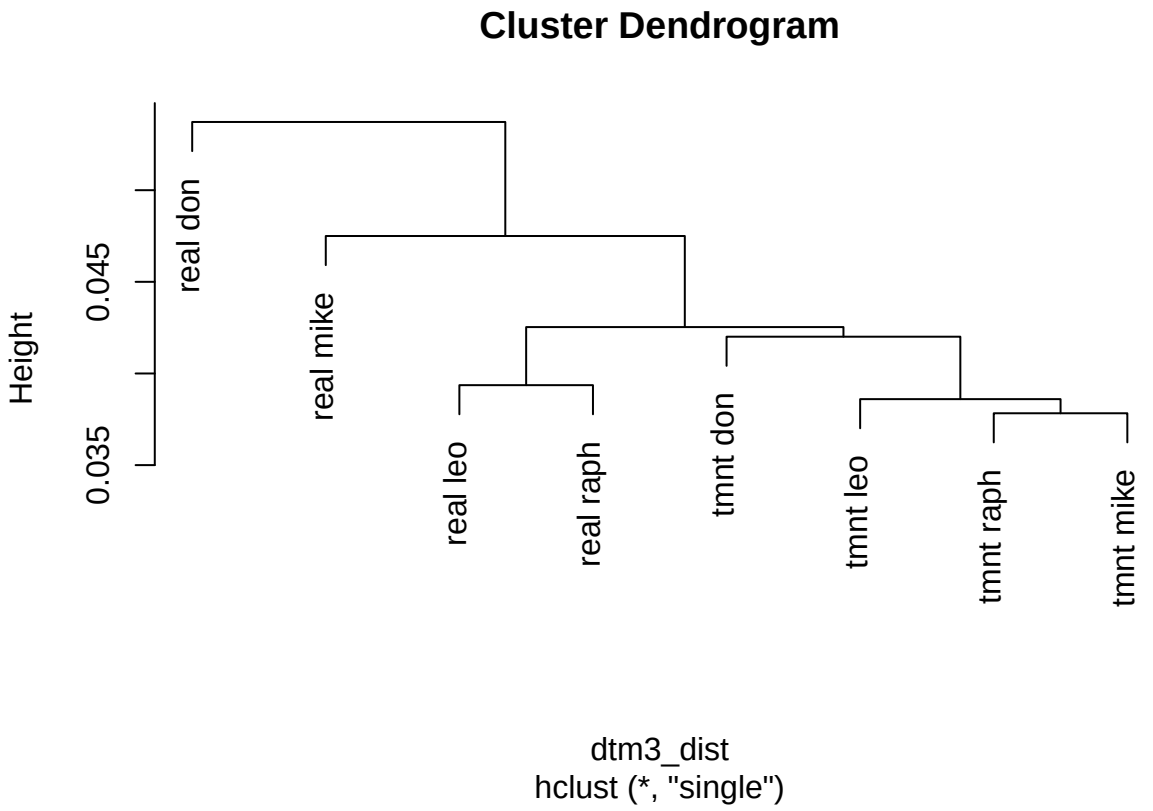
```
#apply the l2norm to the distances to "tmnt mike"; length 7
dist.l2 = sqrt(rowSums((scale(dtm3[-which(rownames(dtm)=="tmnt mike"),],
                              center=dtm3["tmnt mike",],scale=F)^2)))
which(dist.l2==min(dist.l2)) #return the index of closest doc
```

```
## tmnt raph
##         2
```

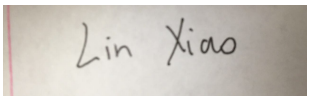
Lin Xiao

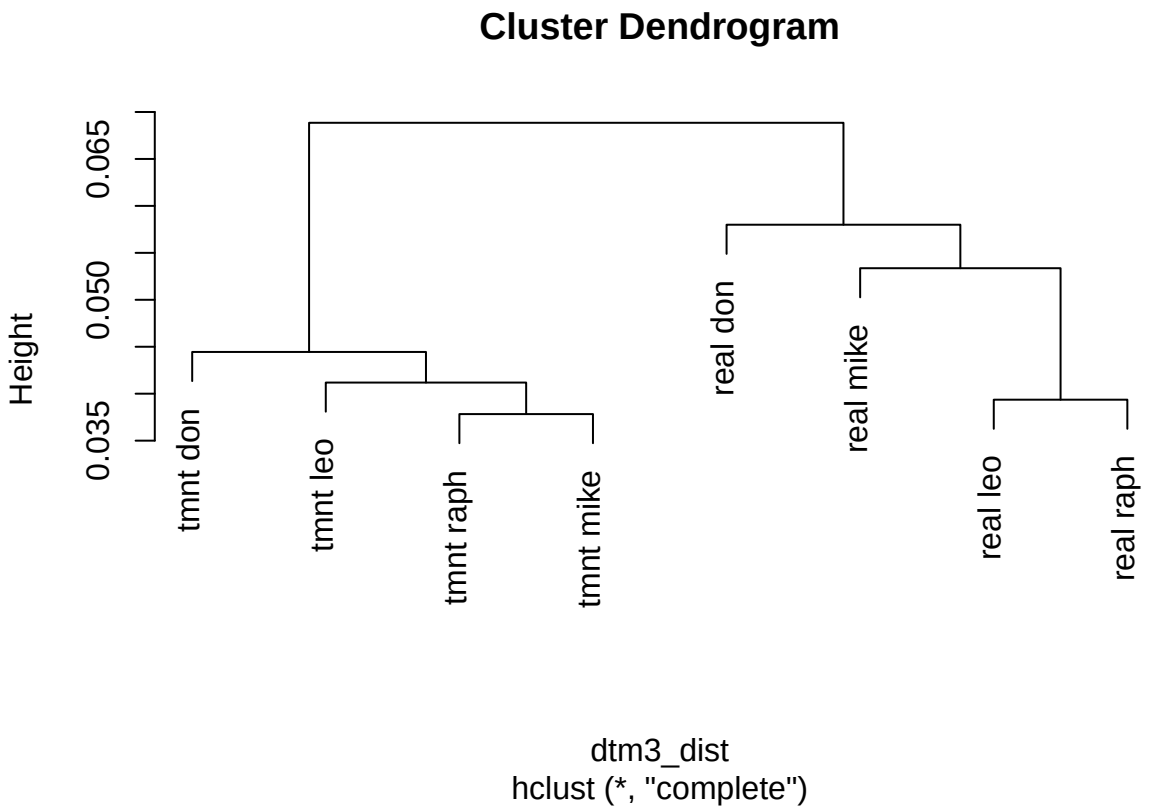
(c).
Sticking with the normalized matrix dtm3, compute the distance between each pair of documents using the function dist. Now run hierarchical agglomerative clustering, both with single linkage and complete linkage, using the function hclust. Plot the resulting dendograms for both linkages. If you had to split the documents into 2 groups, which linkage do you think gives a more reasonable clustering?

```
dtm3_dist<-dist(dtm3)
plot(hclust(dtm3_dist,method = "single"))
```




```
plot(hclust(dtm3_dist,method = "complete"))
```





