S&DS 363 Homework 5

Evan Collins, Kelly Farley, Ken Stier

15 April 2021

```
## ---
## biotools version 4.0
```

Contributors

Evan Collins (evan.collins@yale.edu)

Kelly Farley (kelly.farley@yale.edu)

Ken Stier (ken.stier@yale.edu)

The Dataset

Raw dataset: COVID-19 infection and death statistics from U.S. counties (sourced from NYT), combined with economic, education, and population data (sourced from various government agencies) and also survey responses about mask-wearing frequencies (sourced from NYT). 3141 complete observations on 19 metric variables and 6 categorical variables. To avoid any outliers due to population size differences between counties, all variables are scaled as a percentage of population. Variable descriptions can be found here.

Data of relevance for this pset:

In previous psets, we had determined our favorite discriminating variables to be median household income, poverty rate, and confirmed COVID cases as a percentage of the population; we also determined that the log of each of these variables is more useful and normal than the raw variables, so we will be applying that transformation.

Should we choose to use these categorical variables for reference, the `rural_urban_code` and the `region` variables defined below tend to be great for grouping comparisons. The Rural-Urban Codes are numbered 1-9 according to descriptions provided by the USDA, and we consolidate them into: (1) "Urban" for codes 1-3, (2) "Suburban" for codes 4-6, and (3) "Rural" for codes 7-9. Region is determined for each county/state by the Census Bureau (Northeast, Midwest, South, and West).

```
raw <- readr::read_csv("https://evancollins.com/covid_and_demographics.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   X1 = col_character(),
```

```
##   County_Name = col_character(),
##   State_Name = col_character(),
##   FIPS = col_character()
## )
## i Use `spec()` for the full column specifications.
```

```
# create categorical variables: rural-urban code (3 levels), region (4 variables)
raw <-
  raw %>%
    mutate(region = case_when(
      State_Name %in% c("Washington", "Oregon", "California", "Nevada", "Idaho", "Montana", "Utah", "Ari
      State_Name %in% c("North Dakota", "South Dakota", "Nebraska", "Kansas", "Minnesota", "Iowa", "Miss
      State_Name %in% c("Texas", "Oklahoma", "Arkansas", "Louisiana", "Mississippi", "Tennessee", "Kentu
      State_Name %in% c("Pennsylvania", "New Jersey", "Connecticut", "Rhode Island", "Massachusetts", "
      rural_urban_code = case_when(
        Rural_Urban_Code_2013 %in% c(1, 2, 3) ~ "Urban",
        Rural_Urban_Code_2013 %in% c(4, 5, 6) ~ "Suburban",
        Rural_Urban_Code_2013 %in% c(7, 8, 9) ~ "Rural")
      )
raw$rural_urban_code <- as.factor(raw$rural_urban_code) # Rural is reference

# log transformations of our continuous variables
raw$logMedian_Household_Income_2019 <- log(raw$Median_Household_Income_2019 + 0.0001)
raw$logPercent_Poverty_2019 <- log(raw$Percent_Poverty_2019 + 0.0001)
raw$logCovid_Confirmed_Cases_as_pct <- log(raw$Covid_Confirmed_Cases_as_pct + 0.0001)
```

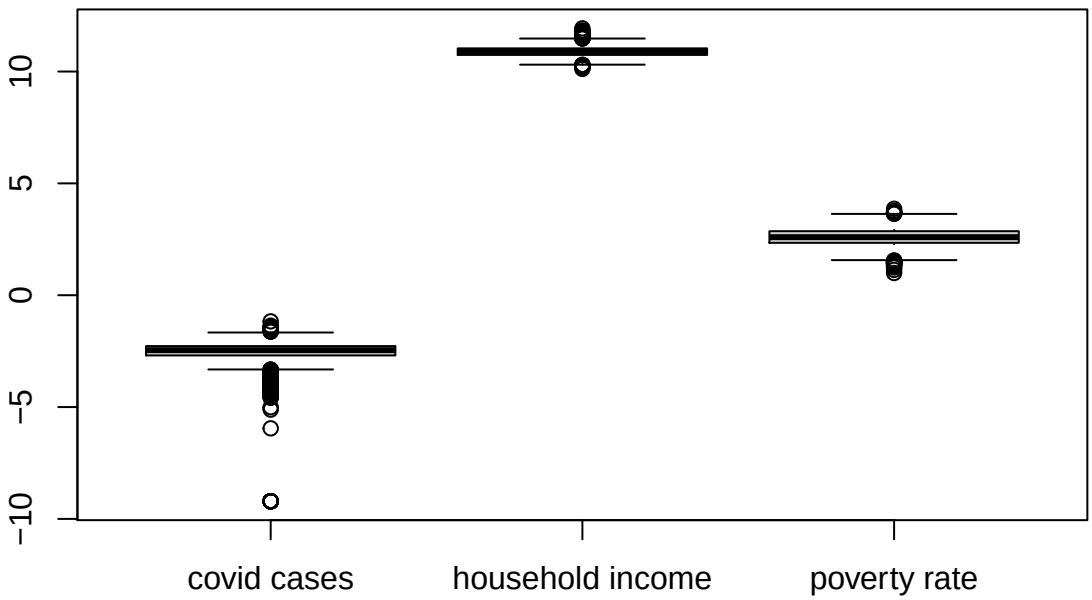
1 Distance Metrics

Think about what distance metrics are appropriate for your data based on data type. Write a few sentences about this. Also think about whether you should standardize and/or transform your data (comment as appropriate).

In our case, it probably makes the most sense to go with Euclidean distance, since we're working with a set of continuous variables without special relationships that might necessitate something like city distance (such as categorical or binary data). We might want to consider using one of the measures JDRS mentioned for ecological data (i.e., Sorenson or Jaccard), given that epidemiological data may bear similarity, but it isn't strictly called for.

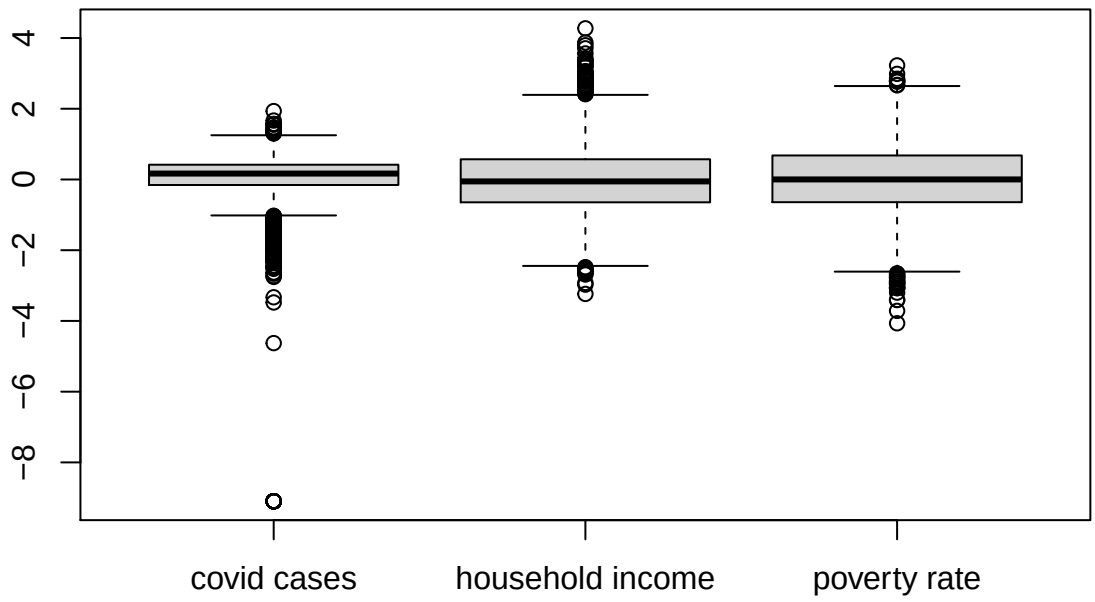
Regarding standardization, that would almost definitely be a good idea. If any variables are scaled or distributed very differently, that difference will weight the greater/broader variables over the others. We might make an exception for groups of variables on standard scales (like a five-point scale) or if the differences are based on some intrinsic property of the variables (like if they're interrelated as in our previous pset), but those cases don't apply here. Let's throw together a very quick box plot just to see how our variables look:

```
boxplot(raw$logCovid_Confirmed_Cases_as_pct, raw$logMedian_Household_Income_2019, raw$logPercent_Poverty
```



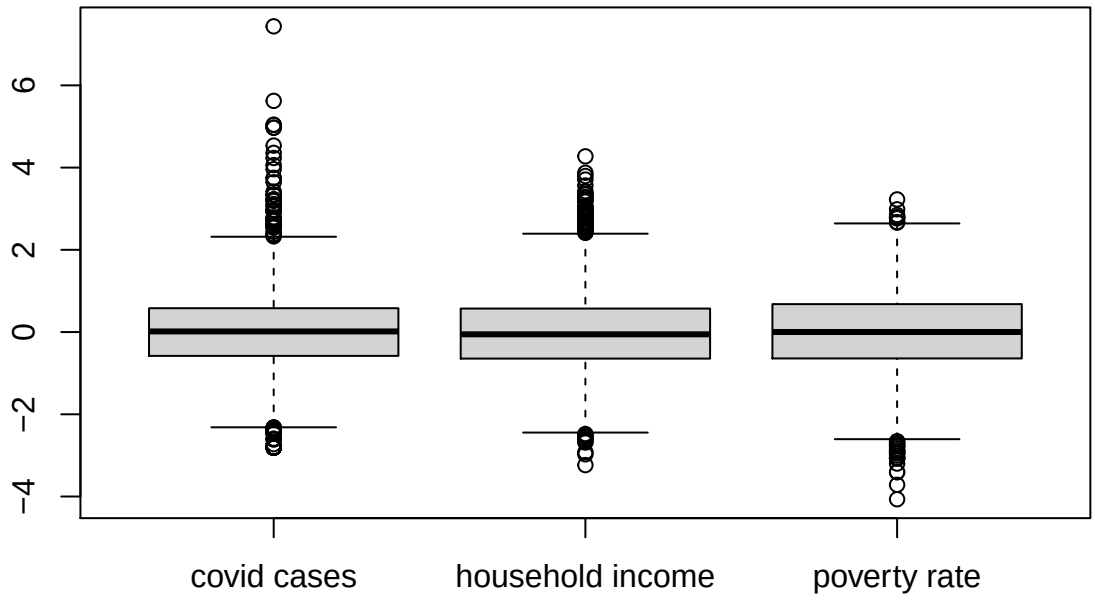
Yeah, those variables are very far from each other and differently distributed too. Let's go ahead and standardize by subtracting the mean and dividing by standard deviation.

```
raw$tr_log_covid <- (raw$logCovid_Confirmed_Cases_as_pct - mean(raw$logCovid_Confirmed_Cases_as_pct)) /
raw$tr_log_income <- (raw$logMedian_Household_Income_2019 - mean(raw$logMedian_Household_Income_2019)) /
raw$tr_log_poverty <- (raw$logPercent_Poverty_2019 - mean(raw$logPercent_Poverty_2019)) / sd(raw$logPer
boxplot(raw$tr_log_covid, raw$tr_log_income, raw$tr_log_poverty, names=c("covid cases", "household incor
```



