Full Data

Claudia Shrefler

6/6/2019

#Data

##Food Environment Atlas Data

Source: https://www.ers.usda.gov/data-products/food-environment-atlas/data-access-and-documentation-downloads/

Loaded in each sheet of the excel formatted data set as separate .csv files.

Subsetted the data sets to only PA variables.

Changed values of -9999 as described in the documentation to NA.

##American Commuity Survey Data

Source: https://www.census.gov/acs/www/data/data-tables-and-tools/

Read in data set. Changed values of (X) described in documentation to NA.

##Census Data

Source: https://www.census.gov/data.html

Read in data set. Changed values of (X) described in documentation to NA.

##Feeding America Data

Source: https://www.feedingamerica.org/research/map-the-meal-gap/by-county

Read in data set and subset it to just PA.

Modify variables that included percent signs to get rid of percent using gsub() and put them in decimal form by dividing by 100. Modify columns to get rid of dollar signs and commas. Use as.numeric() to put vectors back into numeric form since gsub() returns character vectors.

##PA Legislature County Profile Data

Source: https://www.rural.palegislature.us/county_profiles.cfm

Read in data set from PA legislature. Transpose the data set to select variables more easily and convert back into a data frame.

Pick education data by selecting correct column and eliminating first two rows (variable names and PA total), and sort into my own variables. Clean the data to get rid of percent signs and divide by 100 to put into decimal form.

Pick population column and get rid of commas.

Get median household income data from 2016 and eliminate dollar signs and commas.

Get average commuting time to work in minutes in 2016.

Get number of licensed drivers in 2016.

Get number of registered vehicles and number of vehicles per 1000 residents in 2017.

Get number of farms, the percent of land in farms, and the total market value of agricultural products sold in 2012.

Get acres of preserved farm land in 2017.

Make temporary data frame out of County Profile variables to add to full data set.

##US Dept Labor Bureau and Labor Statistics Data

Source: https://www.bls.gov/data/#unemployment

Reading in unemployment rate data.

##Full Data Set

Make new factor variable that designates county urban-rural classification using NCHS classification scheme found in Flynt and Daepp Int J Health Geogr (2015) 14:25 DOI 10.1186/s12942-015-0017-5.

```
urban.rural.class <- factor(levels = c("Noncore", "Micropolitan", "Small Metro", "Medium Metro", "Large
for (i in 1:length(census$HD01)) {
    if (census$HD01[i] < 10000) {
        urban.rural.class[i] = "Noncore"
    } else if (census$HD01[i] >= 10000 & census$HD01[i] < 49999) {
        urban.rural.class[i] = "Micropolitan"
    } else if (census$HD01[i] >= 50000 & census$HD01[i] < 250000) {
        urban.rural.class[i] = "Small Metro"
    } else if (census$HD01[i] >= 250000 & census$HD01[i] < 999999) {
        urban.rural.class[i] = "Medium Metro"
    } else {
        urban.rural.class[i] = "Large Metro"
    }
}
```

Load in shape files for PA counties. Source: https://www.pasda.psu.edu/uci/DataSummary.aspx?dataset=24

```
library(rgdal)
```

```
## Loading required package: sp
```

```
## rgdal: version: 1.4-4, (SVN revision 833)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.2.3, released 2017/11/20
##  Path to GDAL shared files: C:/Users/Claudia/Documents/R/win-library/3.6/rgdal/gdal
##  GDAL binary built with GEOS: TRUE
##  Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
##  Path to PROJ.4 shared files: C:/Users/Claudia/Documents/R/win-library/3.6/rgdal/proj
##  Linking to sp version: 1.3-1
```

```
PaCounty <- readOGR(dsn=path.expand("~/BU/Research/Full Data/drive-download-20190705T172729Z-001"), lay
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\Claudia\Documents\BU\Research\Full Data\drive-download-20190705T172729Z-001", laye
## with 67 features
## It has 19 fields
## Integer64 fields read as strings:  COUNTY_N_1
```

coord_fixed() scales the x and y axes so if we change the outer dimensions of the plot the aspect ratio remains unchanged.

```
county_map <- map_data(PaCounty)
```

```
## Warning in SpatialPolygons2map(database, namefield = namefield): database
## does not (uniquely) contain the field 'name'.
```

```
county_data <- PaCounty@data
county_map$region <- factor(county_map$region)
```

Combine data sets into one fullData set. We will make region into a factor for the purpose of inner joining fullData with the shape files to make a together data set. We also add an SES index variable using the sum of the scaled variables: percent of adults with less than a HS degree, percent of households headed by single females, percent of nonwhite county residents, and poverty rate. We then make the together data set by inner joining county_map and fullData by region.

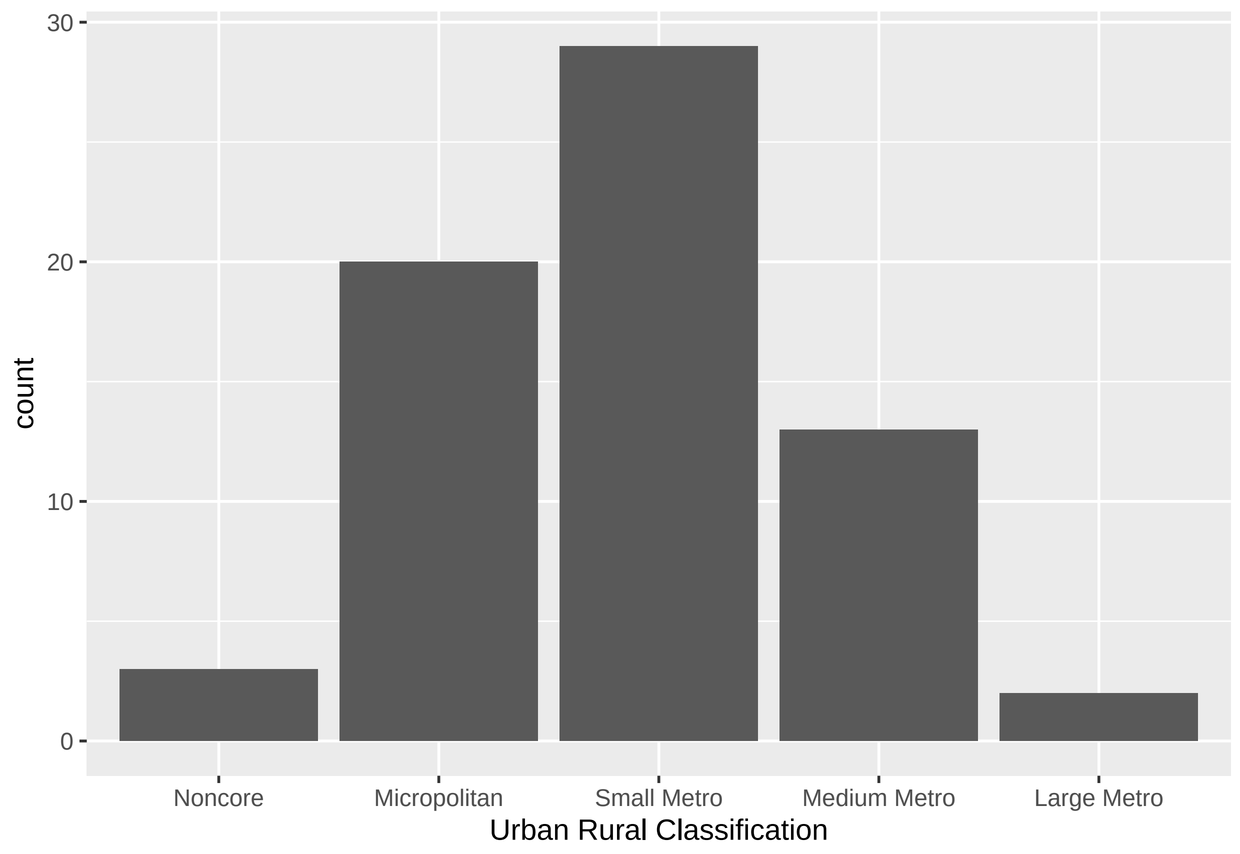
```
fullData <- NULL
together <- NULL
fullData <- fea.access.pa[,1:3]

tempRegion <- vector()
for (i in 1:nrow(fullData)) {
  number <- match(tolower(fullData$County[i]), tolower(county_data$COUNTY_NAM))
  tempRegion[i] <- number
}
region <- factor(tempRegion, labels = 0:66)
fullData <- cbind(fullData, region)
fullData <- cbind(fullData, urban.rural.class)
fullData <- cbind(fullData, census[,8:14])
fullData <- cbind(fullData, acs[,222])
fullData <- cbind(fullData, fea.access.pa[,4:44])
fullData <- cbind(fullData, fea.assistance.pa[,c(4,5,6,10,11,12,26,27,28,29,36,37,38,39,40,41)])
fullData <- cbind(fullData, fea.health.pa[,c(4,5,6,7,9,10,11,12,13,14)])
fullData <- cbind(fullData, fea.local.pa[,4:50])
fullData <- cbind(fullData, fea.restaurants.pa[,4:15])
fullData <- cbind(fullData, fea.socioeconomic.pa[,4:18])
fullData <- cbind(fullData, fea.stores.pa[,4:39])
fullData <- cbind(fullData, feedAmerica.pa[,4:18])
fullData <- cbind(fullData, temp)
fullData <- cbind(fullData, unemployRate)
sesData <- fullData[,c(13,140,149,206)]
pct_nonWhite <- 100-sesData$PCT_NHWHITE10
sesData <- cbind(sesData,pct_nonWhite)
sesData <- sesData[,-2]
sesData <- scale(sesData)
SES <- vector()
for (i in 1:nrow(sesData)) {
  SES[i] <- sum(sesData[i,c(1,2,3,4)])
}
fullData <- cbind(fullData, SES)

together <- inner_join(county_map, fullData, by = "region")
```

```
## Warning: Column `region` joining factors with different levels, coercing to
## character vector
```