Complete linkage gives a more reasonable clustering.

(d).
Combine the word counts from all of the documents into one cumulative word count vector. I.e., for each word, you should now have a count for the number of times it appears across all 8 documents. List the top 20 most common words, and how many times they appear. What percentage of total occurrences do these top 20 words represent? How many of the top words account for 50% of total occurrences?

```
single_word_count<-colSums(dtm) #word counts for each words
top<-sort(single_word_count,decreasing = T)
top[1:20]   #return the top20 and their number of appearances
```

```
##          the          and          his          was     leonardo
##         2664         1127          636          453          342
##         that          for         with michelangelo      raphael
##          297          282          279          277          212
##         from         this        which      turtles    donatello
##          209          189          129          127          117
##          him       series         were          who          one
##          116          115          111          110          103
```

```
sum(top[1:20])/sum(single_word_count)    #Percentage
```

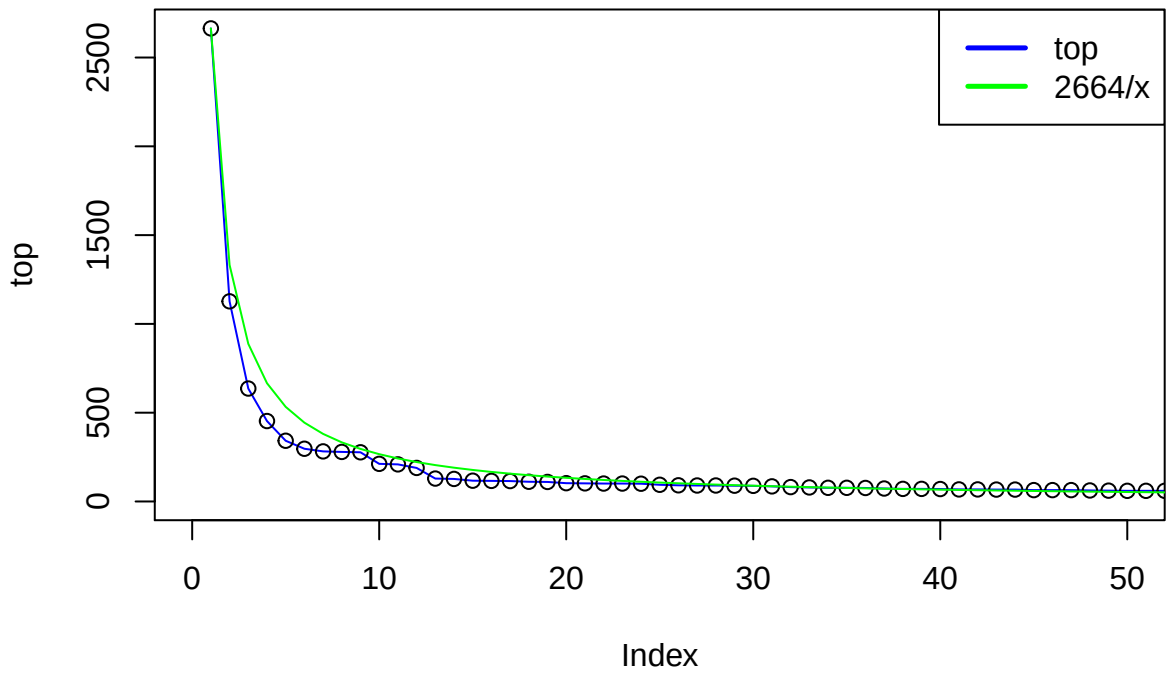
```
## [1] 0.2434249
```

```
for(i in 20:length(single_word_count)){
  if(sum(top[1:i])/sum(single_word_count)>0.5){
    print(i)
    break
  }
}# print the number of top words account for 50% of total occurrences
```

```
## [1] 254
```