It looks a lot better! But the distribution of COVID cases could fit the others better. It looks like the issue might be from outliers created by the log transformation. Let's try that one without.

```
raw$tr_covid <- (raw$Covid_Confirmed_Cases_as_pct - mean(raw$Covid_Confirmed_Cases_as_pct)) / sd(raw$Co
boxplot(raw$tr_covid, raw$tr_log_income, raw$tr_log_poverty, names=c("covid cases", "household income",
```



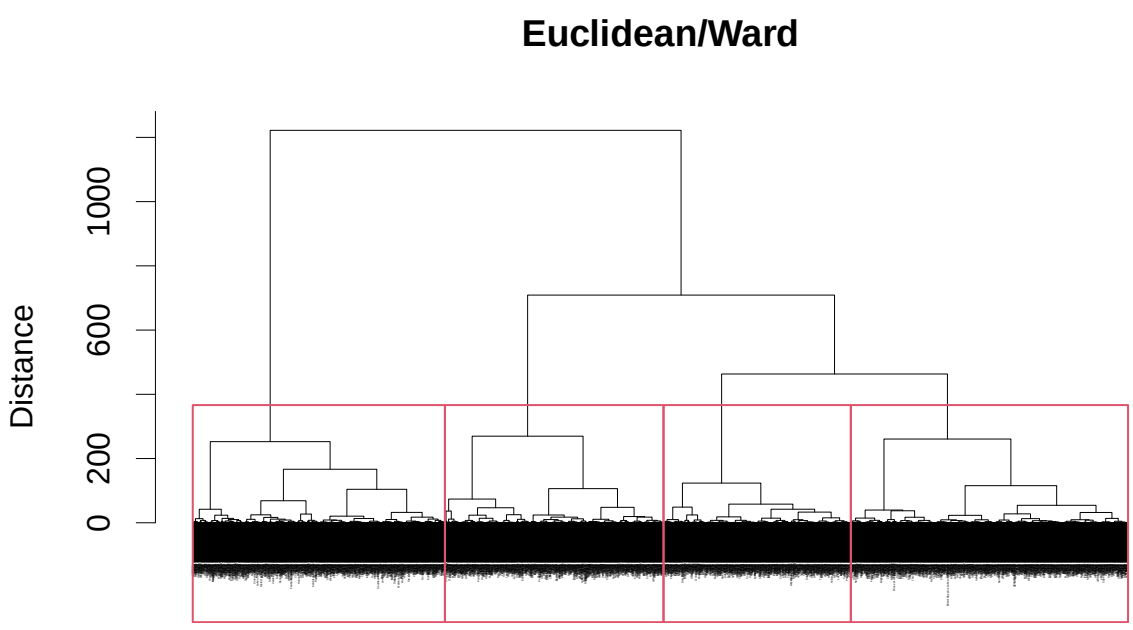
Gorgeous! All nicely standardized.

2 Hierarchical Cluster Analysis

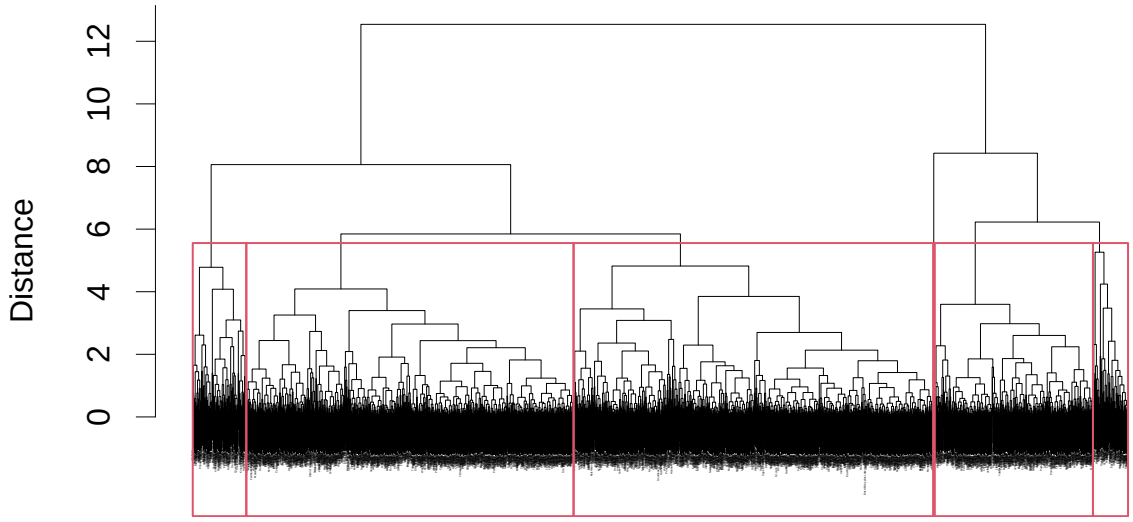
Try various forms of hierarchical cluster analysis. Try at least two different distance metrics and two agglomeration methods. Produce dendrograms and comment on what you observe.

Let's go ahead and make four dendrograms, with two distance metrics and two agglomeration methods. The distance methods are both Minowskis with very different exponents: Euclidean and a Chebychev approximation. For agglomeration methods, there are only so many options in `hclust`, and it made sense to keep our work within the same function, so we're using Ward (specifically, `ward.D`) and complete linkage.

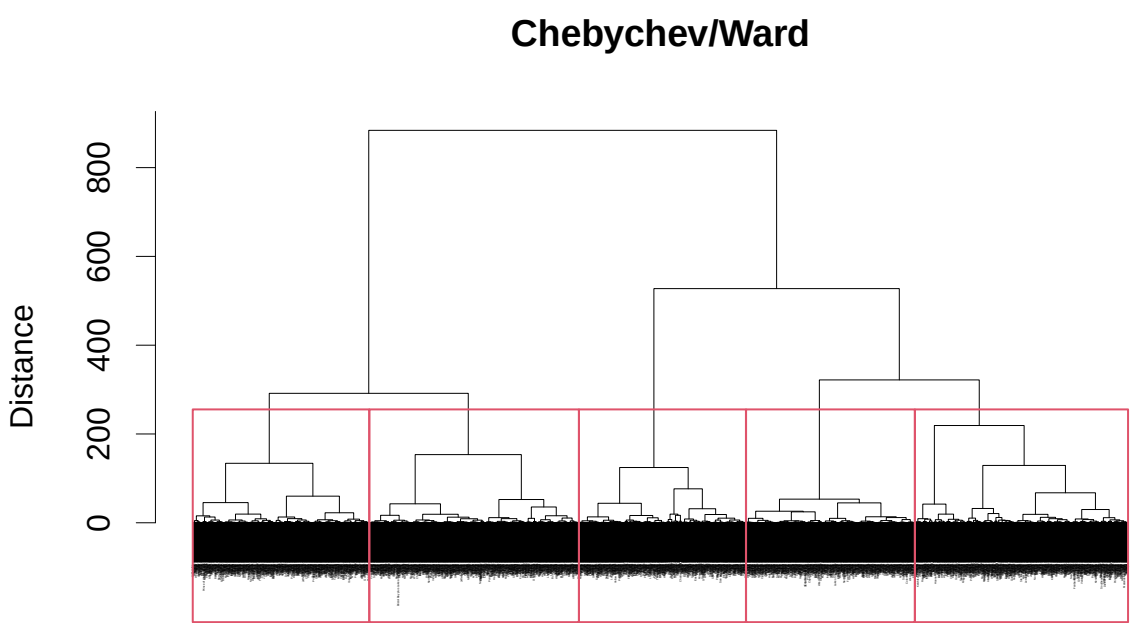
```
euclid <- dist(raw[,c("tr_covid","tr_log_income","tr_log_poverty")], method="euclidean")
euclusters_ward <- hclust(euclid, method="ward.D")
plot(euclusters_ward,labels=as.vector(raw[,"County_Name"][[1]]),cex=0.1,xlab="",ylab="Distance",lwd=0.2
```



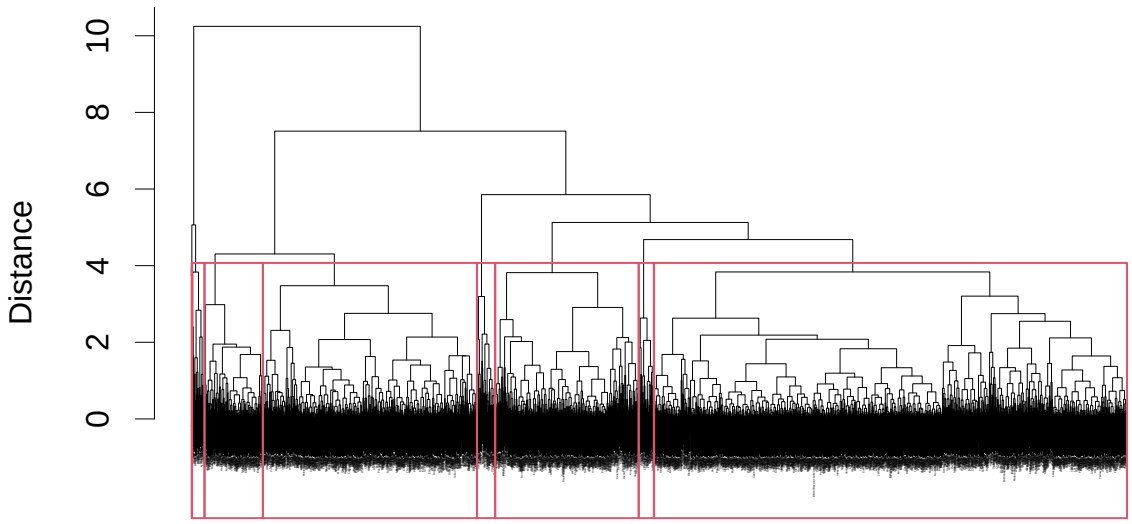

```
euclusters_complete <- hclust(euclid, method="complete")
plot(euclusters_complete,labels=as.vector(raw[,"County_Name"][[1]]),cex=0.1,xlab="",ylab="Distance",lwd=
```




```
cheby <- dist(raw[,c("tr_covid","tr_log_income","tr_log_poverty")], method="minkowski", p=20)
chebsters_ward <- hclust(cheby, method="ward.D")
plot(chebsters_ward,labels=as.vector(raw[,"County_Name"][[1]]),cex=0.1,xlab="",ylab="Distance",lwd=0.2,
```




```
chebsters_complete <- hclust(cheby, method="complete")
plot(chebsters_complete,labels=as.vector(raw[,"County_Name"][[1]]),cex=0.1,xlab="",ylab="Distance",lwd=
```



I can see why the `hclust` documentation lists Ward clustering first. It claims to identify "compact, spherical clusters", and it's hard to tell from these dendrograms exactly what that does, but it's clear that the Ward's method gives much neater groupings than the complete linkage method. That is, the complete linkage clusters become very granular at much higher distances. It looks like Ward's might be the way to go, at least for these data.