#Univariate EDA

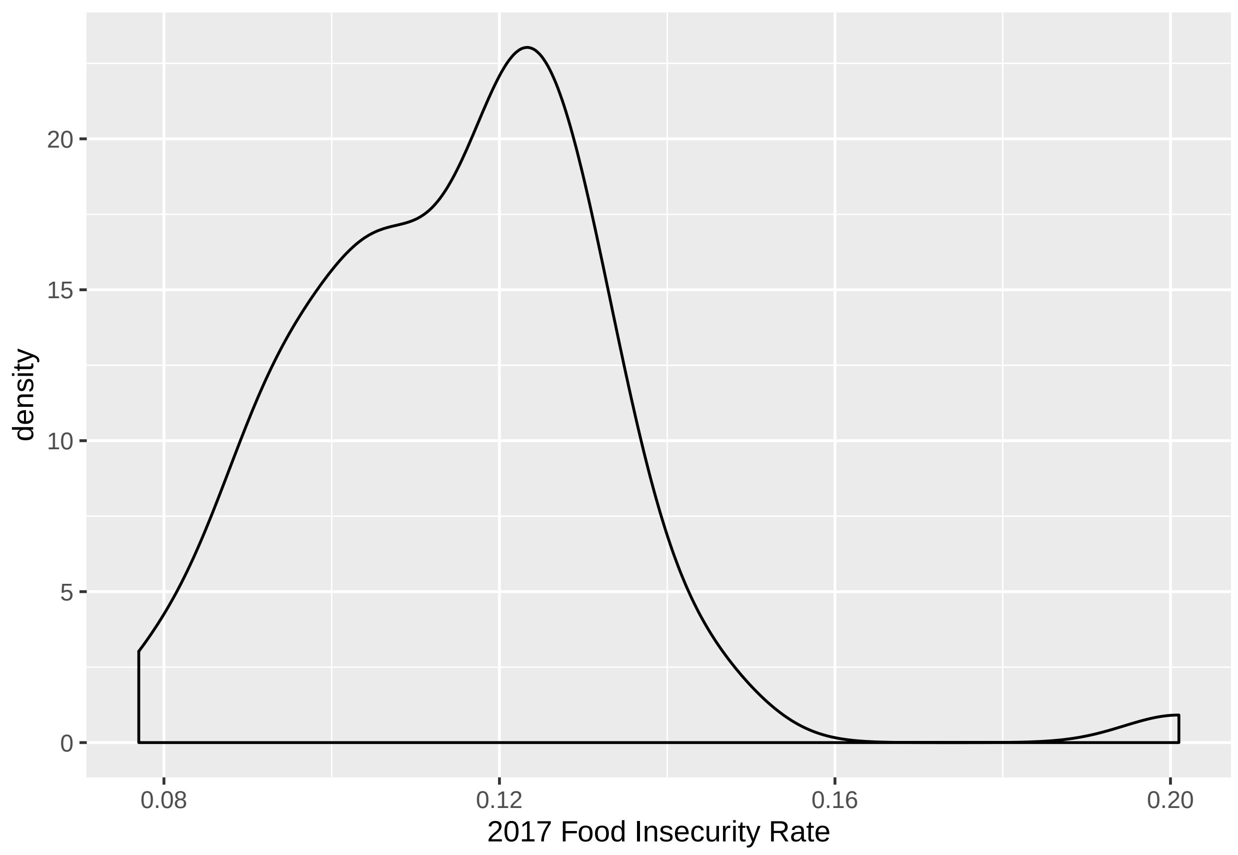
###Urban/Rural Classification Barchart showing the distribution of urban/rural counties in PA. We see that small metro contains the highest number of counties, while large metro contains the smallest number (Allegheny and Philadelphia), with noncore following as the next smallest.

```r
ggplot(fullData) + geom_bar(aes(x= urban.rural.class)) + xlab("Urban Rural Classification")
```

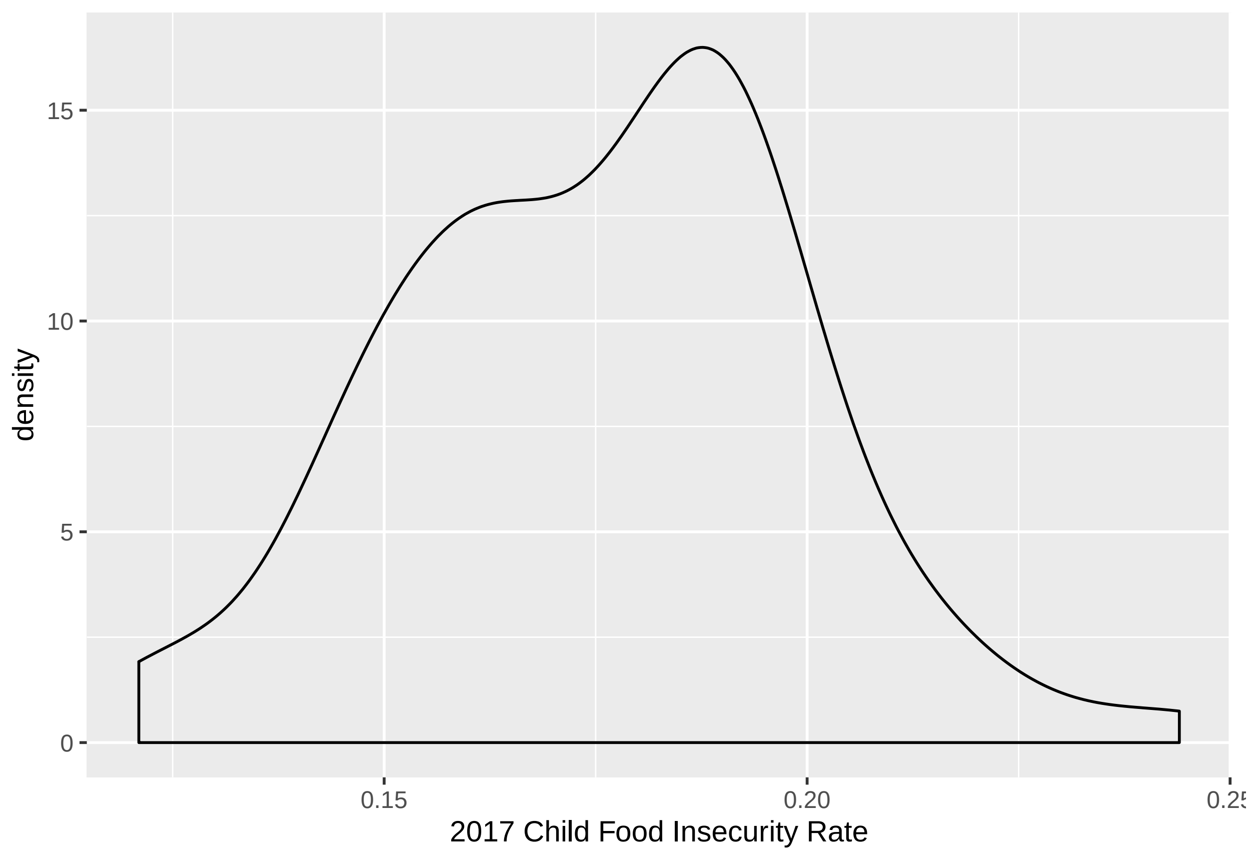


###Food Insecurity Rates Density plots examining variables associated with Food Insecurity rate.

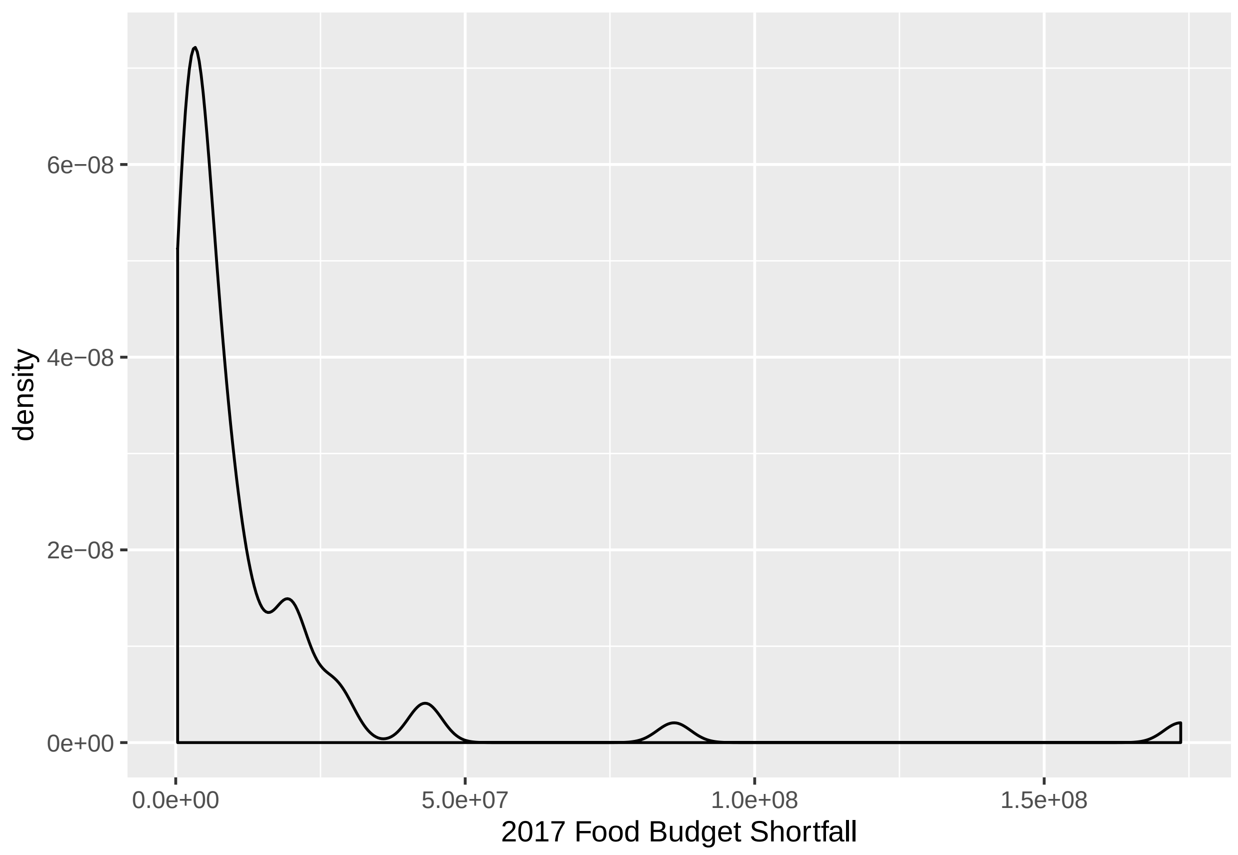
```r
ggplot(fullData) + geom_density(aes(x=X2017.Food.Insecurity.Rate)) + xlab("2017 Food Insecurity Rate")
```