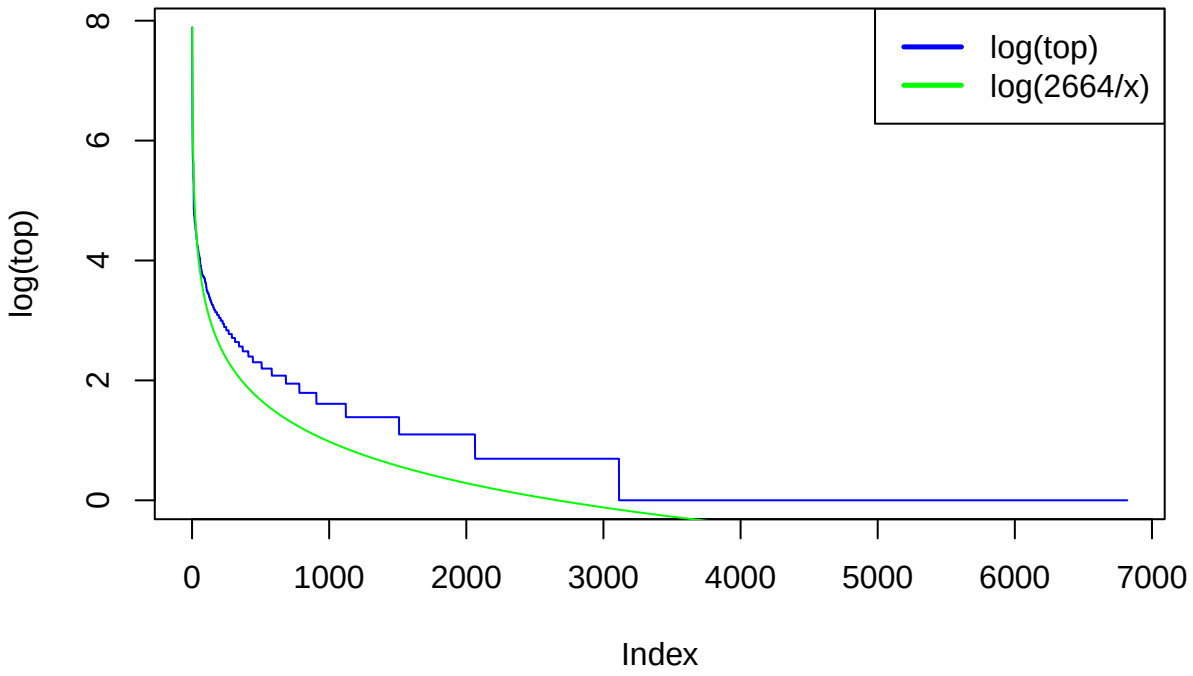
(e).
Zipf's law states that, given a collection of documents like the one we have, the number times a word appears is inversely proportional to its rank (the words being ranked by how common they are). In other words, the second most common word appears half as often as the most common word, the third most common word appears a third as often as the most common word, etc. This "law" is one that has been empirically confirmed, and aside from word counts, these kind of "power" laws have also been observed in a wide variety of different problems.1 Does our collection of 8 wikipedia articles appear to follow Zipf's law? Can you give a plot to provide visual evidence for or against this claim? (Hint: for your plot, think about translating the relationship expressed by Zipf's law into a mathematical one, between the number of occurrences y and the rank x of a word. Now take logs.)

```
par(mfrow=c(1,1))
plot(top,xlim=c(0,50),type="l",col="blue")
points(top)
y<-c(top[1])
for(i in 2:6820){
  y[i]<-y[1]/i
}
lines(y,col="green")
legend("topright",c("top","2664/x"),lty=c(1,1), lwd=c(2.5,2.5),col=c("blue","green"))
```



Lin Xiao

```
plot(log(top),col="blue",type="l")
lines(log(y),col="green")
legend("topright",c("log(top)","log(2664/x)"),lty=c(1,1), lwd=c(2.5,2.5),col=c("blue","green"))
```



Problem 2 Compute the PageRank vector for the following graph, with d = 0.85. Repeat the calculation for d = 1(BrokenRank). What's the difference? Explain.
You should be computing PageRank by explicitly constructing A and finding its leading eigenvector (i.e., you can?t use the page.rank function in the igraph package). (Hint: you don?t have to compute the leading eigenvector by repeatedly multiplying by A, although you can if you really want to. Instead, use the eigen function in R.)

```
graph<-matrix(c(0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,1,0,0,0,0,0,0,0,0,
0,0,1,0,1,1,0,1,0,1,
0,0,0,0,0,0,1,0,0,0,
0,1,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,1,0,1,0,
0,0,0,1,0,0,0,0,0,0,
1,0,1,0,1,0,1,0,0,0),ncol=10,byrow = T)
n<-ncol(graph) #count the number of webpages
E<-matrix(rep(1,n*n),ncol=n)
M<-diag(apply(graph,2,sum))
MInv<-solve(M)
RANK<-function(d){
   PRmatrix<-(1-d)/n*E + d*graph %*% MInv
   #get corresponding eigenvector(s) with eigenvalue 1
   PR<-eigen(PRmatrix)$vectors[,(which(signif(eigen(PRmatrix)$values,4)==1))]
   #normalize the Page Rank vectors by sum of columns
   normalizedPR<-scale(PR,center=F,scale=as.numeric(apply(as.matrix(PR),2,sum)))
   return(normalizedPR)
```

```
}
RANK(1)      #brokenrank
```

```
##               [,1]
##  [1,] 0.0000000+0i
##  [2,] 0.0000000+0i
##  [3,] 0.0000000+0i
##  [4,] 0.3333333+0i
##  [5,] 0.0000000+0i
##  [6,] 0.0000000+0i
##  [7,] 0.0000000+0i
##  [8,] 0.3333333+0i
##  [9,] 0.3333333+0i
## [10,] 0.0000000+0i
## attr(,"scaled:scale")
## [1] -1.732051
```

```
RANK(0.85)   #pagerank
```

```
##               [,1]
##  [1,] 0.01500000+0i
##  [2,] 0.01500000+0i
##  [3,] 0.02137500+0i
##  [4,] 0.30971210+0i
##  [5,] 0.01925000+0i
##  [6,] 0.02137500+0i
##  [7,] 0.01500000+0i
##  [8,] 0.25576699+0i
##  [9,] 0.27825528+0i
## [10,] 0.04926563+0i
## attr(,"scaled:scale")
## [1] -2.027973
```

The difference is that some components of the BorkenPage vector are 0. It is because there is a loop in the graph, among vertices 4,8 and 9. Setting d=0.85 can fix this problem.

Problem 3

You're going to prove some claims from lecture 4.

(a).

Let $X_1, ..., X_p \in R^p$, and C be a function assigning points to clusters 1,...,K. Let n_k be the number of points assigned to the kth cluster. Prove that the within-cluster scatter here is exactly the within-cluster variation:

$$\frac{1}{2}\sum_{k=1}^{K}\frac{1}{n_k}\sum_{C(i)=k}\sum_{C(j)=k}||X_i - X_j||_2{}^2 = \sum_{k=1}^{K}\sum_{C(i)=k}||X_i - \overline{X}_k||_2^2$$

where $\overline{X}_k$ is the average of the points in group k. That is, $\overline{X}_k = \frac{1}{n_k}\sum_{C(i)=k} X_i$.

(here I mean vector addition). Hint 1: don't let the vectors and norms scare you! Prove the result for scalars first, and then see what you can do. Hint 2: don't let the multiple sums scare you! Get rid of the outer sum taken over 1,...,K and consider a fixed group k.