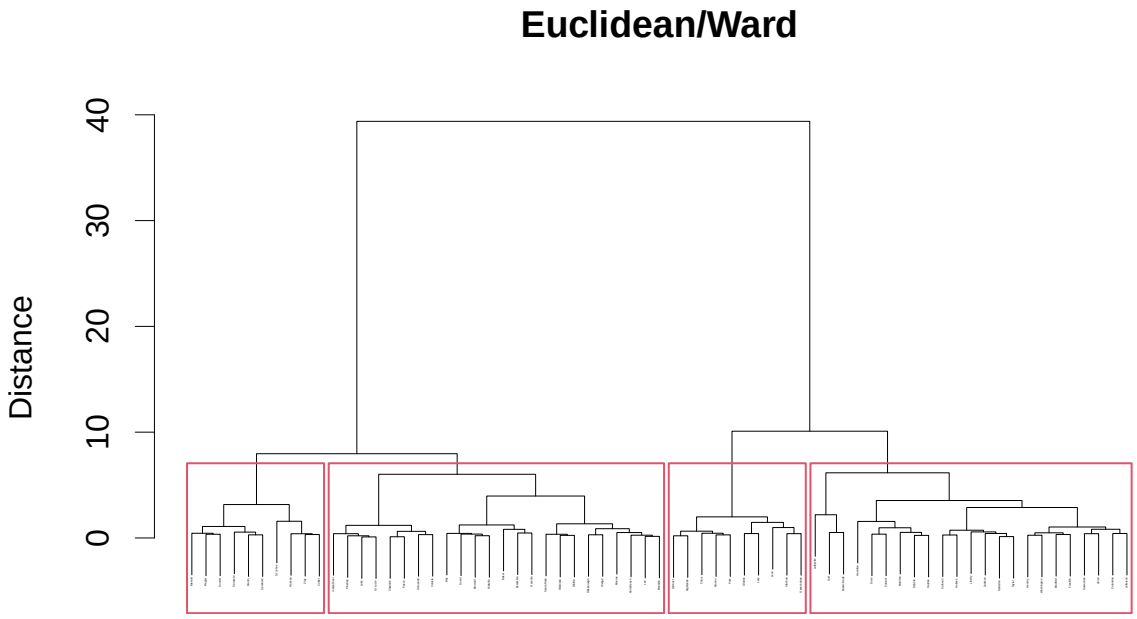
On first glance / visual inspection, the Ward-method plots have a few discrete clusters (4 and 5, respectively), as shown in the overlays on the dendrograms. To get something that reasonably resembles actual clusters on the complete linkage-method plots, you have to add more boxes (6 and 8). These plots also have much smaller clusters, of only a handful of counties by comparison.

It will be interesting to see if these clusters correlate nicely with the region or rural-urban code classifications.

The really unfortunate thing here is that the termini of the dendrograms are so close together that it's difficult to read the labels. That said, the labels don't contribute all that much given the sheer number of counties and the fact that the names don't convey much information. C'est la vie, c'est la data.

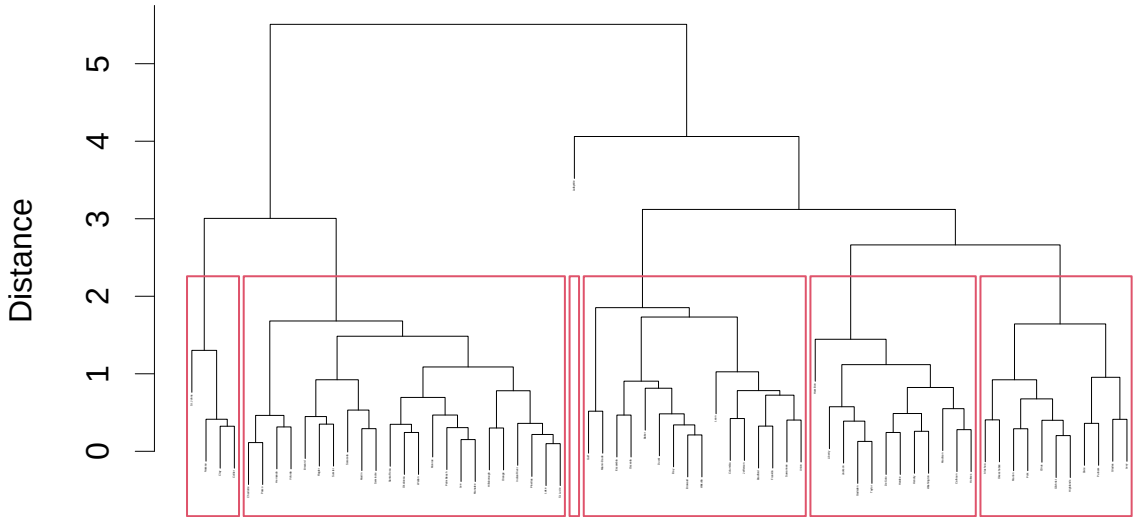
As an alternative in order to get more pedagogically valuable results (i.e., so we can see what's going on in the dendrograms) from hierarchical clustering, let's focus on the 67 counties of everyone's favorite state, Florida.

```
# Create new dataset with just the 8 CT counties
raw_FL <- raw[raw$State_Name=="Florida", ]

euclid <- dist(raw_FL[,c("tr_covid","tr_log_income","tr_log_poverty")], method="euclidean")
euclusters_ward <- hclust(euclid, method="ward.D")
plot(euclusters_ward,labels=as.vector(raw_FL[,"County_Name"][[1]]),cex=0.1,xlab="",ylab="Distance",lwd=
```



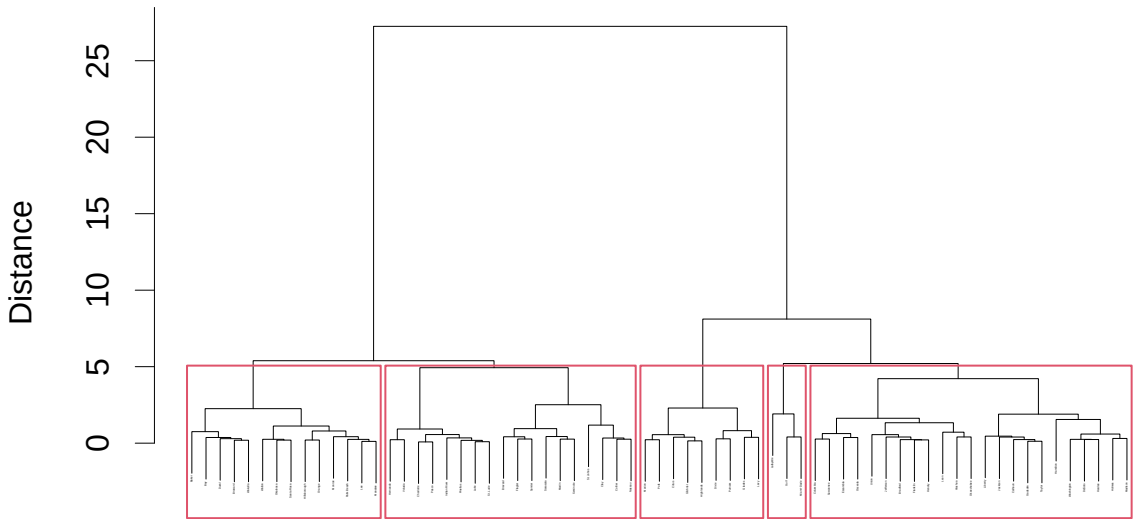

```
euclusters_complete <- hclust(euclid, method="complete")
plot(euclusters_complete,labels=as.vector(raw_FL[,"County_Name"][[1]]),cex=0.1,xlab="",ylab="Distance",
```

Euclidean/Complete

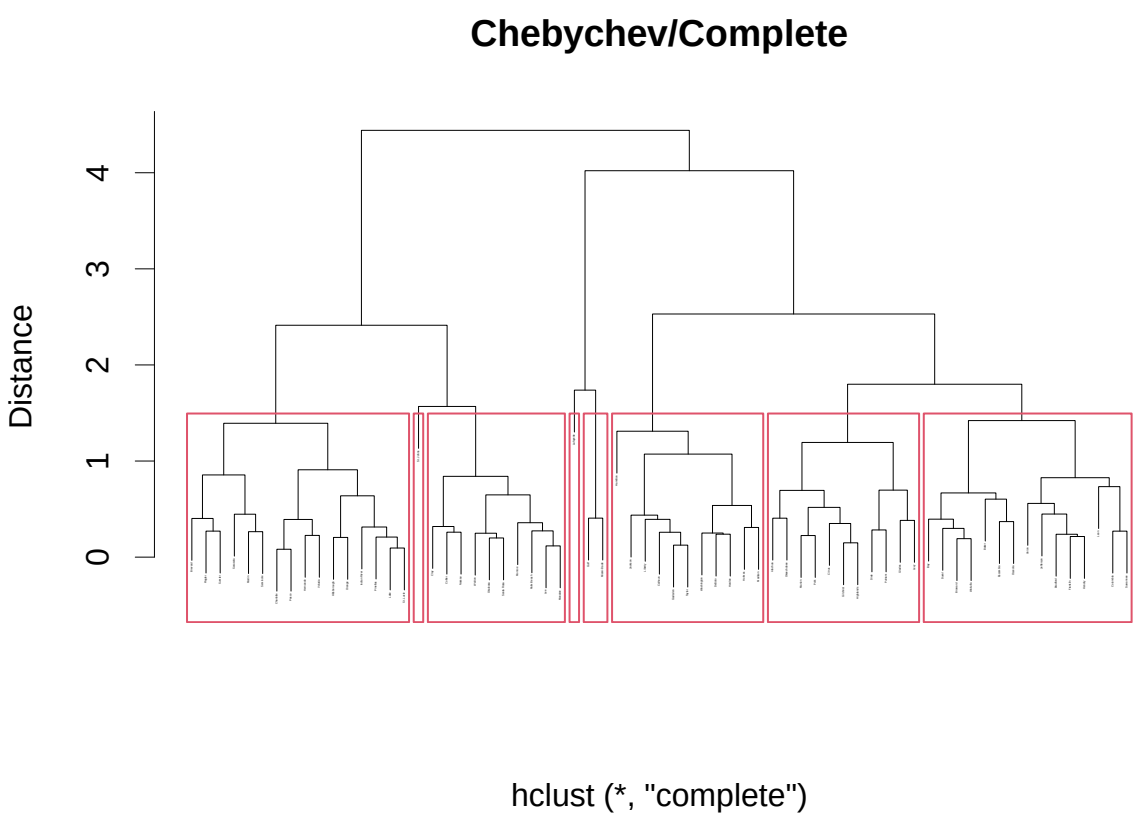


hclust (*, "complete")

```
cheby <- dist(raw_FL[,c("tr_covid","tr_log_income","tr_log_poverty")], method="minkowski", p=20)
chebsters_ward <- hclust(cheby, method="ward.D")
plot(chebsters_ward,labels=as.vector(raw_FL[,"County_Name"][[1]]),cex=0.1,xlab="",ylab="Distance",lwd=0
```