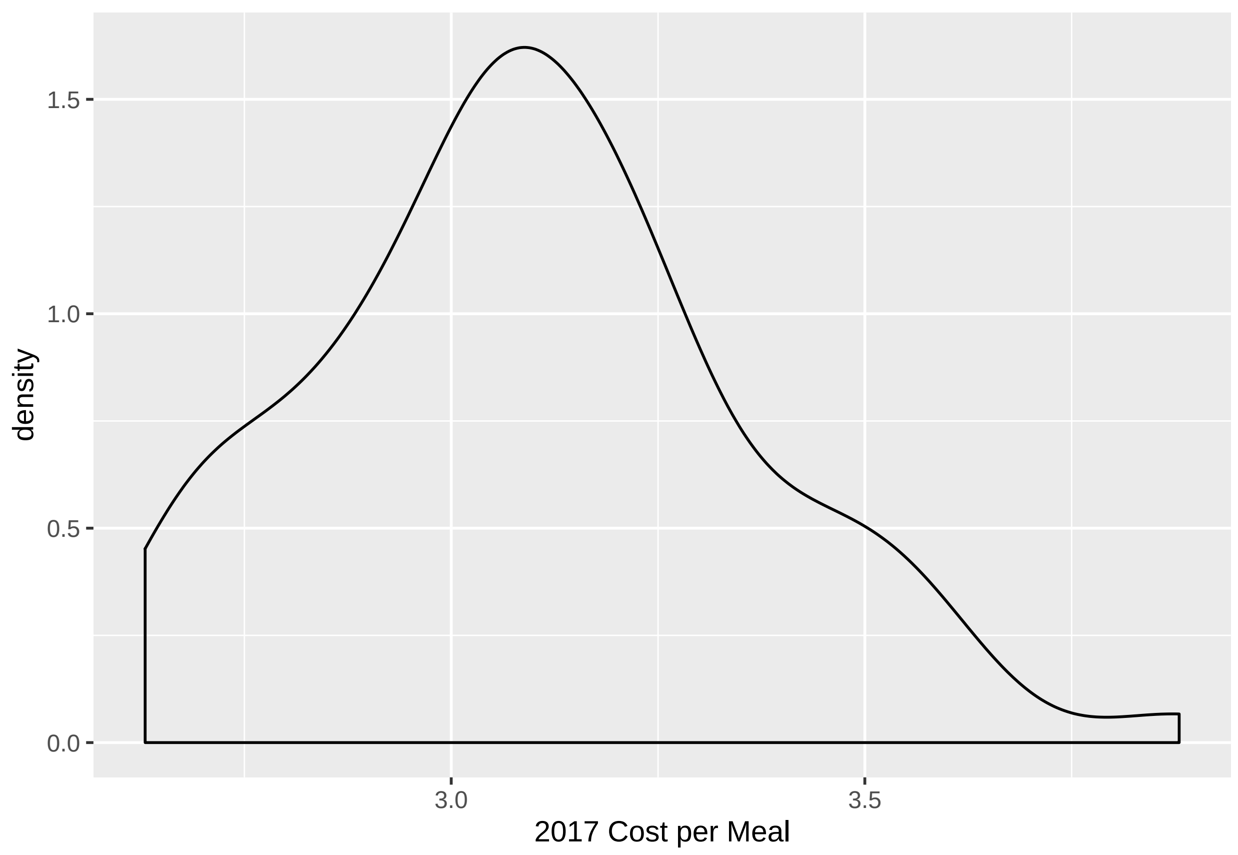
```
ggplot(fullData) + geom_density(aes(x=X2017.Child.food.insecurity.rate)) + xlab("2017 Child Food Insecu
```



```
ggplot(fullData) + geom_density(aes(x=X2017.Weighted.Annual.Food.Budget.Shortfall)) + xlab("2017 Food Bu
```



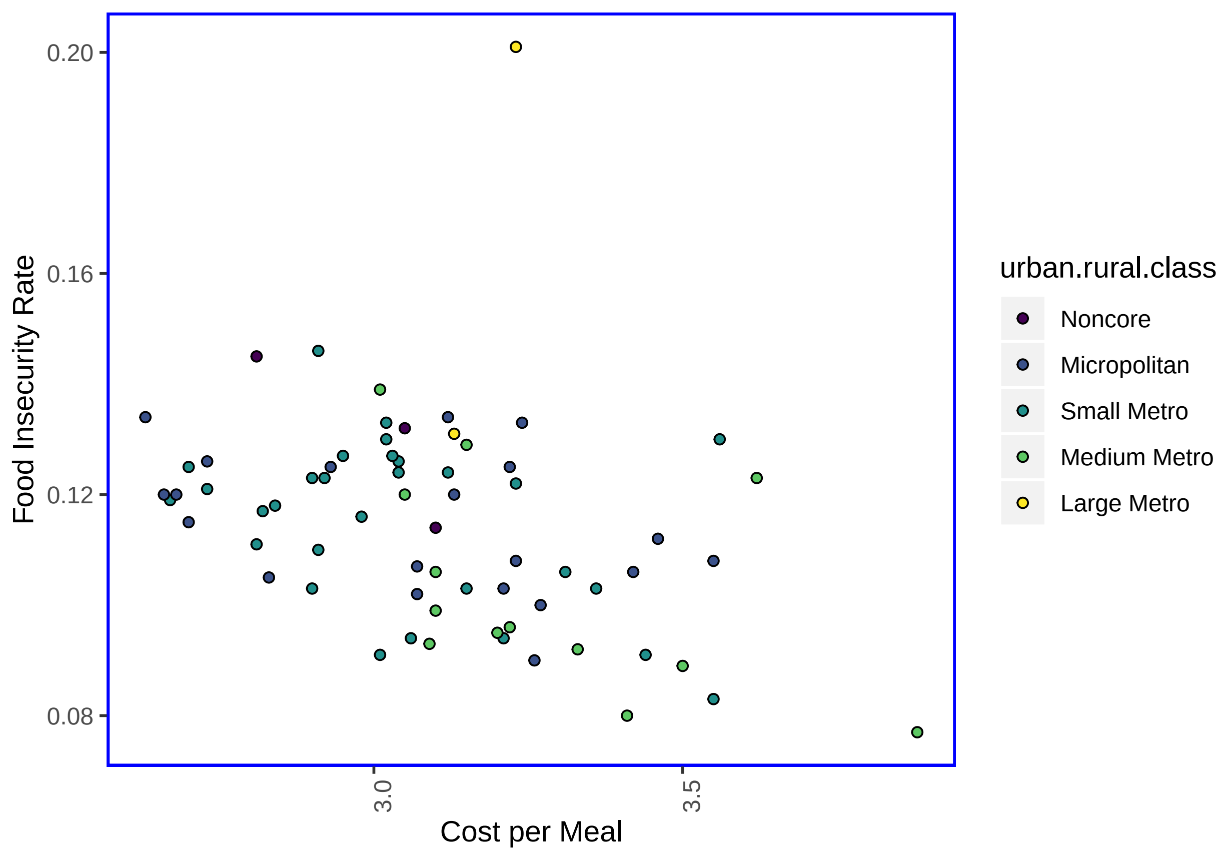
```r
ggplot(fullData) + geom_density(aes(x=X2017.Cost.Per.Meal)) + xlab("2017 Cost per Meal")
```