Proof:

Consider a fixed group k, $X_i = (X_{i1}, ..., X_{ip})$, $X_j = (X_{j1}, ..., X_{jp})$, $\overline{X}_k = (\overline{X}_{k1}, ..., \overline{X}_{kp})$

$$\begin{aligned}
\frac{1}{2}\frac{1}{n_k}\sum_{C(i)=k}\sum_{C(j)=k}||X_i - X_j||_2{}^2 &= \frac{1}{2n_k}\sum_{C(i)=k}\sum_{C(j)=k}\sum_{m=1}^{p}(X_{im}-X_{jm})^2 \\
&= \sum_{m=1}^{p}\frac{1}{2n_k}\sum_{i=1}^{n_k}\sum_{j=1}^{n_k}(X_{im}^2+X_{jm}^2-2X_{im}X_{jm}) \\
&= \sum_{m=1}^{p}(\frac{1}{2}\sum_{i=1}^{n_k}X_{im}^2+\frac{1}{2}X_{jm}^2-n_k\overline{X}_{km}{}^2) \\
&= \sum_{m=1}^{p}(\sum_{i=1}^{n_k}X_{im}^2+n_k\overline{X}_{km}-2n_k\overline{X}_{km}{}^2) \\
&= \sum_{m=1}^{p}(\sum_{i=1}^{n_k}X_{im}^2+\sum_{i=1}^{n_k}\overline{X}_{km}-2\sum_{i=1}^{n_k}X_{im}\overline{X}_{km}) \\
&= \sum_{m=1}^{p}\sum_{C(i)=k}(X_{im}-\overline{X}_{km})^2 \\
&= \sum_{C(i)=k}||X_i-\overline{X}_k||_2^2
\end{aligned}$$

Therefore

$$\frac{1}{2}\sum_{k=1}^{K}\frac{1}{n_k}\sum_{C(i)=k}\sum_{C(j)=k}||X_i - X_j||_2{}^2 = \sum_{k=1}^{K}\sum_{C(i)=k}||X_i-\overline{X}_k||_2^2$$

(b).
Let $Z_1, ..., Z_m \in R^p$. Prove

$$\sum_{i=1}^{m}||Z_j - c||_2^2$$

is minimized by taking $c = \overline{Z}$, the average of the points, i.e.$\overline{Z} = \frac{1}{m}\sum_i Z_i$. (again this is vector addition)

Proof:
$Z_i = (Z_{i1}, ..., Z_{ip})$, $c = (c_1, ...c_p)$

$$\begin{aligned}
\sum_{i=1}^{m}||Z_j - c||_2^2 &= \sum_{i=1}^{m}\sum_{k=1}^{p}(Z_{ik}-c_k)^2 \\
&= \sum_{k=1}^{p}(\sum_{i=1}^{m}Z_{ik}^2-2\sum_{i=1}^{m}Z_{ik}c_k+\sum_{i=1}^{m}c_k^2) \\
&= \sum_{k=1}^{p}(mc_k^2-2m\overline{Z}_k+\sum_{i=1}^{m}Z_{ik}^2) \\
&= \sum_{k=1}^{p}m[(c_k-\overline{Z}_k)^2+\sum_{i=1}^{m}Z_{ik}^2-\overline{Z}_k{}^2]
\end{aligned}$$

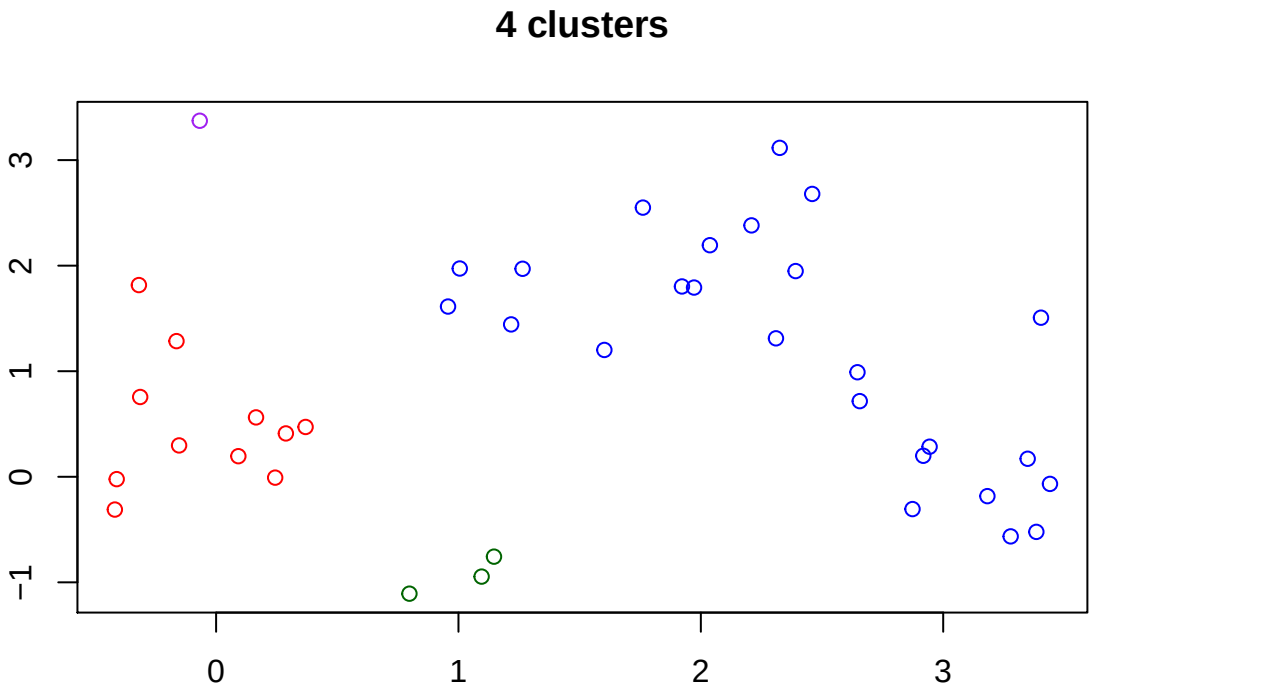
So in order to get the minimum we take $c_k = \overline{Z}_k$, i.e. $c = \overline{Z}$