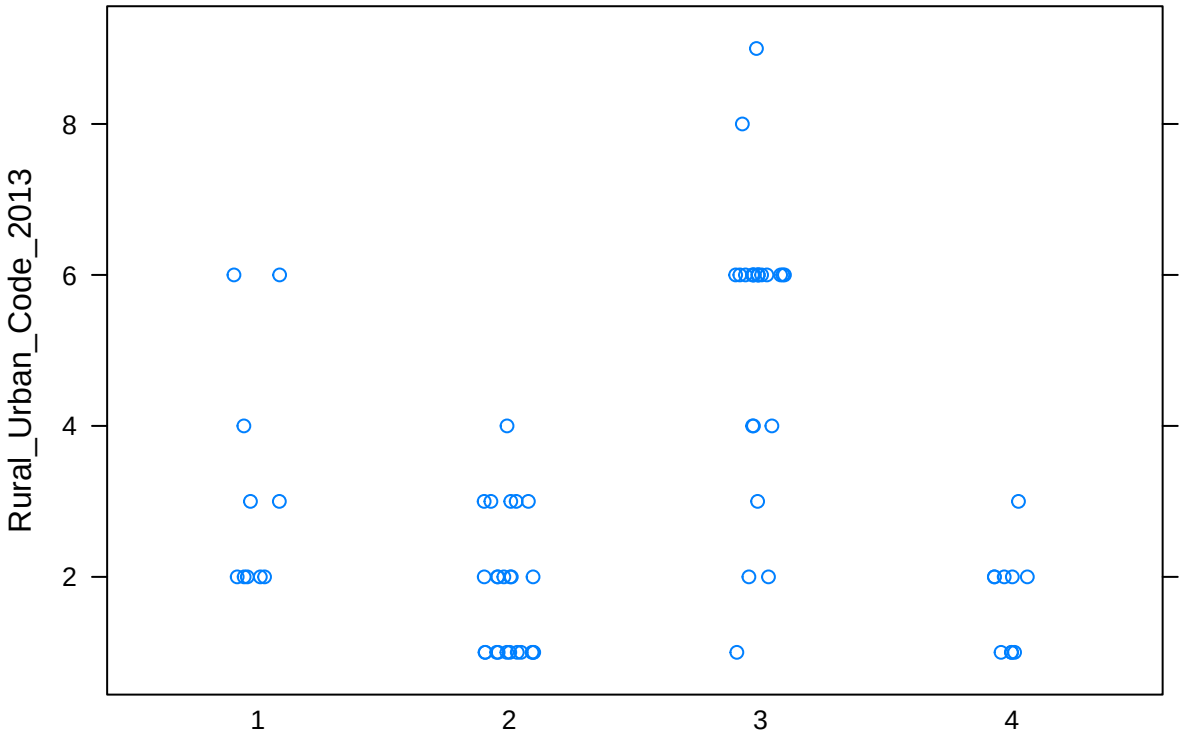

```
chebsters_complete <- hclust(cheby, method="complete")
plot(chebsters_complete,labels=as.vector(raw_FL[,"County_Name"][[1]]),cex=0.1,xlab="",ylab="Distance",lw
```



It seems that Euclidean/Ward gives the cleanest, most sensible clusters (4 in total).

To get an idea of the characteristics of these groups, let's examine the clusters.

```
cuts <- cutree(euclusters_ward, k = 4)

# add cuts to dataset
raw_FL$cuts <- as.character(cuts)
library(lattice)
stripplot(Rural_Urban_Code_2013 ~ cuts, data = raw_FL, jitter=0.1)
```



The greater the `Rural_Urban_Code_2013`, the more rural the county. It seems that the third cluster has the greatest proportion of rural counties, and cluster 2 has the greatest proportion of urban counties.

Further by-cluster analysis could be conducted.

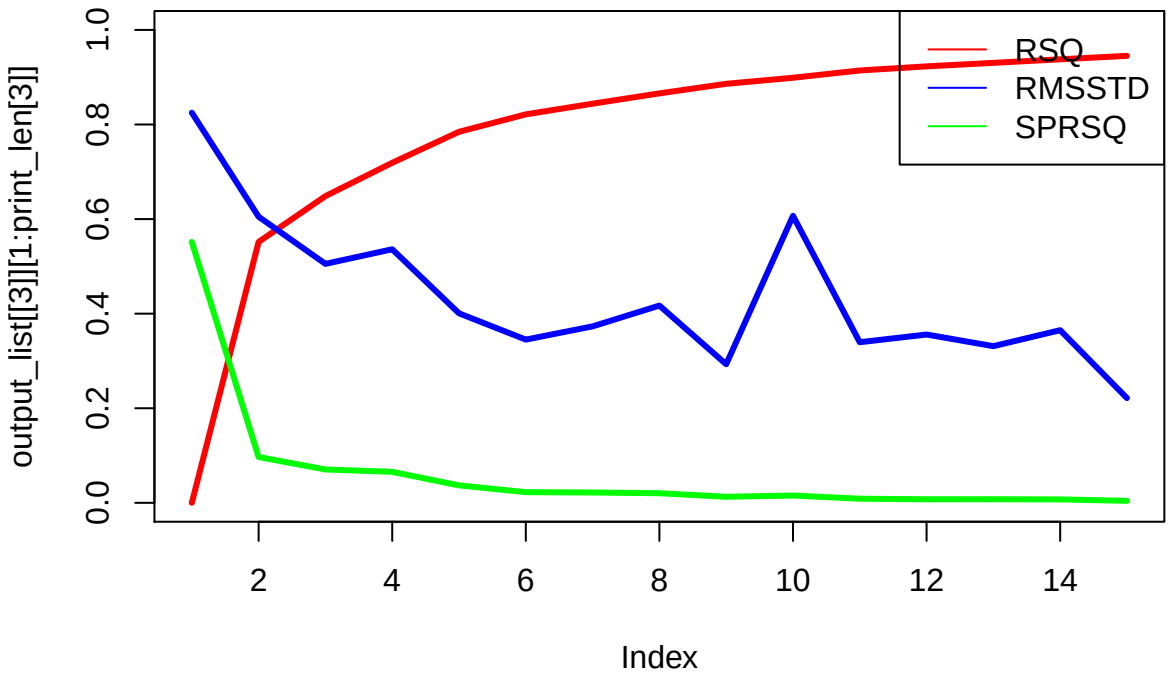
3 If possible, run the SAS macro or the R function to think about how many groups you want to retain (i.e. get the plots of cluster distance, R-squared, etc). If you can't run this, discuss how many groups you think are present.

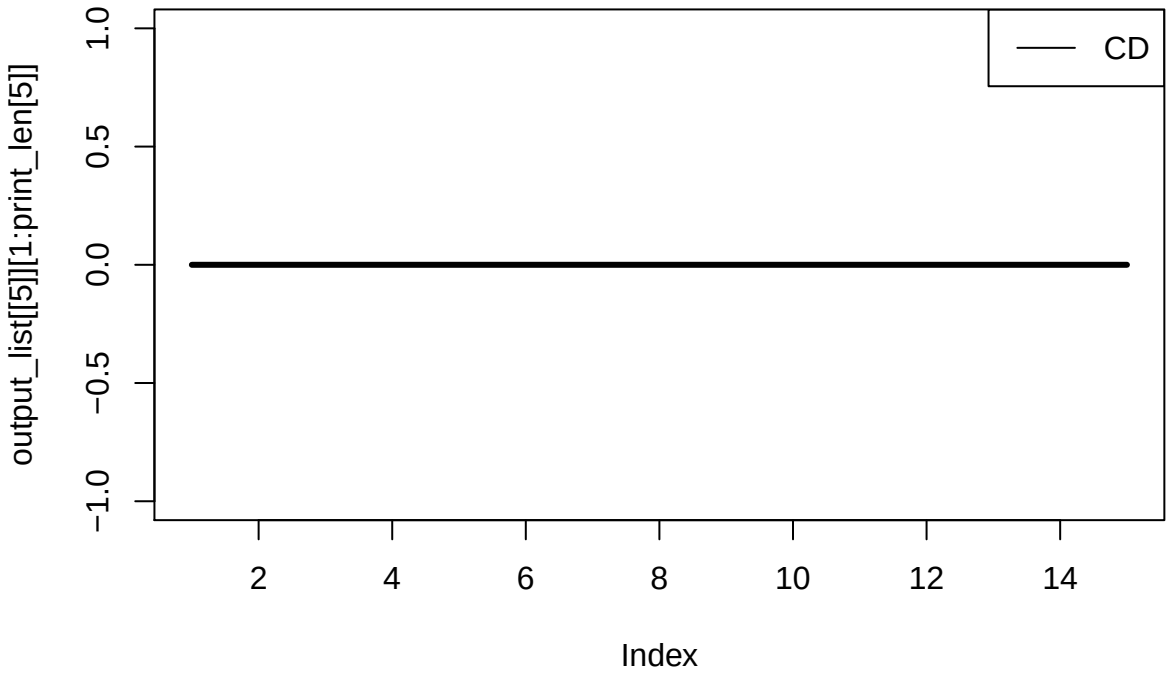
Data is approximately normal, so we will run this 'number of clusters' metric from online.

```
source("http://reuningscherer.net/multivariate/R/HClusEval3.R.txt")
#Call the function
hclus_eval(raw_FL[,c("tr_covid","tr_log_income","tr_log_poverty")], dist_m = 'euclidean', clus_m = 'ward
```

```
## [1] "Creating Distance Matrix using euclidean"
## [1] "Clustering using ward.D"
## [1] "Clustering Complete. Access the Cluster object in first element of output"
## [1] "Calculating RMSSTD"
## [1] "RMSSTD Done. Access in Element 2"
## [1] "Calculating RSQ"
## [1] "RSQ Done. Access in Element 3"
```

```
## [1] "Calculating SPRSQ"
## [1] "SPRSQ Done. Access in Element 4"
## [1] "Calculating Cluster Dist. "
## [1] "CD Done. Access in Element 5"
```






```
## [[1]]
##
## Call:
## hclust(d = dist1, method = clus_m)
##
## Cluster method   : ward.D
## Distance         : euclidean
## Number of objects: 67
##
##
## [[2]]
##  [1] 0.82499459 0.60500656 0.50531899 0.53613852 0.40070889 0.34513189
##  [7] 0.37314664 0.41712324 0.29303554 0.60693307 0.33964058 0.35586893
## [13] 0.33125007 0.36503604 0.22147681 0.25257921 0.18820170 0.23456962
## [19] 0.23022987 0.29939673 0.22975521 0.18750123 0.23164503 0.26903922
## [25] 0.18535424 0.17010961 0.15051681 0.16242382 0.17681583 0.16376943
## [31] 0.21117830 0.13451493 0.15052584 0.19087126 0.15447770 0.16284963
## [37] 0.14076947 0.17257174 0.12067625 0.16833414 0.16554663 0.10341776
## [43] 0.16451214 0.14814926 0.11975749 0.14709229 0.14301667 0.12176098
## [49] 0.13272246 0.13202196 0.12835975 0.12270834 0.11919547 0.11884435
## [55] 0.11398114 0.08897457 0.10526396 0.09914042 0.09909836 0.08614899
## [61] 0.06487088 0.08272727 0.06181640 0.05201839 0.04593390 0.04029365
## [67] 0.00000000
##
## [[3]]
##  [1] 0.0000000 0.5516648 0.6485261 0.7190388 0.7845212 0.8214391 0.8439576
```