#Bivariate EDA

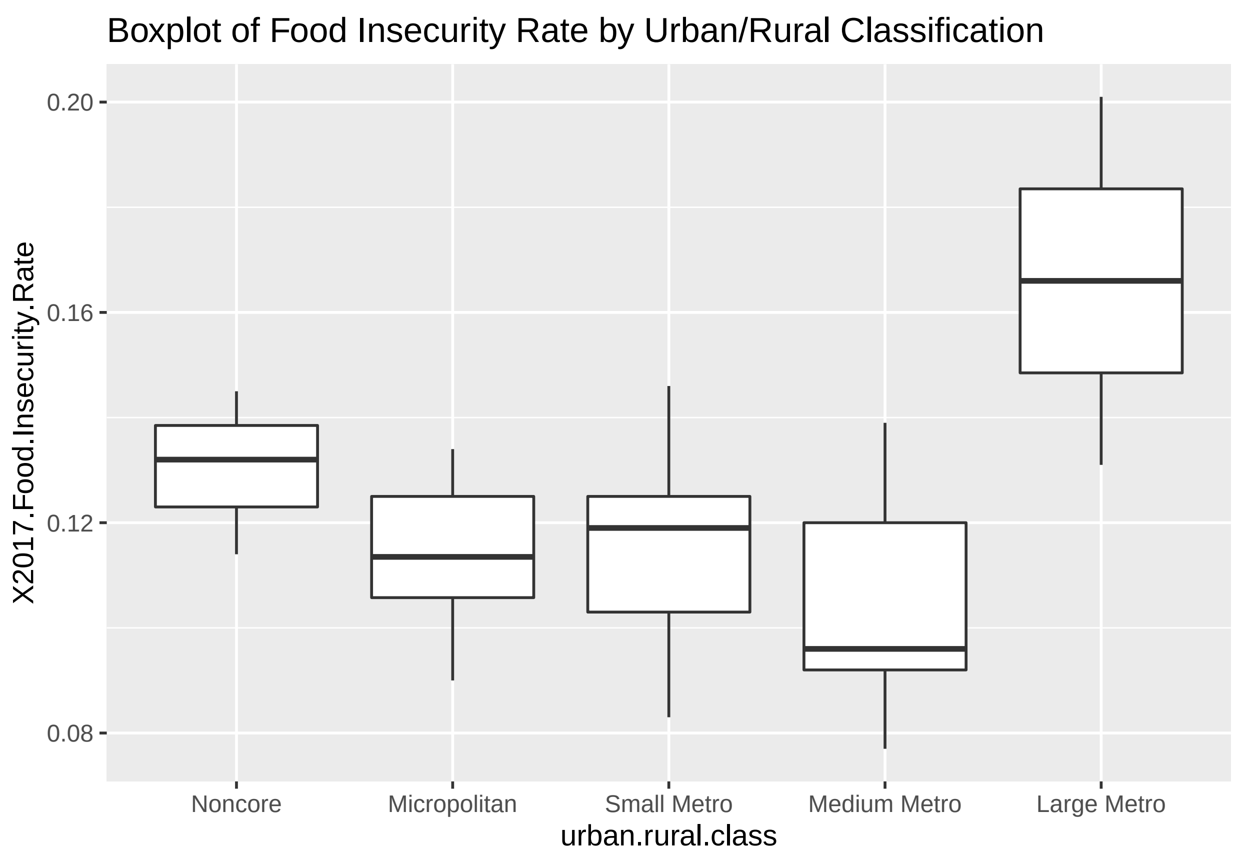
Scatterplot of food insecurity rate vs cost per meal colored by urban/rural classification.

```
ggplot(fullData, aes(x = X2017.Cost.Per.Meal, y = X2017.Food.Insecurity.Rate)) + geom_point(aes(fill=url
```



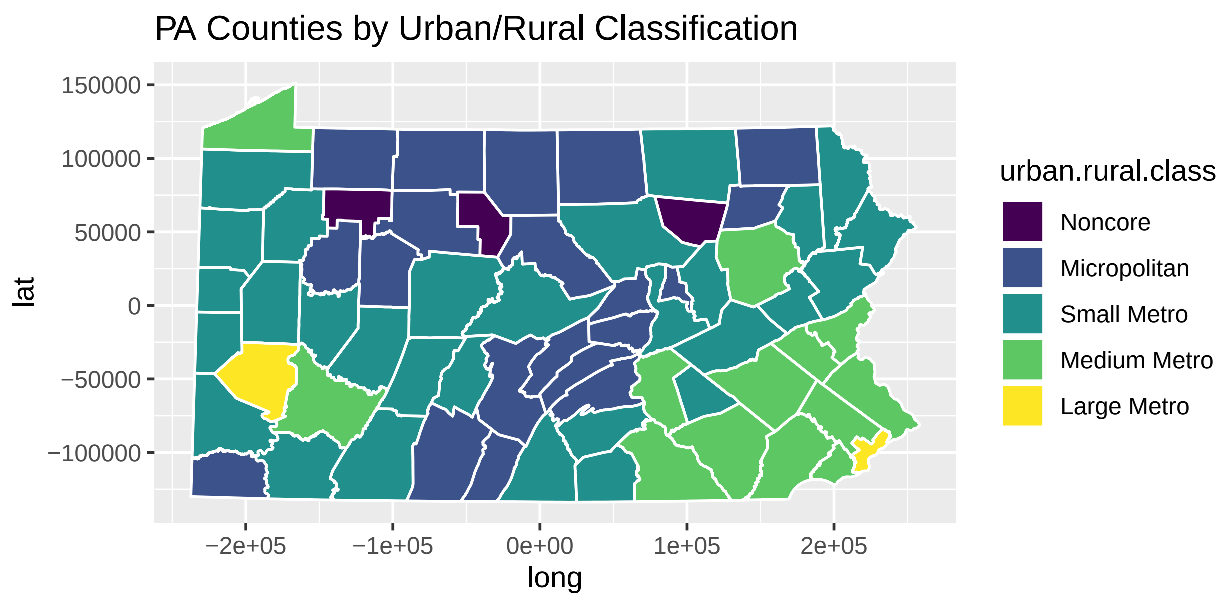
Boxplot of food insecurity rate for each urban/rural classification. We can see that large metro has the highest overall food insecurity rate with its first quartile higher than the maximum of noncore, the second highest food insecurity rate overall. We can see that medium metro has the lowest median food insecurity rate, and that micropolitan and small metro have similar insecurity distributions.

```
ggplot(fullData) + geom_boxplot(aes(x=urban.rural.class, y=X2017.Food.Insecurity.Rate)) + ggtitle("Boxpl
```

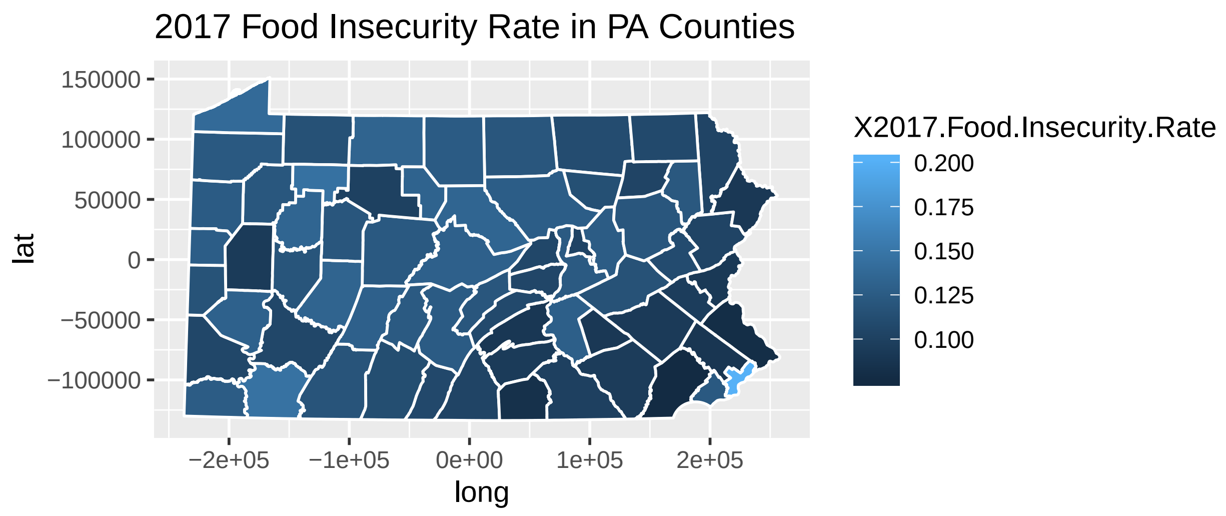


#Mapping Variables

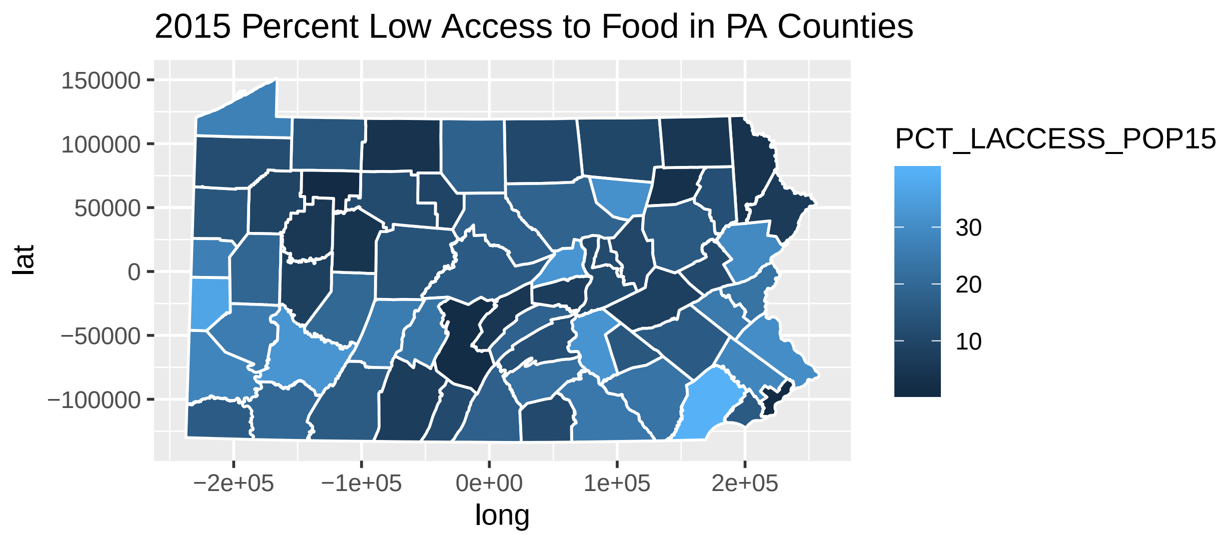
```
ggplot() + geom_polygon(data = together, aes(x = long, y = lat, fill = urban.rural.class, group=group),
```



```
#X2017.Food.Insecurity.Rate: the percentage of the population that experienced food insecurity at some
#Hint: Higher value = higher food insecurity
ggplot() + geom_polygon(data = together, aes(x = long, y = lat, fill = X2017.Food.Insecurity.Rate, grou
```



```
#PCT_LACCESS_POP15: Percentage of people in a county living more than 1 mile from a supermarket, superc
#Hint: Higher value = lower food access
ggplot() + geom_polygon(data = together, aes(x = long, y = lat, fill = PCT_LACCESS_POP15, group=group),
```



I can see that Philadelphia county has the highest rate of food insecurity by the first map, which is interesting because the second map shows Philadelphis having the best access to food. Looking at the 3 noncore regions, I can see that they each have a middle range (.125-.175) food insecurity rate, but one has relatively good food access, one has medium food access, and the other has low food access. In fact, the county with the lowest food insecurity of the noncore counties has the lowest access to food, and vice versa. This demonstrates that proximity to a grocery store is not a direct indication of food insecurity. In fact, by looking at the maps and the correlation, there appears to be a negative relationship between the two.