(c).
Let W_t denote the within-cluster variation at the start of iteration t of K-mean clustering. Prove that for any t, $W_{t+1} \leq W_t$

With the conclusion above, we can prove this from two steps:
1.Minimize over C: For each k=1,...K, find the cluster center c_k closest to X_i, then for all i, $(X_i - c_k)^2$ should have the smallest value, so we are able to reduce W;
2.Minimize over $c_1, ...c_K$: We get the minimum of $\sum_{C(i)=k} ||X_i - c_k||_2^2$ by taking the average of the points, that is, $c_k = \overline{X}_k$(from part b's conclusion)
Every step is reducing(or not raising) W compared to the previous step. Therefore, after an iteration, we have $W_{t+1} \leq W_t$

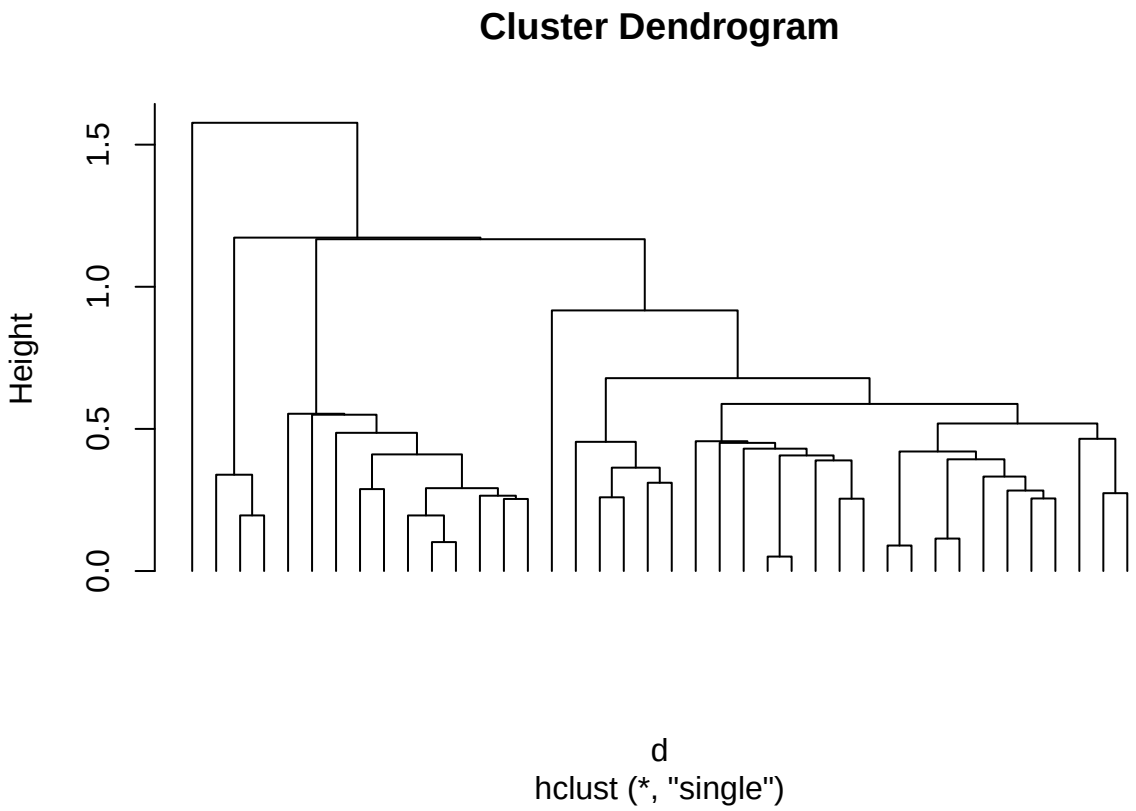
Problem 4
In this problem you're going to investigate the invariance of agglomerative clustering using either single or complete linkage under a monotone transformation of the distances. Download the file "hw1prob3.Rdata" from the course website and load it into your R session using load("hw1prob3.Rdata"). Now you'll have two objects loaded into memory:
• **x**, a 40×2 matrix containing 40 observations along its rows;
• **d**, the pairwise Euclidean distances.
(a).
Run hierarchical agglomerative clustering with single linkage, using the function hclust. Cut the tree at K = 4 clusters using the function cutree, which returns a vector of cluster assignments. Plot the points in x with different colors (or different pch values) indicating the cluster assignments. Also plot the dendogram. (Note: if you want your dendograms to look like the ones in lecture, choose a really small negative value for the hang parameter, e.g., hang=-1e-10).

```
load("/Users/LinXIAO/Downloads/hw1prob3.Rdata")
hc_d_single<-hclust(d,method="single")    #single clustering for d
d.single = cutree(hc_d_single,k=4)        #cut tree with K=4
cols = c("red","darkgreen","blue","purple")
plot(x,col=cols[d.single],main="4 clusters",xlab="",ylab="")   #clustering assignments
```