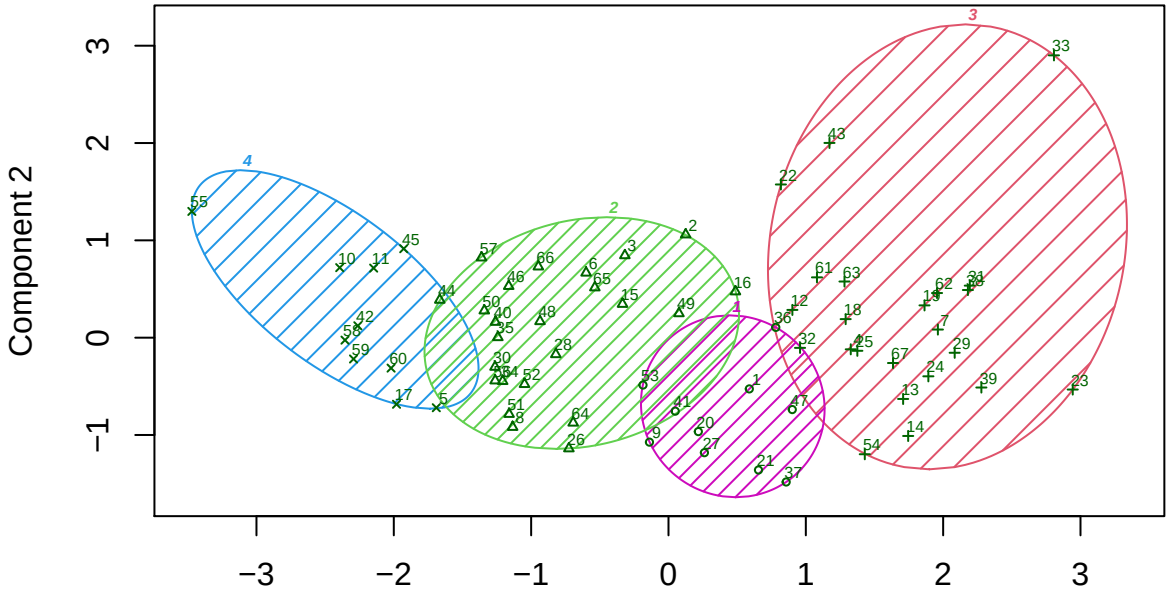
```
##  [8] 0.8657870 0.8860646 0.8988313 0.9142393 0.9229090 0.9303895 0.9379024
## [15] 0.9451461 0.9493086 0.9532837 0.9565759 0.9601276 0.9632907 0.9666716
## [22] 0.9696963 0.9720659 0.9743841 0.9767957 0.9785687 0.9799306 0.9810299
## [29] 0.9821434 0.9832191 0.9841946 0.9851874 0.9860433 0.9869177 0.9877287
## [36] 0.9884768 0.9892022 0.9898871 0.9905500 0.9911382 0.9917690 0.9923791
## [43] 0.9929060 0.9935085 0.9940977 0.9945710 0.9950526 0.9955080 0.9959493
## [50] 0.9963414 0.9967294 0.9970962 0.9974314 0.9977477 0.9980621 0.9983513
## [57] 0.9986187 0.9988654 0.9990842 0.9993028 0.9994680 0.9996192 0.9997716
## [64] 0.9998566 0.9999169 0.9999639 1.0000000
##
## [[4]]
##  [1] 5.516648e-01 9.686129e-02 7.051274e-02 6.548243e-02 3.691788e-02
##  [6] 2.251852e-02 2.182935e-02 2.027761e-02 1.276665e-02 1.540804e-02
## [11] 8.669666e-03 7.480559e-03 7.512873e-03 7.243706e-03 4.162468e-03
## [16] 3.975124e-03 3.292194e-03 3.551747e-03 3.163093e-03 3.380875e-03
## [21] 3.024729e-03 2.369587e-03 2.318154e-03 2.411638e-03 1.773010e-03
## [26] 1.361926e-03 1.099267e-03 1.113496e-03 1.075677e-03 9.755059e-04
## [31] 9.927787e-04 8.559503e-04 8.743936e-04 8.110264e-04 7.480464e-04
## [36] 7.254175e-04 6.848612e-04 6.629690e-04 5.881393e-04 6.308096e-04
## [41] 6.100909e-04 5.269120e-04 6.024899e-04 5.891857e-04 4.733248e-04
## [46] 4.816523e-04 4.553310e-04 4.412813e-04 3.921414e-04 3.880129e-04
## [51] 3.667850e-04 3.351985e-04 3.162812e-04 3.144206e-04 2.892144e-04
## [56] 2.673976e-04 2.466682e-04 2.188041e-04 2.186185e-04 1.652168e-04
## [61] 1.512196e-04 1.523531e-04 8.506703e-05 6.023759e-05 4.696999e-05
## [66] 3.614325e-05 0.000000e+00
##
## [[5]]
##  [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 0 0 0 0 0 0 0 0 0 0 0 0 0
## [39] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

From the guidance statistics, the number of clusters is not totally clear. There is a localized peak at the root mean squared standard deviation (RMSSTD) at 4, which reinforces our conclusion that there are 4 distinct clusters. The r-squared tapers off at 2 (and subtly at 4), which may suggest a lower number of clusters is optimal. The semi-partial r-squared also flattens near 2 (subtly at 4), which may suggest a lower number of clusters is optimal.

Addendum

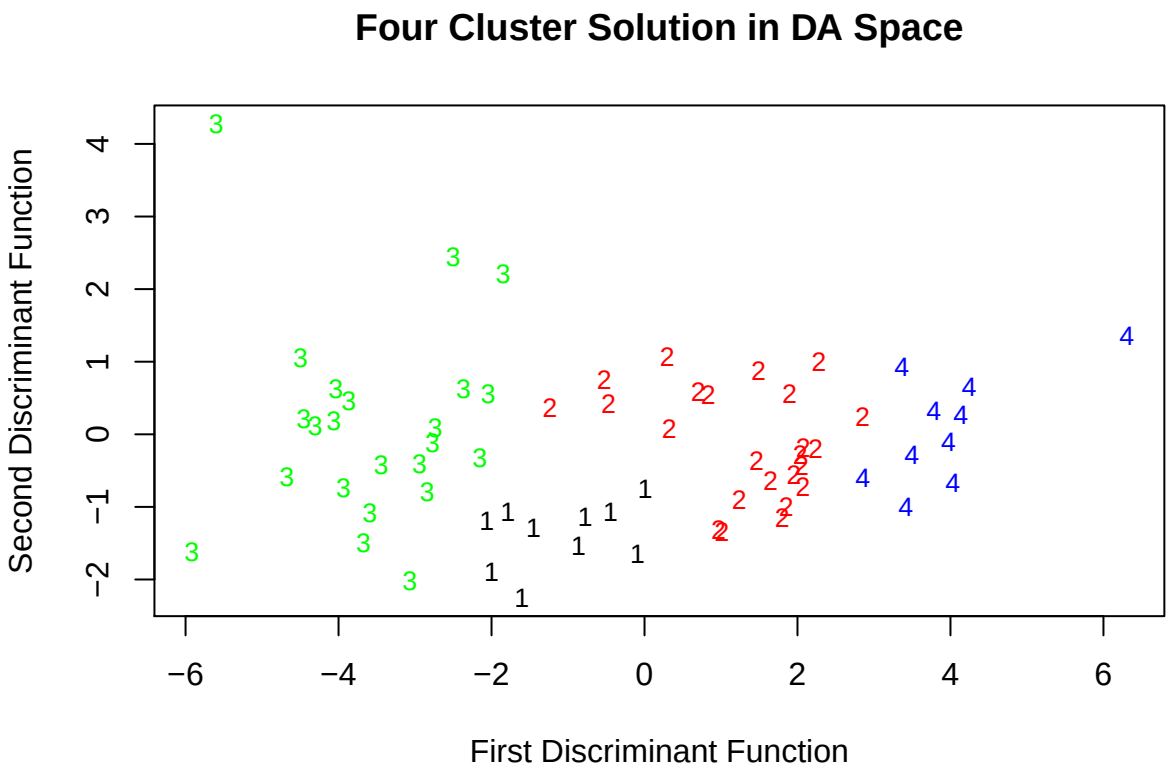
Make plot of five cluster solution in space designated by first two principal components.

```
#ctnorm <- raw_FL[,c("tr_covid","tr_log_income","tr_log_poverty")]
#rownames(ctnorm) <- as.factor(pull(raw_FL[, 2]))
#ctnorm <- scale(na.omit(ctnorm))

clusplot(raw_FL[,c("tr_covid","tr_log_income","tr_log_poverty")], cuts, color = TRUE, shade = TRUE, labe
         main = "County Cluster Plot, Ward's Method, First two PC", cex = .5)
```



Make plot of five cluster solution in space designated by first two discriminant functions.