*The rm(list = ls()) code commented throughout is useful if needing to "chunk output in console" for making maps. This code is not necessary if just running the code in line.

#Feeding America Variable Clustering

```
#rm(list = ls())
```

Reset fullData and together for each new cluster.

Grab poverty rate, median income, percent hispanic, percent black, and unemployment rate variables. Make data set out of these variables (used in Feeding America food insecurity variable) from the fullData.

###Spatial Clustering

Source for spatial agglomerative clustering algorithm courtesy of Craig Anderson, The University of Glasgow.

```
n.prior <- ncol(data)
###Creating the storage matrices###
cluster.store <- matrix(rep(0,nrow(data)*nrow(data)), ncol=nrow(data))
###Putting row numbers in the first row of storage matrix
cluster.store[1,] <- 1:nrow(data)
```

```r
###Making a vector of 1's that is the same size as the number of rows in the data to indicate cluster s
cluster.size <- rep(1,nrow(data))
###Making a vector of 0's that is the one less than the number of rows in the data for the dissimilar s
dissim.size <- rep(0,nrow(data)-1)
###Making a matrix of 2 columns of 0's. Used for when merging clusters.
merged <- matrix(rep(0,2*(nrow(data)-1)),ncol=2)
###Makes negative values for the column numbers in cluster.store to use as indices for merging
merge.ind <- -cluster.store[1,]
###the initial matrix of county neighbors
initial.W <- W

update.data <- data
```

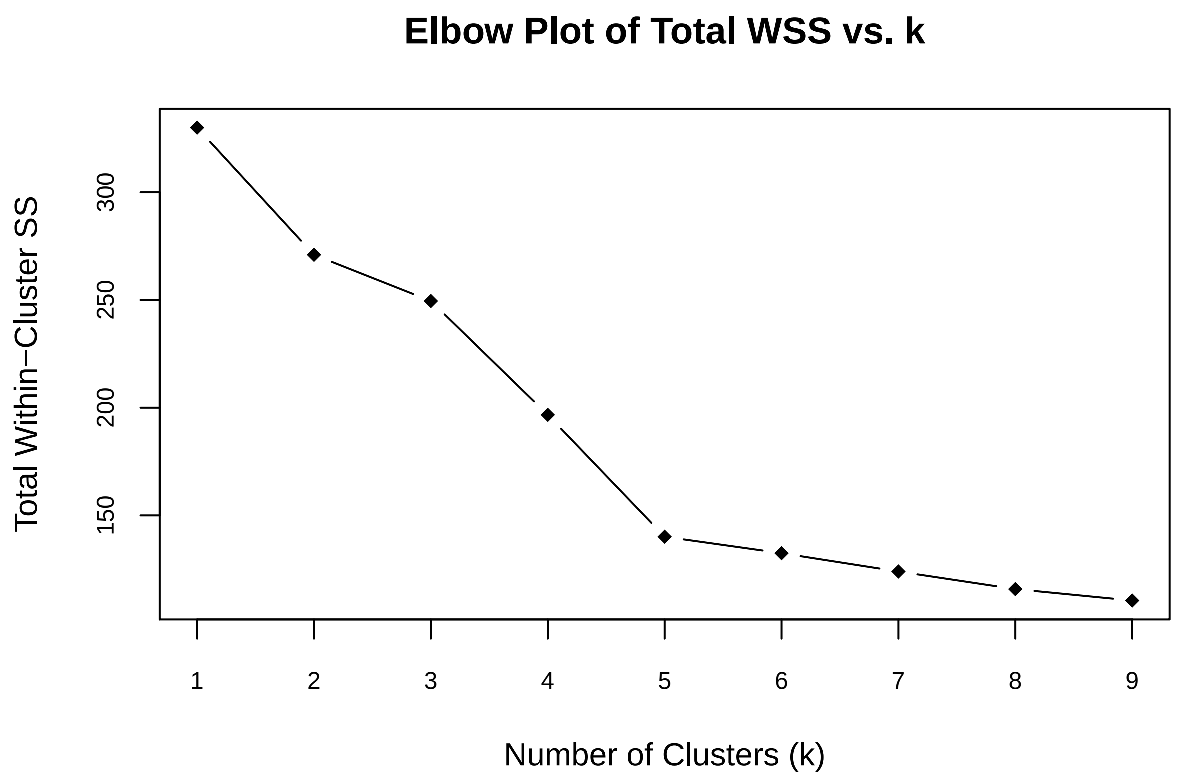
```r
  ###Loop##

for (i in 1:(nrow(data)-1)){

###Count cluster size###

    for (j in 1:nrow(data)){
      cluster.size[j] <- sum(cluster.store[i,]==j)
    }
    cluster.size


###Induce cluster structure in dissimilarity matrix###
    clusts <- cluster.store[i,][cluster.size>1]
    clusts

    if(length(clusts)>0){
      for (j in 1:length(clusts)){
        num.row <- sum(cluster.store[i,]==clusts[j])
        update.data[cluster.store[i,]==clusts[j],] <- matrix(rep(apply(as.matrix(data[cluster.store[i,]=
      }
    }


    ###distance matrix to indicate how similar county clusters are
    sim.mat <- as.matrix(dist(update.data, diag=TRUE, upper=TRUE))

    ######
    sim.mat[lower.tri(sim.mat)] <- max(sim.mat)+1
    sim.mat[W==0] <- max(sim.mat)
    diag(sim.mat) <- max(sim.mat)

    ###Ensure points in same cluster aren't compared###
    #########
    clust.mat <- as.matrix(dist(cbind(cluster.store[i,], cluster.store[i,]), method="maximum", diag=TRUE
    sim.mat[clust.mat==0] <- max(sim.mat)

    ###Find most similar###
    similar <- which(sim.mat==min(sim.mat), arr.ind=TRUE)
    choice <- sample(x=1:nrow(similar), size=1)
    join.1 <- cluster.store[i,similar[choice,1]]
    join.2 <- cluster.store[i,similar[choice,2]]


    ###Store results###
    cluster.store[(i+1),] <- cluster.store[i,]
    cluster.store[(i+1),][cluster.store[(i+1),]==join.1] <- min(join.1,join.2)
    cluster.store[(i+1),][cluster.store[(i+1),]==join.2] <- min(join.1,join.2)

    dissim.size[i] <- min(sim.mat)
    merged[i,] <- c(merge.ind[similar[choice,1]], merge.ind[similar[choice,2]])
    old.merge.ind <- merged[i,]
    merge.ind[merge.ind==old.merge.ind[1]] <- i
    merge.ind[merge.ind==old.merge.ind[2]] <- i



    ###Update the W matrix ###
    new.W.row <- apply((W[cluster.store[(i+1),]==min(join.1,join.2),]),2,sum)
    num.replaced <- nrow(W[cluster.store[(i+1),]==min(join.1,join.2),])
    W[cluster.store[(i+1),]==min(join.1,join.2),] <- matrix(rep(new.W.row,num.replaced),nrow=num.replac
    W[,cluster.store[(i+1),]==min(join.1,join.2)] <- matrix(rep(new.W.row,num.replaced),ncol=num.replac

    W[W>0] <- 1
  }



  W.list <- vector("list", nrow(data))
  for (i in 1:nrow(data)){
    W.list[[i]] <- matrix(data = NA, nrow = nrow(data), ncol = nrow(data),
                          byrow= FALSE, dimnames = NULL)
  }


  for(i in 1:nrow(data)){
    clust.mat <- as.matrix(dist(cbind(cluster.store[i,], cluster.store[i,]), method="maximum", diag=TRU
    W.list[[i]] <- initial.W
    W.list[[i]][clust.mat>0] <- 0
  }

result <- list(W.list=W.list, cluster.store=cluster.store, merge.ind=merge.ind, dissim.size=dissim.size
```

```
# Calculates the squared distance between two vectors x and y
sq.dist = function(x, y){
  return(sum((x-y)^2))
}

# Calculate the WSS for a single cluster
wss.cluster = function(clustermat){
  temp = apply(clustermat, 2,  mean)
  return(sum(apply(clustermat, 1,  function(row){sq.dist(row, temp)})))
}
```

```r
# Compute the Total WSS across all the clusters.
wss.total = function(data, labels){
  wss.tot = 0
  k = length(unique(labels))
  for(i in 1:k){
    wss.tot = wss.tot + wss.cluster(subset(data, labels == i))
  }
  return(wss.tot)
}
```

```r
see <- result[["cluster.store"]]
hc.wss = NULL
for (k in 59:67) {
    hc.wss[k] = wss.total(data, see[k,])
}
```

```r
plot(1:9, hc.wss[67:59], type = "b", pch = 18, xlab = "Number of Clusters (k)", ylab = "Total Within-Cl
axis(1, 1:9, 1:9, cex.axis = .75)
```



We can see that the elbow is at 5 clusters, which means we will use row 63 from the cluster result.

```r
table(see[63,])
```

```
##
```