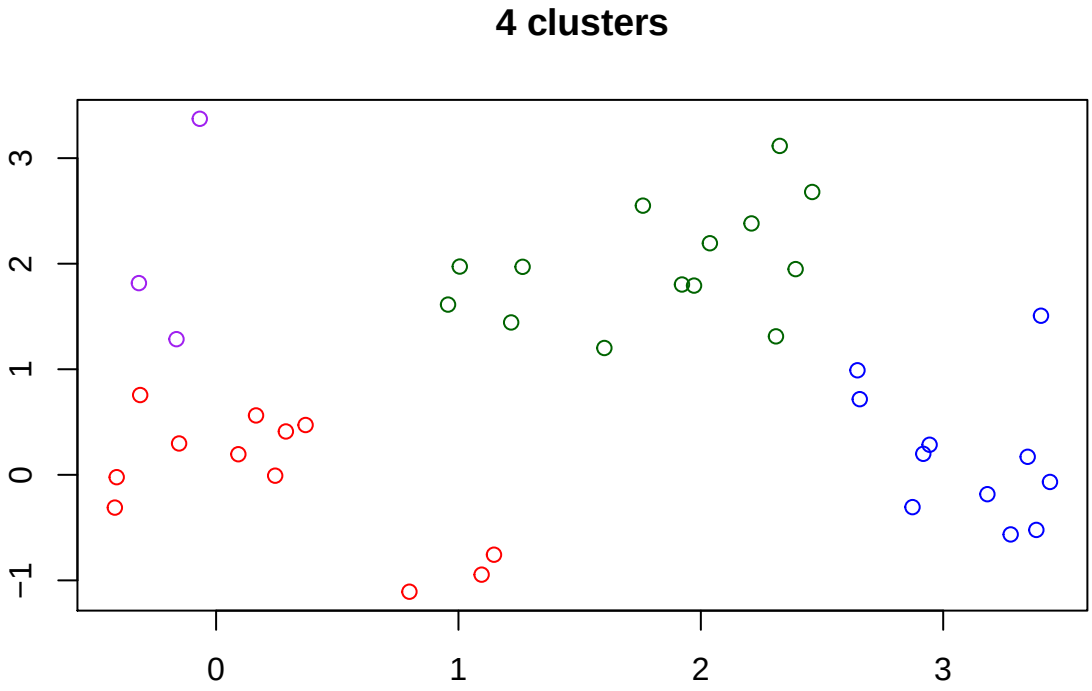

```
plot(hc_d_single,labels=F,hang=-1e-10)  #plot dendogram  #dendogram
```

Lin Xiao

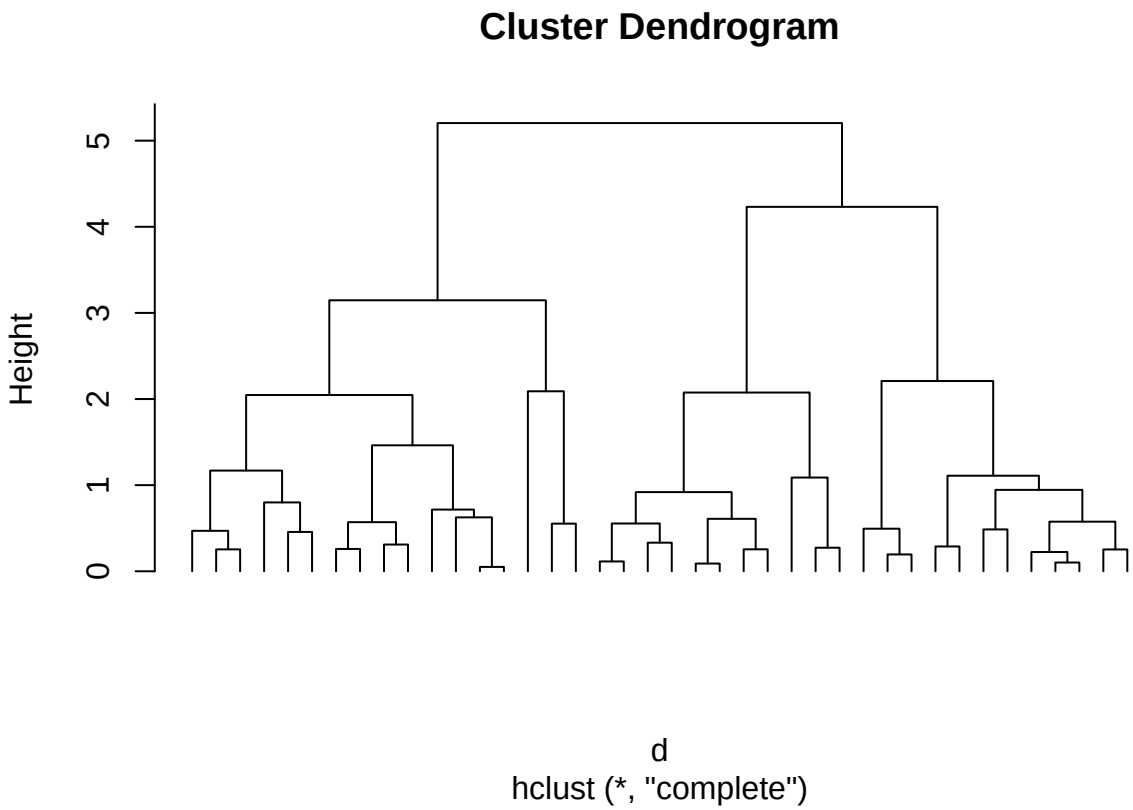


(b).
Repeat part (a) for complete linkage.

```
hc_d_complete<-hclust(d,method="complete")  #complete clustering for d
d.complete = cutree(hc_d_complete,k=4)   #cut tree with K=4
cols = c("red","darkgreen","blue","purple")
plot(x,col=cols[d.complete],main="4 clusters",xlab="",ylab="")
```