```
plotcluster(raw_FL[,c("tr_covid","tr_log_income","tr_log_poverty")], cuts, main = "Four Cluster Solutio
```



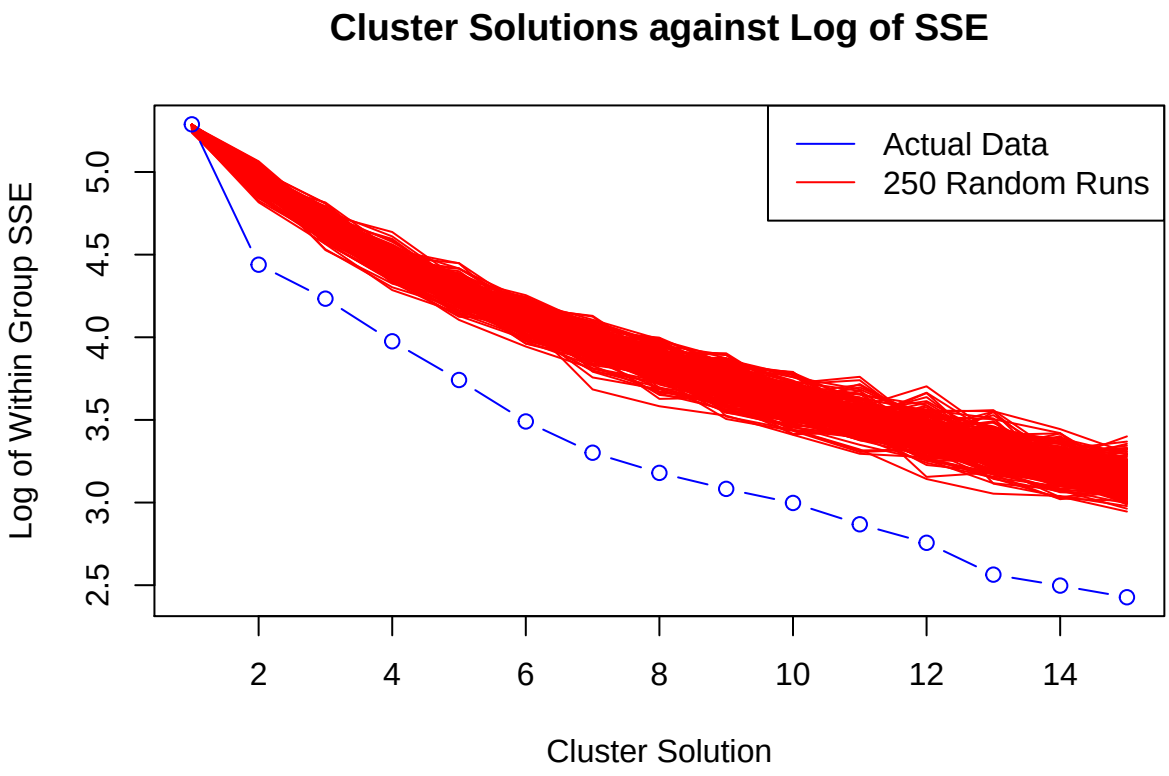
4 Run k-means clustering on your data. Compare results to what you got in 3.) Include a sum of squares vs. k (number of clusters) plot and comment on how many groups exist.

Produces screeplot like diagram with randomized comparison based on randomization within columns (i.e. as if points had been randomly assigned data values, one from each column. Keeps total internal SS the same.

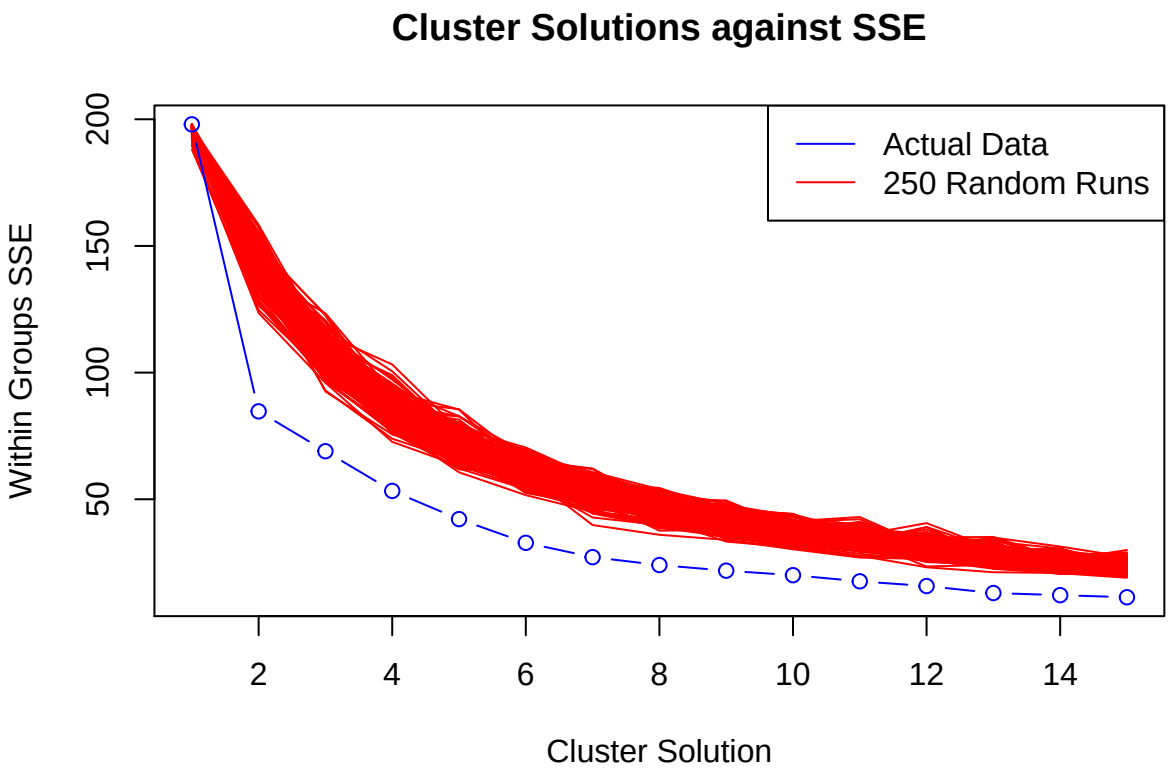
Evaluates total cluster number 1-15 to optimize.

```
#kdata is just normalized input dataset
ctnorm <- raw_FL[,c("tr_covid","tr_log_income","tr_log_poverty")]
rownames(ctnorm) <- as.vector(raw_FL[,"County_Name"][[1]])
ctnorm <- scale(na.omit(ctnorm))

kdata <- ctnorm
n.lev <- 15  #set max value for number of clusters k

# Calculate the within groups sum of squared error (SSE) for the number of cluster solutions selected by
wss <- rnorm(10)
while (prod(wss==sort(wss,decreasing=T))==0) {
  wss <- (nrow(kdata)-1)*sum(apply(kdata,2,var))
  for (i in 2:n.lev) wss[i] <- sum(kmeans(kdata, centers=i)$withinss)}

# Calculate the within groups SSE for 250 randomized data sets (based on the original input data)
```

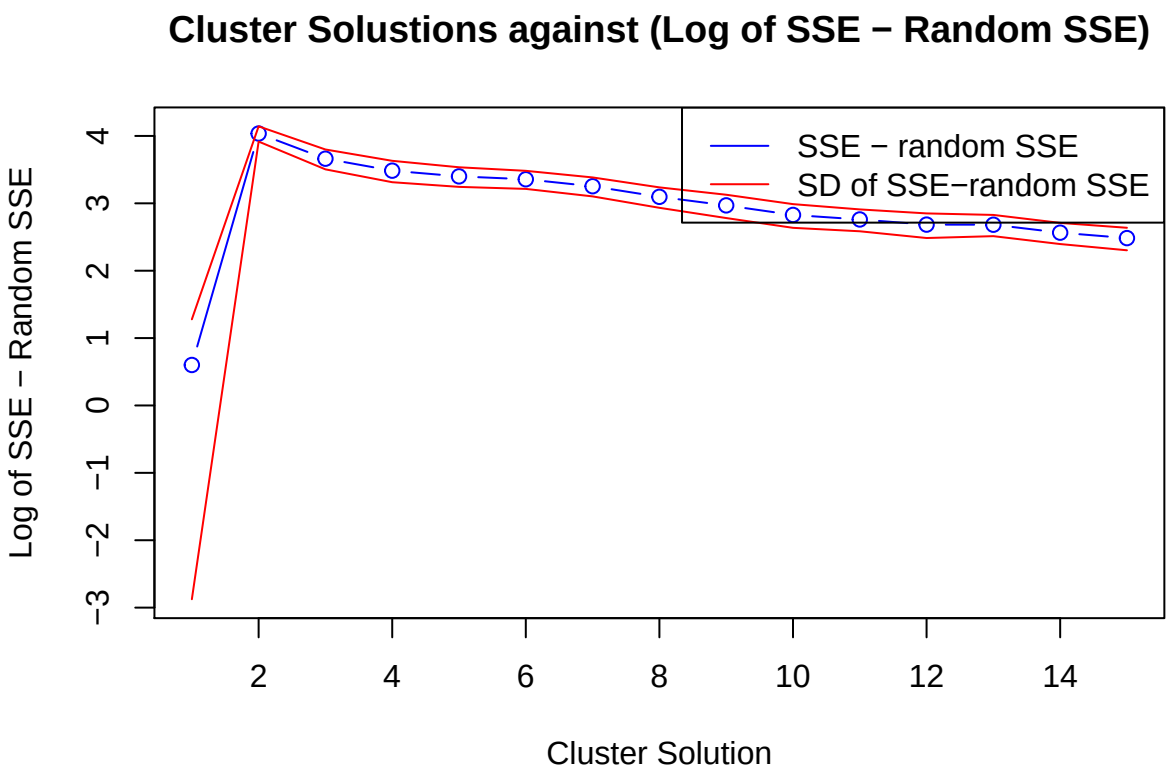
```r
k.rand <- function(x){
  km.rand <- matrix(sample(x),dim(x)[1],dim(x)[2])
  rand.wss <- as.matrix(dim(x)[1]-1)*sum(apply(km.rand,2,var))
  for (i in 2:n.lev) rand.wss[i] <- sum(kmeans(km.rand, centers=i)$withinss)
  rand.wss <- as.matrix(rand.wss)
  return(rand.wss)
}

rand.mat <- matrix(0,n.lev,250)

k.1 <- function(x) {
  for (i in 1:250) {
    r.mat <- as.matrix(suppressWarnings(k.rand(kdata)))
    rand.mat[,i] <- r.mat}
  return(rand.mat)
}

# Same function as above for data with < 3 column variables
k.2.rand <- function(x){
  rand.mat <- matrix(0,n.lev,250)
  km.rand <- matrix(sample(x),dim(x)[1],dim(x)[2])
  rand.wss <- as.matrix(dim(x)[1]-1)*sum(apply(km.rand,2,var))
  for (i in 2:n.lev) rand.wss[i] <- sum(kmeans(km.rand, centers=i)$withinss)
  rand.wss <- as.matrix(rand.wss)
  return(rand.wss)
}

k.2 <- function(x){
  for (i in 1:250) {
    r.1 <- k.2.rand(kdata)
    rand.mat[,i] <- r.1}
  return(rand.mat)
}

# Determine if the data data table has > or < 3 variables and call appropriate function above
if (dim(kdata)[2] == 2) { rand.mat <- k.2(kdata) } else { rand.mat <- k.1(kdata) }