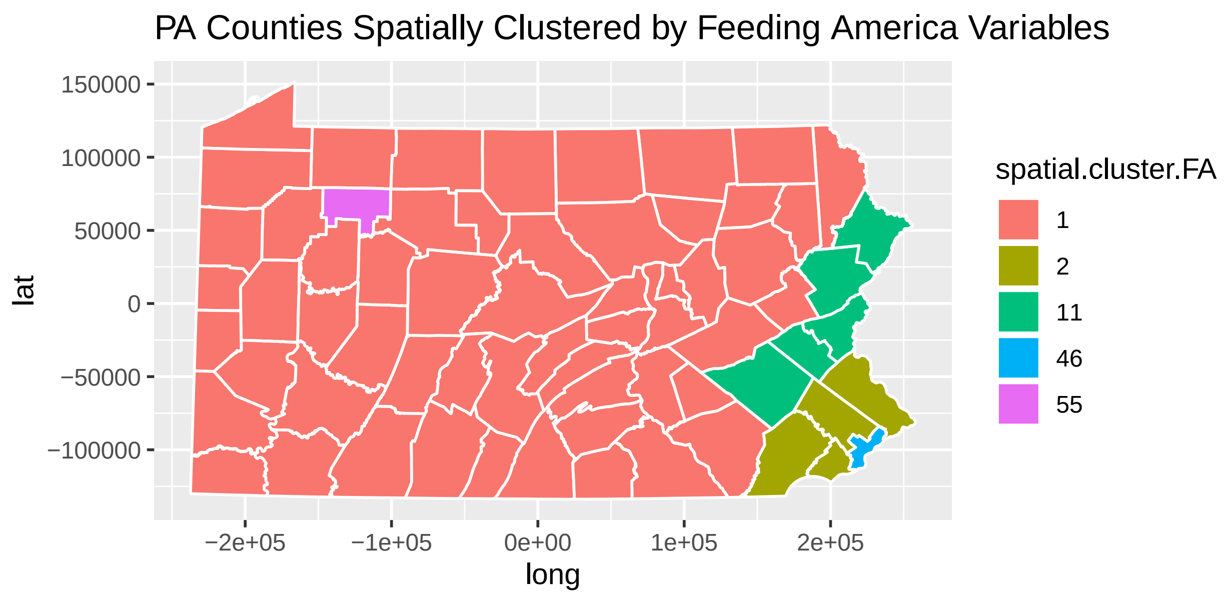
```
##  1  2 11 46 55
## 56  4  5  1  1
```

We add the correct clustered row into the data set. Thus, we can match and inner join with the together data set to map the clusters.

```
t <- PaCounty@data[1]
t.vec <- as.vector(t[1])
spatial.cluster.FA <- factor(see[63,])
cluster.feed.america <- cbind(t.vec, spatial.cluster.FA)

region <- match(tolower(cluster.feed.america$COUNTY_NAM), tolower(together$County))

region <- factor(region, labels = 0:66)
cluster.feed.america <- cbind(cluster.feed.america, region)
cluster.feed.america <- cluster.feed.america[,2:3]
together <- inner_join(together, cluster.feed.america, by = "region")
```

```
## Warning: Column `region` joining character vector and factor, coercing into
## character vector
```

```
ggplot() + geom_polygon(data = together, aes(x = long, y = lat, fill = spatial.cluster.FA, group=group)
```

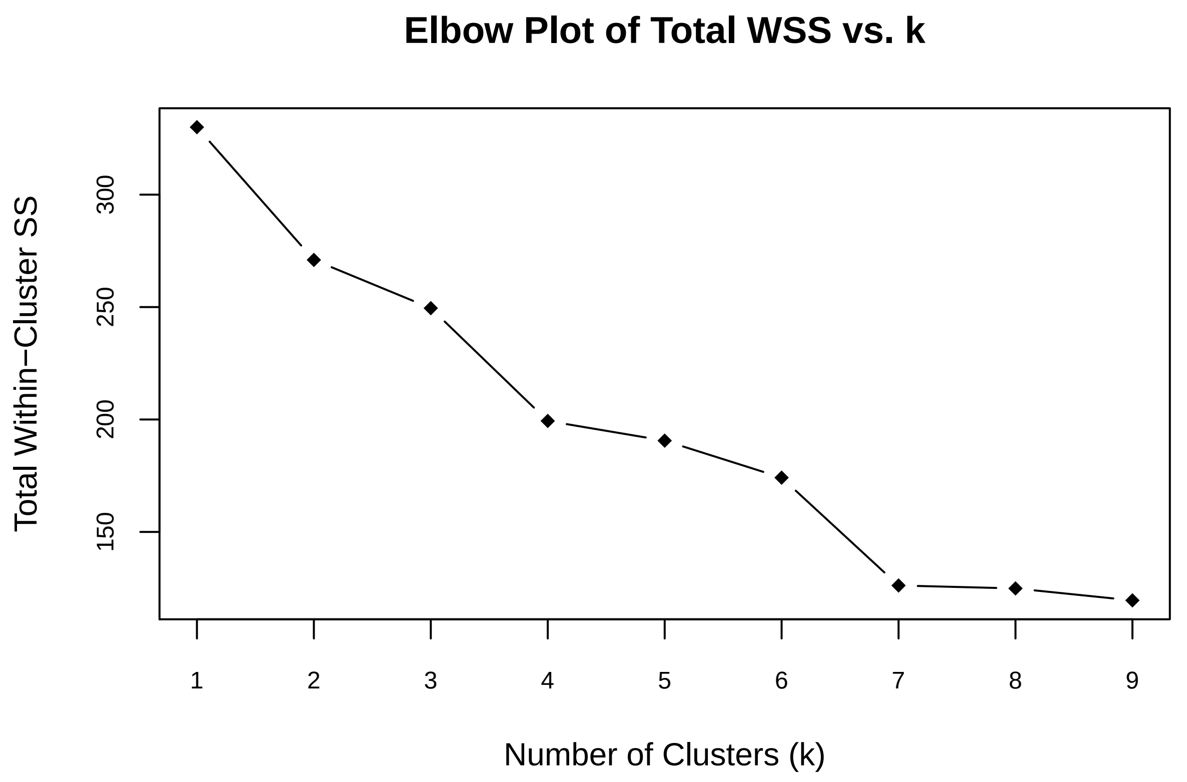


###Centroid Hierarchical Clustering

Reset fullData and together for each new cluster.

Source for centroid hierarchical clustering code courtesty of Professor Abby Flynt, Bucknell Univeristy.

```
hc.centroid = hclust(dist(data), method="centroid")
hc.wss = NULL
for (k in 1:9) {
    hc.wss[k] = wss.total(data, cutree(hc.centroid, k))
}

plot(1:9, hc.wss, type = "b", pch = 18, xlab = "Number of Clusters (k)", ylab = "Total Within-Cluster SS
axis(1, 1:9, 1:9, cex.axis = .75)
```

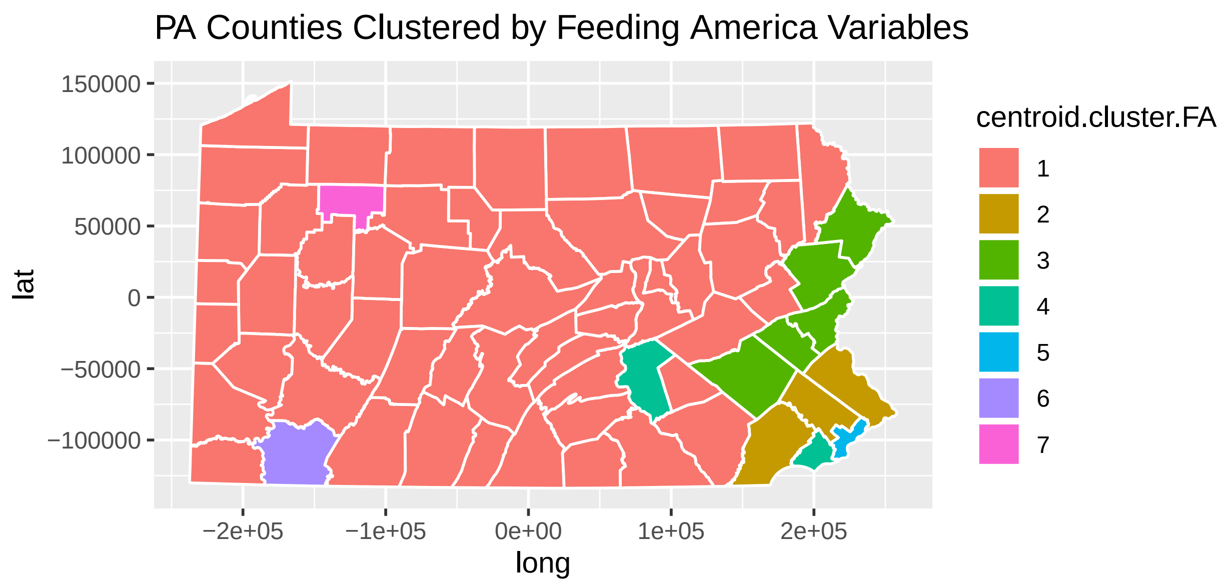


We can see that the elbow is at 7 clusters. We cut the centroid tree at 7 and add this to our data set for mapping.

```
t <- PaCounty@data[1]
t.vec <- as.vector(t[1])
centroid.cluster <- cutree(hc.centroid, 7)
centroid.cluster.FA <- factor(centroid.cluster)
cluster.feed.america <- cbind(t.vec, centroid.cluster.FA)

region <- match(tolower(cluster.feed.america$COUNTY_NAM), tolower(together$County))

region <- factor(region, labels = 0:66)
cluster.feed.america <- cbind(cluster.feed.america, region)
cluster.feed.america <- cluster.feed.america[,2:3]
together <- inner_join(together, cluster.feed.america, by = "region")
```

```
## Warning: Column `region` joining character vector and factor, coercing into
## character vector
```

```
ggplot() + geom_polygon(data = together, aes(x = long, y = lat, fill = centroid.cluster.FA, group=group)
```



#Rodriguez Variable Clustering

Reset fullData and together for each new cluster.