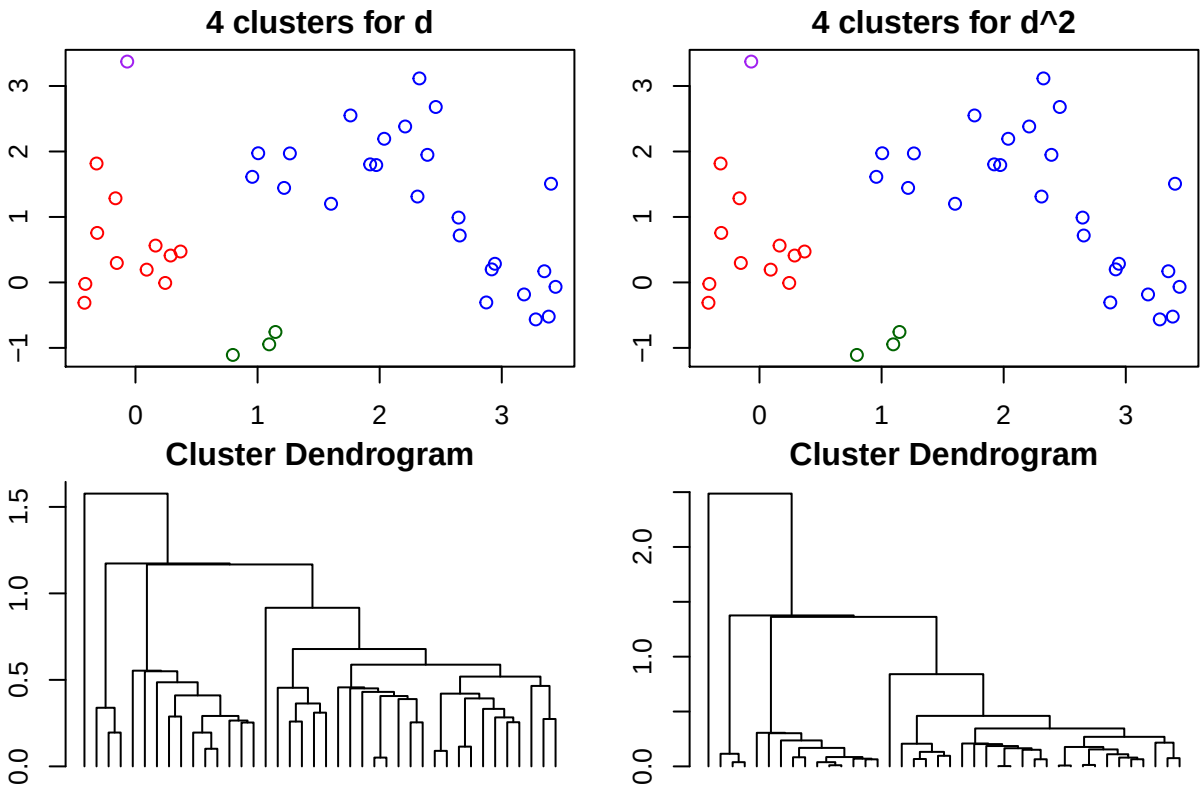



```
plot(hc_d_complete,labels=F,hang=-1e-10)
```



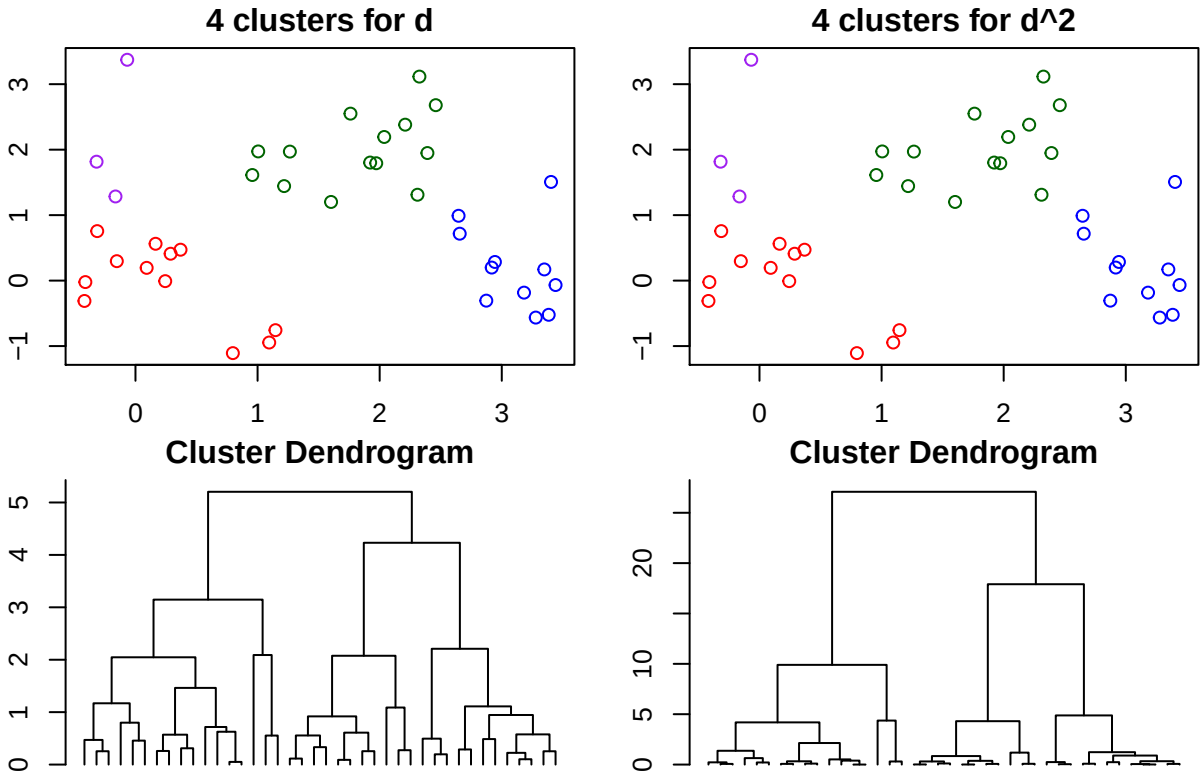
(c).
Repeat parts (a) and (b), but passing d^2 to the function hclust instead of d. Did the clustering assignments change? Did the dendograms change?

```
hc_d2_single<-hclust(d^2,method="single")  #single clustering for d^2
d2.single = cutree(hc_d2_single,k=4)
cols = c("red","darkgreen","blue","purple")

par(mfrow=c(2,2),mai=c(0.3,0.3,0.3,0.3))
plot(x,col=cols[d.single],main="4 clusters for d",xlab="",ylab="")
plot(x,col=cols[d2.single],main="4 clusters for d^2",xlab="",ylab="")
plot(hc_d_single,labels=F,hang=-1e-10)
plot(hc_d2_single,labels=F,hang=-1e-10)
```




```
hc_d2_complete<-hclust(d^2,method="complete")  #complete clustering for d^2
d2.complete = cutree(hc_d2_complete,k=4)
cols = c("red","darkgreen","blue","purple")

par(mfrow=c(2,2))
plot(x,col=cols[d.complete],main="4 clusters for d",xlab="",ylab="")
plot(x,col=cols[d2.complete],main="4 clusters for d^2",xlab="",ylab="")
plot(hc_d_complete,labels=F,hang=-1e-10)
plot(hc_d2_complete,labels=F,hang=-1e-10)
```

Lin Xiao



No change to the clustering assignments but the height of dendogram for d^2 is doubled

(d).
Prove that for single linkage, running agglomerative clustering with dissimilarities d_{ij} and running it again with dissimilarities $h(d_{ij})$ produces the same sequence of clustering assignments, provided that h is a monotone increasing function. Recall that a monotone increasing function h is one such that $x \leq x^{'}$ implies $h(x) \leq h(x^{'})$. Prove the same thing for complete linkage.