# Plot within groups SSE against all tested cluster solutions for actual and randomized data - 1st: Log

xrange <- range(1:n.lev)
yrange <- range(log(rand.mat),log(wss))
plot(xrange,yrange, type='n', xlab='Cluster Solution', ylab='Log of Within Group SSE', main='Cluster Sol
for (i in 1:250) lines(log(rand.mat[,i]),type='l',col='red')
lines(log(wss), type="b", col='blue')
legend('topright',c('Actual Data', '250 Random Runs'), col=c('blue', 'red'), lty=1)
```



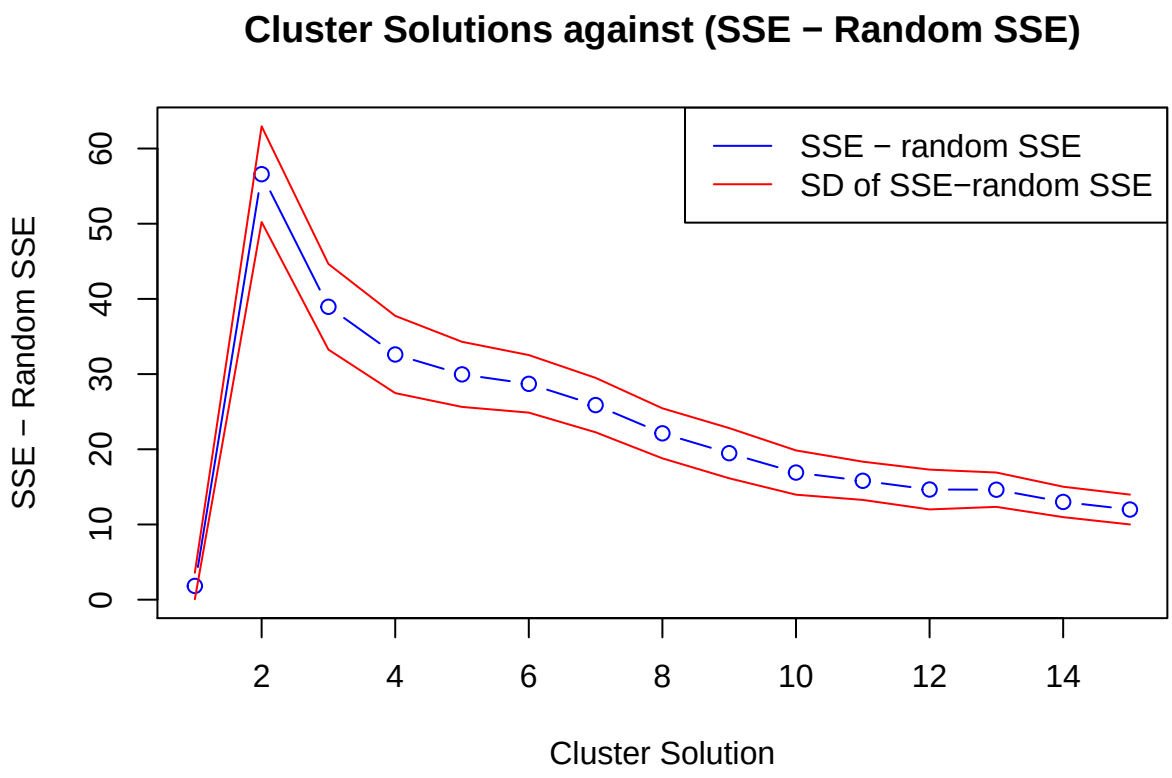

```
yrange <- range(rand.mat,wss)
plot(xrange,yrange, type='n', xlab="Cluster Solution", ylab="Within Groups SSE", main="Cluster Solutions
for (i in 1:250) lines(rand.mat[,i],type='l',col='red')
lines(1:n.lev, wss, type="b", col='blue')
legend('topright',c('Actual Data', '250 Random Runs'), col=c('blue', 'red'), lty=1)
```




```
# Calculate the mean and standard deviation of difference between SSE of actual data and SSE of 250 ran
r.sse <- matrix(0,dim(rand.mat)[1],dim(rand.mat)[2])
wss.1 <- as.matrix(wss)
for (i in 1:dim(r.sse)[2]) {
  r.temp <- abs(rand.mat[,i]-wss.1[,1])
  r.sse[,i] <- r.temp}
r.sse.m <- apply(r.sse,1,mean)
r.sse.sd <- apply(r.sse,1,sd)
r.sse.plus <- r.sse.m + r.sse.sd
r.sse.min <- r.sse.m - r.sse.sd

# Plot differeince between actual SSE mean SSE from 250 randomized datasets - 1st: Log scale, 2nd: Norm

xrange <- range(1:n.lev)
yrange <- range(log(r.sse.plus),log(r.sse.min))
plot(xrange,yrange, type='n',xlab='Cluster Solution', ylab='Log of SSE - Random SSE', main='Cluster Solu
lines(log(r.sse.m), type="b", col='blue')
lines(log(r.sse.plus), type='l', col='red')
lines(log(r.sse.min), type='l', col='red')
legend('topright',c('SSE - random SSE', 'SD of SSE-random SSE'), col=c('blue', 'red'), lty=1)
```




```
xrange <- range(1:n.lev)
yrange <- range(r.sse.plus,r.sse.min)
plot(xrange,yrange, type='n',xlab='Cluster Solution', ylab='SSE - Random SSE', main='Cluster Solutions a
lines(r.sse.m, type="b", col='blue')
lines(r.sse.plus, type='l', col='red')
lines(r.sse.min, type='l', col='red')
legend('topright',c('SSE - random SSE', 'SD of SSE-random SSE'), col=c('blue', 'red'), lty=1)
```



The first plot (i.e., the log of within group sum of squared errors) suggests that there is less change after 4-7 clusters.

The second plot (i.e., original scale of within group sum of squared errors) also indicates that 4 clusters is optimal.

The third plot (i.e., the standard deviation of the difference between log SSE and random SSE) suggests that 2 clusters is probably optimal (it stabilizes roughtly at 2).

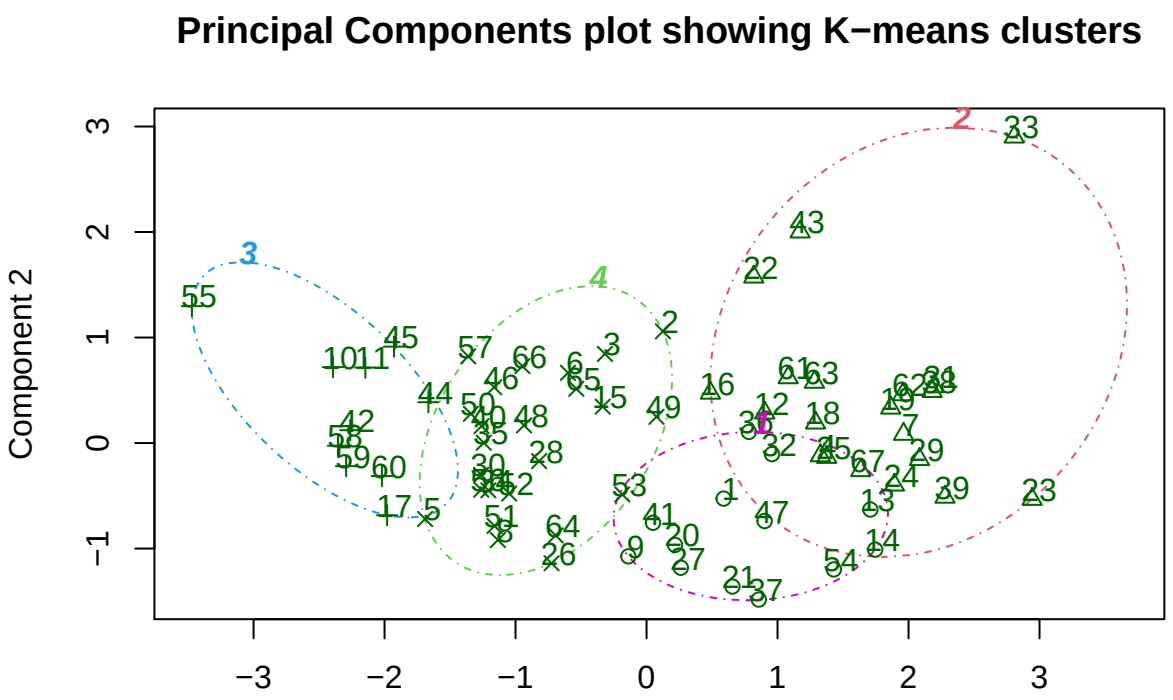
The fourth plot (i.e., the standard deviation of the difference between SSE and random SSE) suggests that 2 or 5 clusters are optimal.

Now, let's take a look at our K-means results in principal components space (with 4 clusters).

```
# Ask for user input - Select the appropriate number of clusters
#choose.clust <- function(){readline("What clustering solution would you like to use? ")}
#clust.level <- as.integer(choose.clust())
clust.level <- 4

# Apply K-means cluster solutions - append clusters to CSV file
fit <- kmeans(kdata, clust.level)
aggregate(kdata, by=list(fit$cluster), FUN=mean)
```

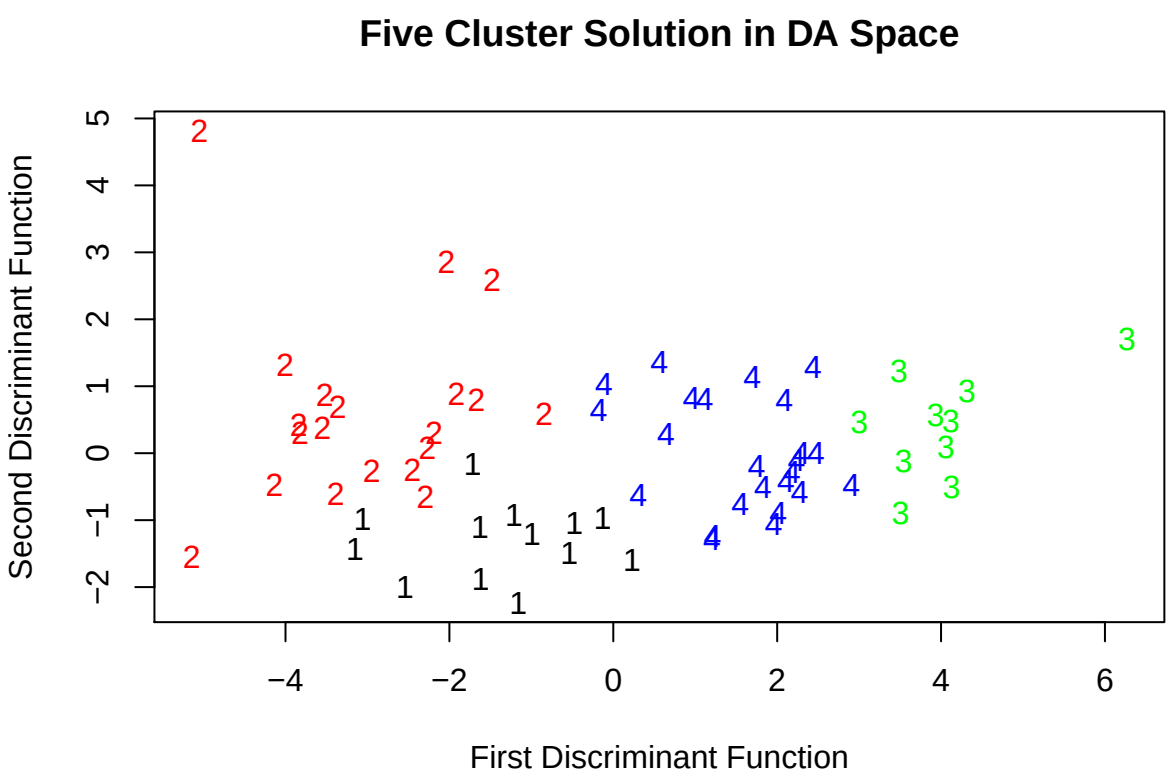
```
##   Group.1   tr_covid tr_log_income tr_log_poverty
## 1       1 -0.3756755    -0.7695112      0.7445722
## 2       2  1.1568257    -0.8598410      0.9311827
## 3       3 -0.8081572     1.4941987     -1.4803722
## 4       4 -0.4237983     0.5107699     -0.5624738
```

```
clust.out <- fit$cluster
kclust <- as.matrix(clust.out)
kclust.out <- cbind(kclust, ctnorm)
write.table(kclust.out, file="kmeans_out.csv", sep=",")

# Display Principal Components plot of data with clusters identified

clusplot(kdata, fit$cluster, shade=F, labels=2, lines=0, color=T, lty=4, main='Principal Components plo
```



These two components explain 97.48 % of the point variability.