Using the Rodriguez article variables, we will cluster by food cost, transportation, access to food services, and education. Unfortunately, we do not have a transportation variable relating to food, so we will instead use the average commuting time to work.

```
W <- as.matrix(read.csv("County Neighbors.csv"))

tmp = PaCounty@data[1]
tmp.vec = as.vector(tmp[[1]])
count.vec = as.vector(fullData$County)

data = fullData[match(tolower(tmp.vec), tolower(count.vec)), c(18,204,206,212)]
HS.Diploma <- 1-data$noHSDiploma
data = cbind(data, HS.Diploma)
data = data[,-3]

#used to analyze clusters and make summary tables
analyzeData = fullData[match(tolower(tmp.vec), tolower(count.vec)), c(3,18,204,206,212)]
HS.Diploma <- 1-analyzeData$noHSDiploma
```

```
analyzeData = cbind(analyzeData, HS.Diploma)
analyzeData = analyzeData[,-4]
analyzeDataOut = analyzeData[-c(2,11,12,17,30,46,51,55),]


data <- scale(data)
```

```
n.prior <- ncol(data)
###Creating the storage matrices###
cluster.store <- matrix(rep(0,nrow(data)*nrow(data)), ncol=nrow(data))
###Putting row numbers in the first row of storage matrix
cluster.store[1,] <- 1:nrow(data)
###Making a vector of 1's that is the same size as the number of rows in the data to indicate cluster s
cluster.size <- rep(1,nrow(data))
###Making a vector of 0's that is the one less than the number of rows in the data for the dissimilar s
dissim.size <- rep(0,nrow(data)-1)
###Making a matrix of 2 columns of 0's. Used for when merging clusters.
merged <- matrix(rep(0,2*(nrow(data)-1)),ncol=2)
###Makes negative values for the column numbers in cluster.store to use as indices for merging
merge.ind <- -cluster.store[1,]
###the initial matrix of county neighbors
initial.W <- W

update.data <- data
```

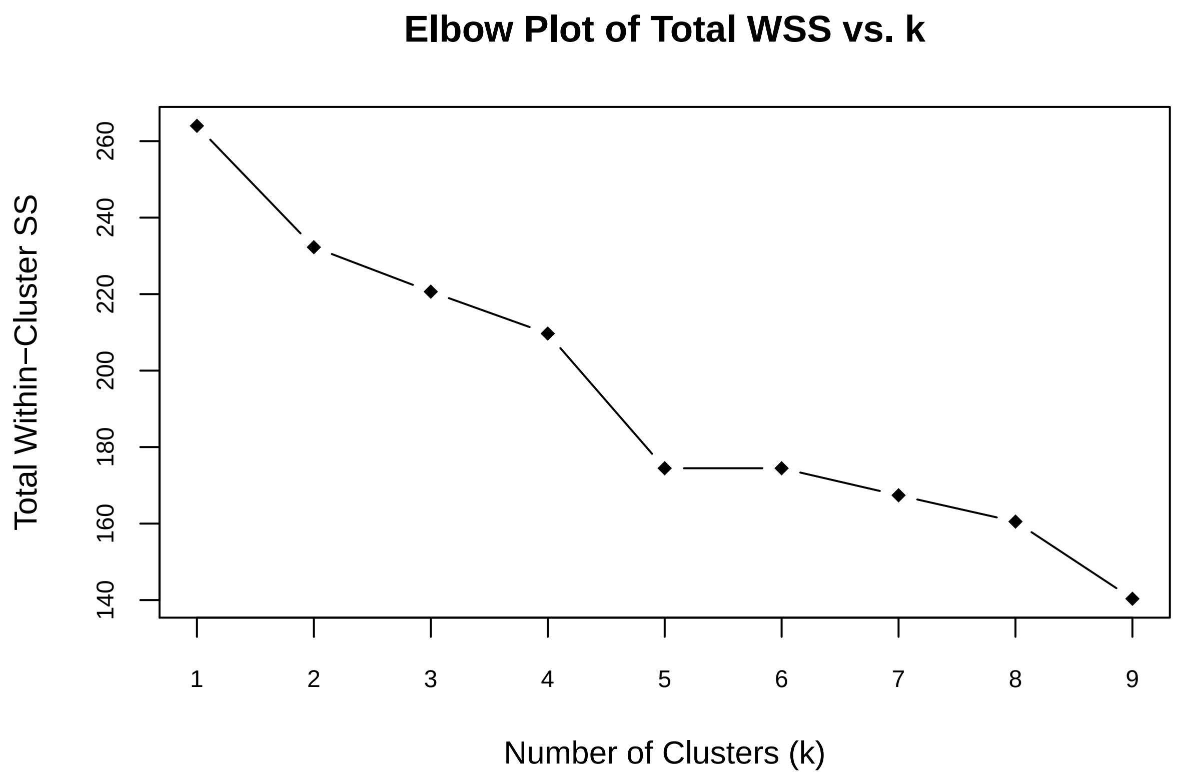
```
  ###Loop##

for (i in 1:(nrow(data)-1)){

###Count cluster size###

    for (j in 1:nrow(data)){
      cluster.size[j] <- sum(cluster.store[i,]==j)
    }
    cluster.size


###Induce cluster structure in dissimilarity matrix###
    clusts <- cluster.store[i,][cluster.size>1]
    clusts

    if(length(clusts)>0){
      for (j in 1:length(clusts)){
        num.row <- sum(cluster.store[i,]==clusts[j])
        update.data[cluster.store[i,]==clusts[j],] <- matrix(rep(apply(as.matrix(data[cluster.store[i,]=
      }
    }


    ###distance matrix to indicate how similar county clusters are
    sim.mat <- as.matrix(dist(update.data, diag=TRUE, upper=TRUE))

    ######
  sim.mat[lower.tri(sim.mat)] <- max(sim.mat)+1
  sim.mat[W==0] <- max(sim.mat)
  diag(sim.mat) <- max(sim.mat)

  ###Ensure points in same cluster aren't compared###
  #########
  clust.mat <- as.matrix(dist(cbind(cluster.store[i,], cluster.store[i,]), method="maximum", diag=TRU
  sim.mat[clust.mat==0] <- max(sim.mat)

  ###Find most similar###
  similar <- which(sim.mat==min(sim.mat), arr.ind=TRUE)
  choice <- sample(x=1:nrow(similar), size=1)
  join.1 <- cluster.store[i,similar[choice,1]]
  join.2 <- cluster.store[i,similar[choice,2]]


  ###Store results###
  cluster.store[(i+1),] <- cluster.store[i,]
  cluster.store[(i+1),][cluster.store[(i+1),]==join.1] <- min(join.1,join.2)
  cluster.store[(i+1),][cluster.store[(i+1),]==join.2] <- min(join.1,join.2)

  dissim.size[i] <- min(sim.mat)
  merged[i,] <- c(merge.ind[similar[choice,1]], merge.ind[similar[choice,2]])
  old.merge.ind <- merged[i,]
  merge.ind[merge.ind==old.merge.ind[1]] <- i
  merge.ind[merge.ind==old.merge.ind[2]] <- i



  ###Update the W matrix ###
  new.W.row <- apply((W[cluster.store[(i+1),]==min(join.1,join.2),]),2,sum)
  num.replaced <- nrow(W[cluster.store[(i+1),]==min(join.1,join.2),])
  W[cluster.store[(i+1),]==min(join.1,join.2),] <- matrix(rep(new.W.row,num.replaced),nrow=num.replac
  W[,cluster.store[(i+1),]==min(join.1,join.2)] <- matrix(rep(new.W.row,num.replaced),ncol=num.replac

  W[W>0] <- 1
}



W.list <- vector("list", nrow(data))
for (i in 1:nrow(data)){
  W.list[[i]] <- matrix(data = NA, nrow = nrow(data), ncol = nrow(data),
                        byrow= FALSE, dimnames = NULL)
}


for(i in 1:nrow(data)){
  clust.mat <- as.matrix(dist(cbind(cluster.store[i,], cluster.store[i,]), method="maximum", diag=TRU
  W.list[[i]] <- initial.W
  W.list[[i]][clust.mat>0] <- 0
}
```

```r
result <- list(W.list=W.list, cluster.store=cluster.store, merge.ind=merge.ind, dissim.size=dissim.size
```

```r
# Calculates the squared distance between two vectors x and y
sq.dist = function(x, y){
  return(sum((x-y)^2))
}

# Calculate the WSS for a single cluster
wss.cluster = function(clustermat){
  temp = apply(clustermat, 2,  mean)
  return(sum(apply(clustermat, 1,  function(row){sq.dist(row, temp)})))
}

# Compute the Total WSS across all the clusters.
wss.total = function(data, labels){
  wss.tot = 0
  k = length(unique(labels))
  for(i in 1:k){
    wss.tot = wss.tot + wss.cluster(subset(data, labels == i))
  }
  return(wss.tot)
}
```

```r
see <- result[["cluster.store"]]
hc.wss = NULL
for (k in 59:67) {
    hc.wss[k] = wss.total(data, see[k,])
}
```

```r
plot(1:9, hc.wss[67:59], type = "b", pch = 18, xlab = "Number of Clusters (k)", ylab = "Total Within-Cl
axis(1, 1:9, 1:9, cex.axis = .75)
```



We can see that the elbow is at 5 clusters.

We select row 65 from the clustering result and add this to our data set for mapping purposes.