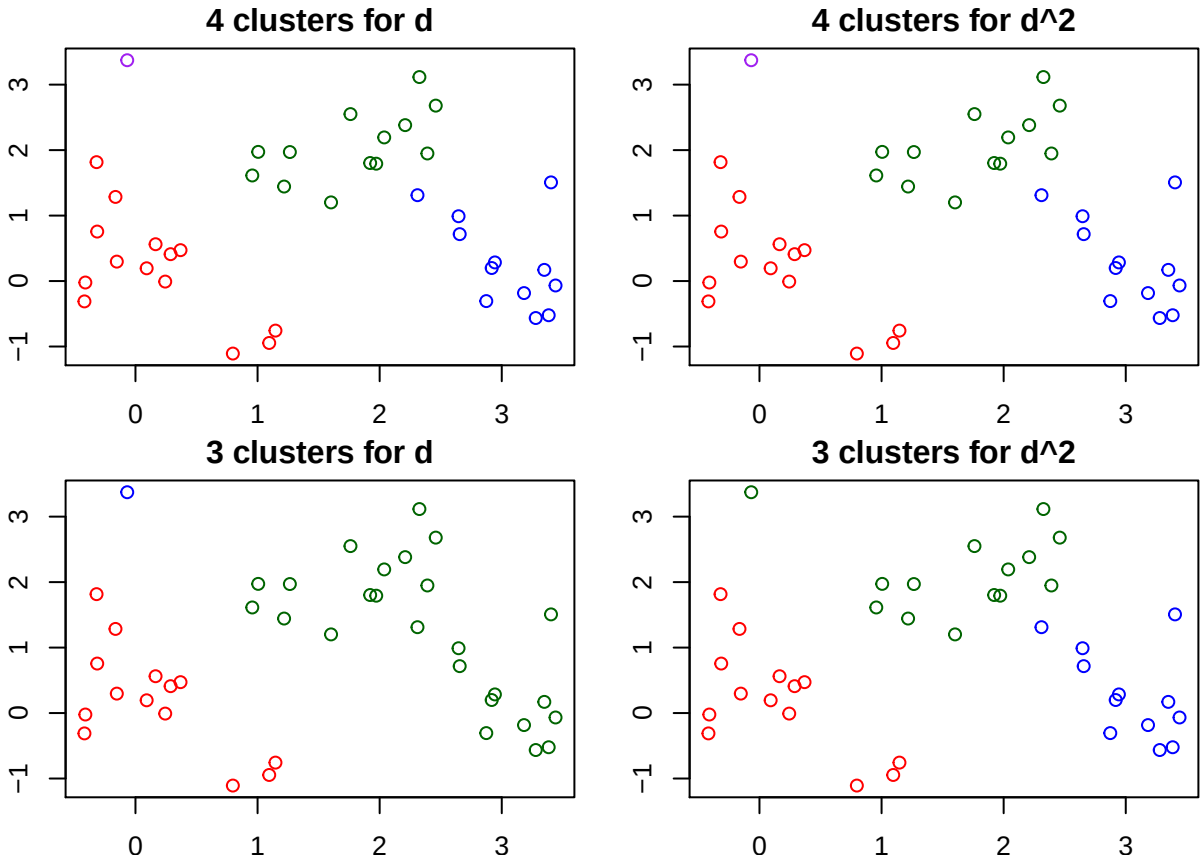
Proof:
1.For any i_0, i_1, j_0, j_1 belong to two separate groups, if $d_{i_0,j_0} \leq d_{i_1,j_1}$, then for monotone increasing function we have $h(d_{i_0,j_0}) \leq h(d_{i_1,j_1})$
2.Within-cluster:The i,j that satisfied $min(d_{ij})$ or $max(d_{ij})$ also satisfy $min(h(d_{ij}))$ and $max(h(d_{ij}))$, for any 2 groups.
3.Between-clusters:For any two groups,we have a sequence to decide how to cluster:$d_{i_1j_1} \leq d_{i_2j_2} \leq \cdots \leq d_{i_nj_n}$, with point 1, we have $h(d_{i_1j_1}) \leq h(d_{i_2j_2}) \leq \cdots \leq h(d_{i_nj_n})$
With the above 3 points we prove that for single and complete linkages, $h(d_{ij})$ produces the same sequence of clustering assignments.

(e).
Run agglomerative clustering with average linkage on each of d and d^2. Cut both trees at K = 4. Are the clustering assignments the same? How about for K = 3? (You should produce plots of x, like the ones you made above, with different colors or pch values in order to display the clustering results.)

```
#Average when K=4
hc_d_average_col4<-hclust(d,method="average")
d_average_col4 = cutree(hc_d_average_col4,k=4)
hc_d2_average_col4<-hclust(d^2,method="average")
d2_average_col4 = cutree(hc_d2_average_col4,k=4)

#Average when K=3
hc_d_average_col3<-hclust(d,method="average")
```

Lin Xiao

```
d_average_col3 = cutree(hc_d_average_col3,k=3)
hc_d2_average_col3<-hclust(d^2,method="average")
d2_average_col3 = cutree(hc_d2_average_col3,k=3)


cols3 = c("red","darkgreen","blue")
cols4 = c("red","darkgreen","blue","purple")
par(mfrow=c(2,2),mai=c(0.3,0.3,0.3,0.3))
plot(x,col=cols4[d_average_col4],main="4 clusters for d",xlab="",ylab="")
plot(x,col=cols4[d2_average_col4],main="4 clusters for d^2",xlab="",ylab="")
plot(x,col=cols3[d_average_col3],main="3 clusters for d",xlab="",ylab="")
plot(x,col=cols3[d2_average_col3],main="3 clusters for d^2",xlab="",ylab="")
```



Clustering assignments are the same for $dandd^2$ when $K = 4$, but there are differences between $dandd^2$ when K=3