```
#Make plot of four cluster solution in space desginated by first two
#  two discriminant functions

plotcluster(kdata, fit$cluster, main="Five Cluster Solution in DA Space",
            xlab="First Discriminant Function", ylab="Second Discriminant Function")
```



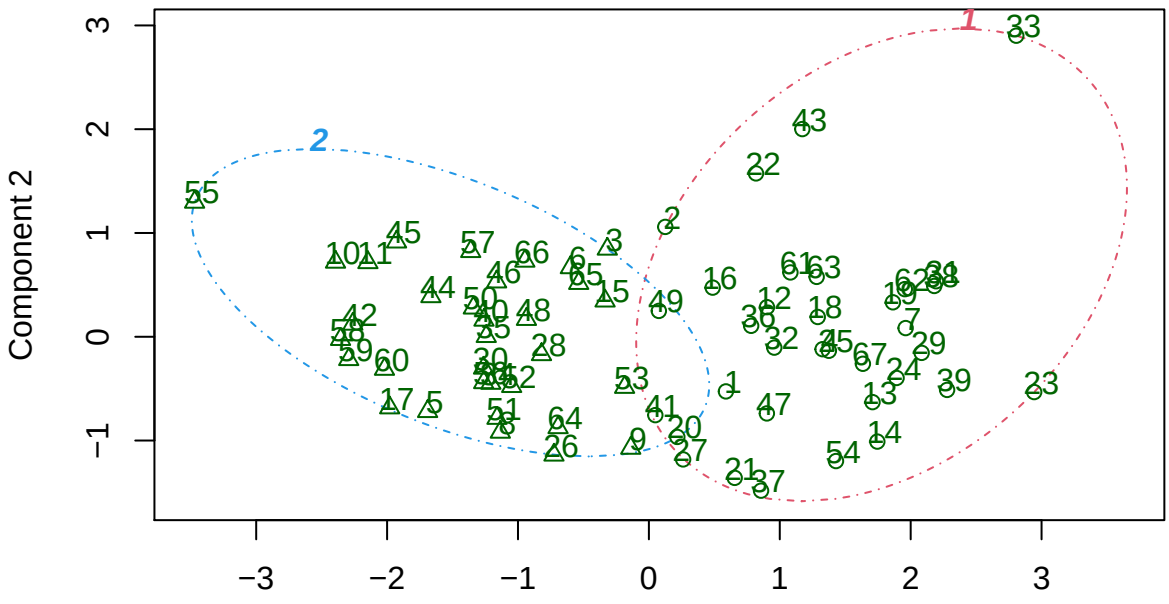
Based on visual inspection, we note that k-means clustering results in a similar grouping result as observed in Part 3.

Let's try it with 2 clusters:

```
# Ask for user input - Select the appropriate number of clusters
#choose.clust <- function(){readline("What clustering solution would you like to use? ")}
#clust.level <- as.integer(choose.clust())
clust.level <- 2

# Apply K-means cluster solutions - append clusters to CSV file
fit <- kmeans(kdata, clust.level)
aggregate(kdata, by=list(fit$cluster), FUN=mean)
```

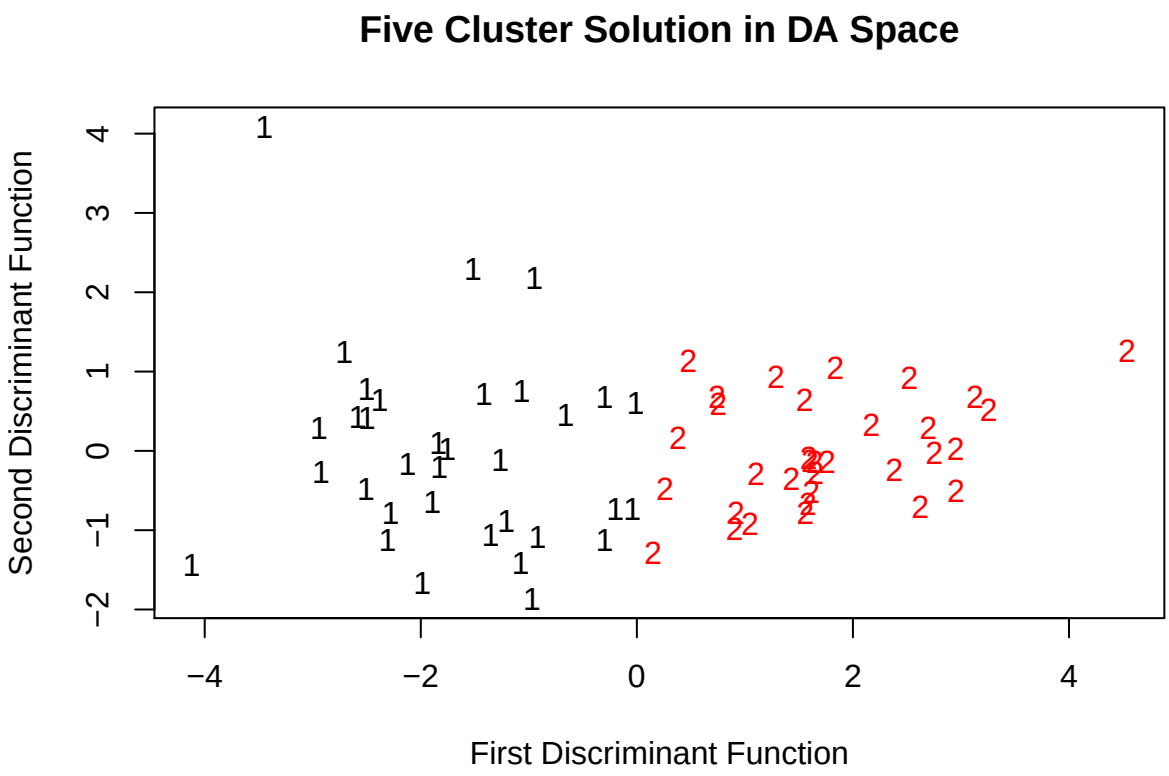
```
##   Group.1   tr_covid tr_log_income tr_log_poverty
## 1       1  0.6015628    -0.7735006      0.8252400
## 2       2 -0.6197920     0.7969400     -0.8502472
```

```
clust.out <- fit$cluster
kclust <- as.matrix(clust.out)
kclust.out <- cbind(kclust, ctnorm)
write.table(kclust.out, file="kmeans_out2.csv", sep=",")

# Display Principal Components plot of data with clusters identified

clusplot(kdata, fit$cluster, shade=F, labels=2, lines=0, color=T, lty=4, main='Principal Components plo
```



These two components explain 97.48 % of the point variability.

```
#Make plot of four cluster solution in space desginated by first two
#  two discriminant functions

plotcluster(kdata, fit$cluster, main="Five Cluster Solution in DA Space",
            xlab="First Discriminant Function", ylab="Second Discriminant Function")
```



5 Comment on the number of groups that seem to be present based on what you find above.

Recall that we are clustering U.S. counties based on 3 metric, standardized variables: the percentage of confirmed COVID-19 cases relative to the county population (from February 2021), the log of the median income (from 2019), and the log of the percentage of people of all ages in poverty (from 2019). We are trying to find groups of counties with members that are similar to each other but different from other groups - we are finding clusters of *observations*, like PCA where we found clusters of *variables*.

From our work with hierarchial cluster analysis in Part 2, there appear to be 4 or 5 groups of counties (using Euclidean distance, with 4 groups from Ward agglomeration and 5 groups from complete linkage). To make the dataset more manageable, we zoom in on our favorite state Florida and focus on clustering only Florida's counties from here on.

Our analysis indicates that either 2 clusters or 4 clusters would be appropriate for the Florida counties.

The evidence in support of 2 clusters: R-squared, semi-partial R-squared, the standard deviation of the difference between log SSE and random SSE, the standard deviation of the difference between SSE and random SSE

The evidence in support of 4 clusters: Euclidean/Ward, root mean squared standard deviation (RMSSTD), log of within group sum of squared errors, original scale of within group sum of squared errors

There are 4 points in evidence in support of either 2 clusters or 4 clusters. Both can be justified using statistical techniques, so we will look at what makes sense from an interpretation perspective below. Ultimately, 4 clusters is what makes most sense to us.

6 Write a few sentences describing your final groups.

With 2 clusters, we get Cluster 1 with high COVID-19 percentages, low income, and high poverty; and Cluster 2 with low COVID-19 percentages, high income, and low poverty. These clusters are opposites in every way. There are equal amounts of counties in both groups.

With 4 clusters, we get Cluster 1 with low COVID-19 percentages, high income, and low poverty; Cluster 2 with high COVID-19 percentages, low income, and high poverty; Cluster 3 with low COVID-19 percentages, low income, and high poverty; and Cluster 4 with low COVID-19 percentages, high income, and low poverty. In short, Cluster 1 and Cluster 4 are healthy and wealthy. Cluster 2 is impoverished and experiencing high rates of COVID-19; Cluster 3 is impoverished and experiencing low rates of COVID-19. We note that Cluster 2 seems to have more urban character, while Cluster 3 has more rural character.

Ultimately, we chose 4 clusters to describe how COVID-19 has spread through communities with different incomes and poverty levels. The story told here mimics the one we have been hearing over and over in the news. Poorer urban communities are being hit hardest by COVID-19. Members of these communities tend to have jobs as essential workers that require work outside of the home. They may have to take public transit. They may be unable to afford grocery delivery services. Affluent communities have the resources to avoid COVID-19 transmission, while impoverished communities may not. We find it of note that our 4 cluster model further distinguishes between impoverished urban and impoverished rural areas - though their incomes may be similar, a city grocery store cashier likely has more work-related exposure to COVID-19 than a rural area farmer. Poor urban communities are particularly vulnerable to COVID-19 transmission.

We also note that these same communities are particularly vulnerable to severe COVID-19. They are likely to have preexisting conditions that worsen its effects and may lack quality healthcare or health insurance. As we continue to distribute COVID-19 vaccinations, special attention should be placed on supporting these communities most at risk for COVID-19.