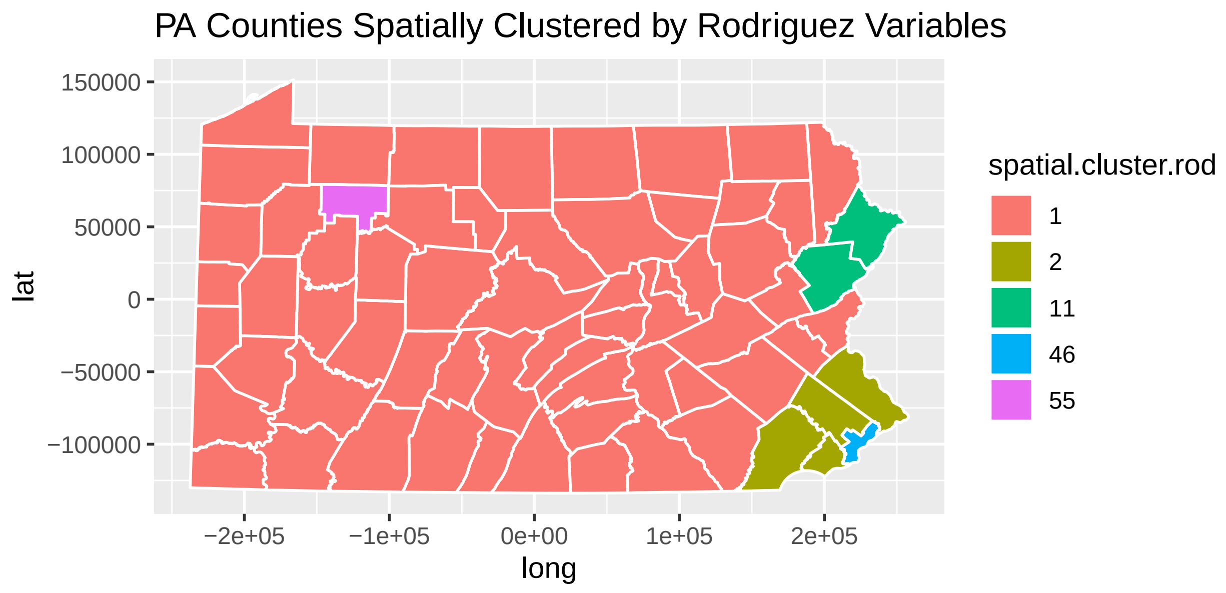
```
t <- PaCounty@data[1]
t.vec <- as.vector(t[1])
spatial.cluster.rod <- factor(see[63,])
cluster.rod <- cbind(t.vec, spatial.cluster.rod)

region <- match(tolower(cluster.rod$COUNTY_NAM), tolower(together$County))

region <- factor(region, labels = 0:66)
cluster.rod <- cbind(cluster.rod, region)
cluster.rod <- cluster.rod[,2:3]
together <- inner_join(together, cluster.rod, by = "region")
```

```
## Warning: Column `region` joining character vector and factor, coercing into
## character vector
```

```
table(see[63,])
```

```
##
##  1  2 11 46 55
## 59  4  2  1  1
```

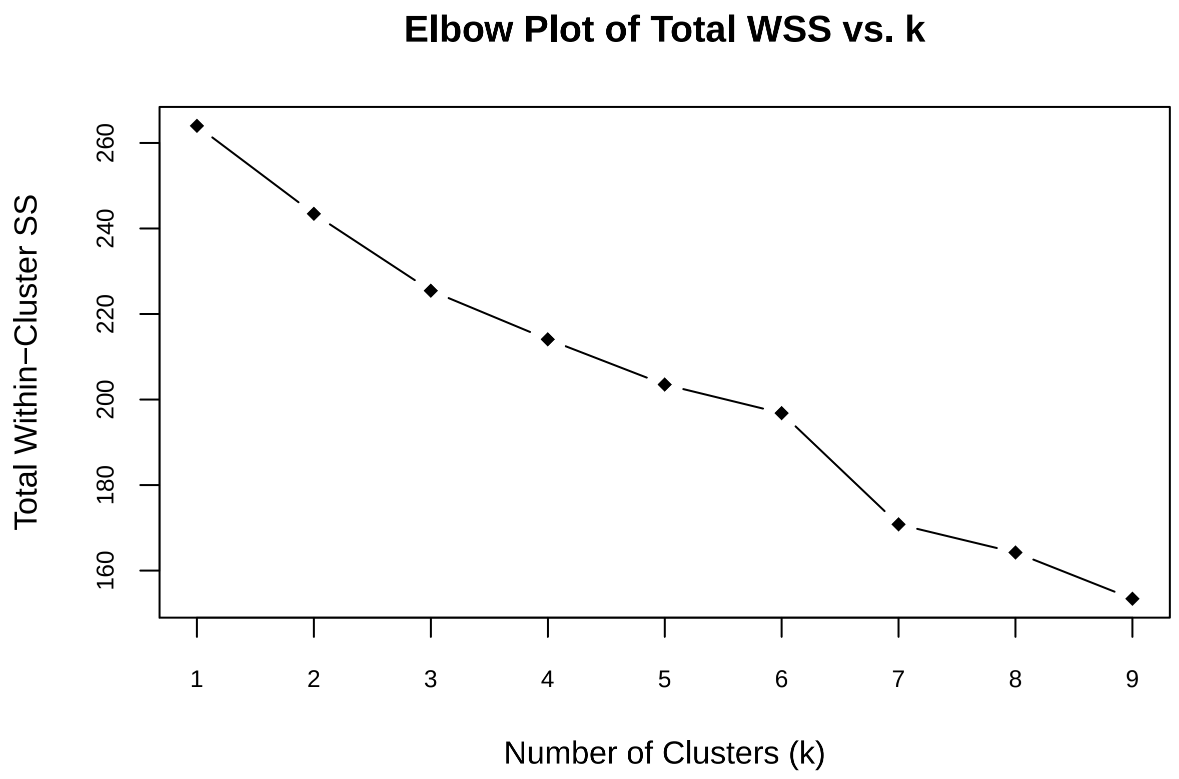
```
ggplot() + geom_polygon(data = together, aes(x = long, y = lat, fill = spatial.cluster.rod, group=group
```



###Centroid Hierarchical Clustering

Reset fullData and together for each new cluster.

```
hc.centroid = hclust(dist(data), method="centroid")
hc.wss = NULL
for (k in 1:9) {
    hc.wss[k] = wss.total(data, cutree(hc.centroid, k))
}

plot(1:9, hc.wss, type = "b", pch = 18, xlab = "Number of Clusters (k)", ylab = "Total Within-Cluster S
axis(1, 1:9, 1:9, cex.axis = .75)
```

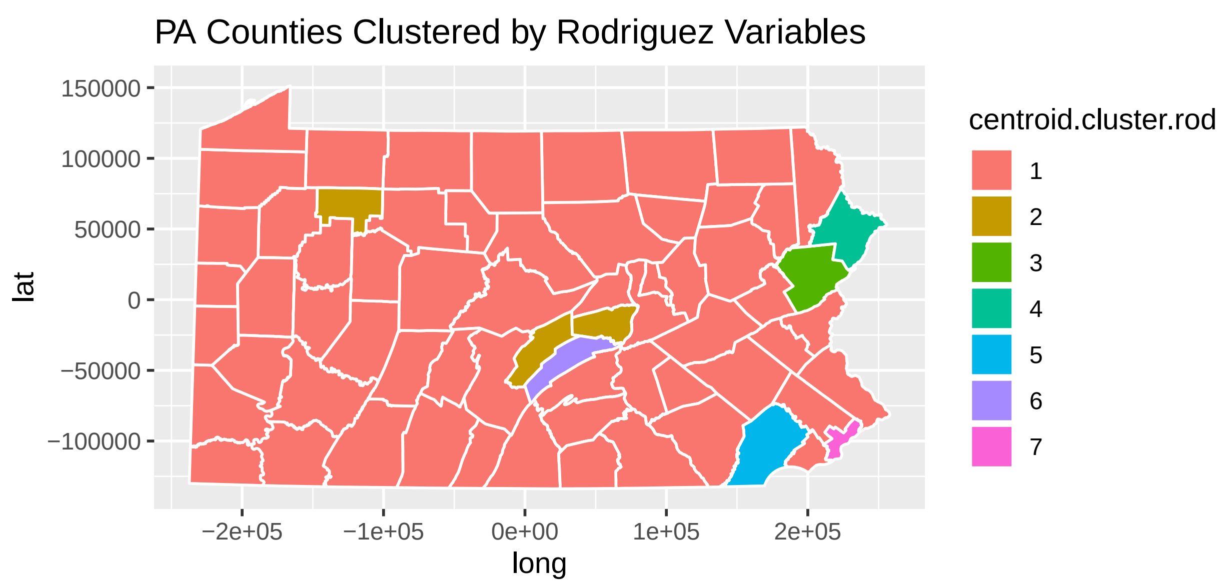


We can see that the elbow is at 7 clusters. We cut the tree at 7 and add this to our data set for mapping purposes.

```
t <- PaCounty@data[1]
t.vec <- as.vector(t[1])
centroid.cluster <- cutree(hc.centroid, 7)
centroid.cluster.rod <- factor(centroid.cluster)
cluster.rod <- cbind(t.vec, centroid.cluster.rod)

region <- match(tolower(cluster.rod$COUNTY_NAM), tolower(together$County))

region <- factor(region, labels = 0:66)
cluster.rod <- cbind(cluster.rod, region)
cluster.rod <- cluster.rod[,2:3]
together <- inner_join(together, cluster.rod, by = "region")
```

```
## Warning: Column `region` joining character vector and factor, coercing into
## character vector
```

```
ggplot() + geom_polygon(data = together, aes(x = long, y = lat, fill = centroid.cluster.rod, group=grou
```



#Flynt Variable Spatial Clustering

Reset fullData and together for each new cluster.

Using the Flynt variables, we will cluster by unemployment, median household income, pct obesity, pct diabetes, and SES index. The SES index is the sum of the scaled variables: percent of adults with less than a hs degree, percent of households headed by single females, percent of nonwhite county residents, and the poverty rate. This variable was already added to the data set when making the fullData in Full Data.Rmd.

```
W <- as.matrix(read.csv("County Neighbors.csv"))

tmp = PaCounty@data[1]
tmp.vec = as.vector(tmp[[1]])
count.vec = as.vector(fullData$County)

data = fullData[match(tolower(tmp.vec), tolower(count.vec)), c(72,74,220,221,211)]

#used to analyze clusters and make summary tables
analyzeData = fullData[match(tolower(tmp.vec), tolower(count.vec)), c(3,72,74,220,221,211)]
analyzeDataOut = analyzeData[-c(2,12,30,46,51,55),]

data <- scale(data)
```

```
n.prior <- ncol(data)
###Creating the storage matrices###
cluster.store <- matrix(rep(0,nrow(data)*nrow(data)), ncol=nrow(data))
###Putting row numbers in the first row of storage matrix
```

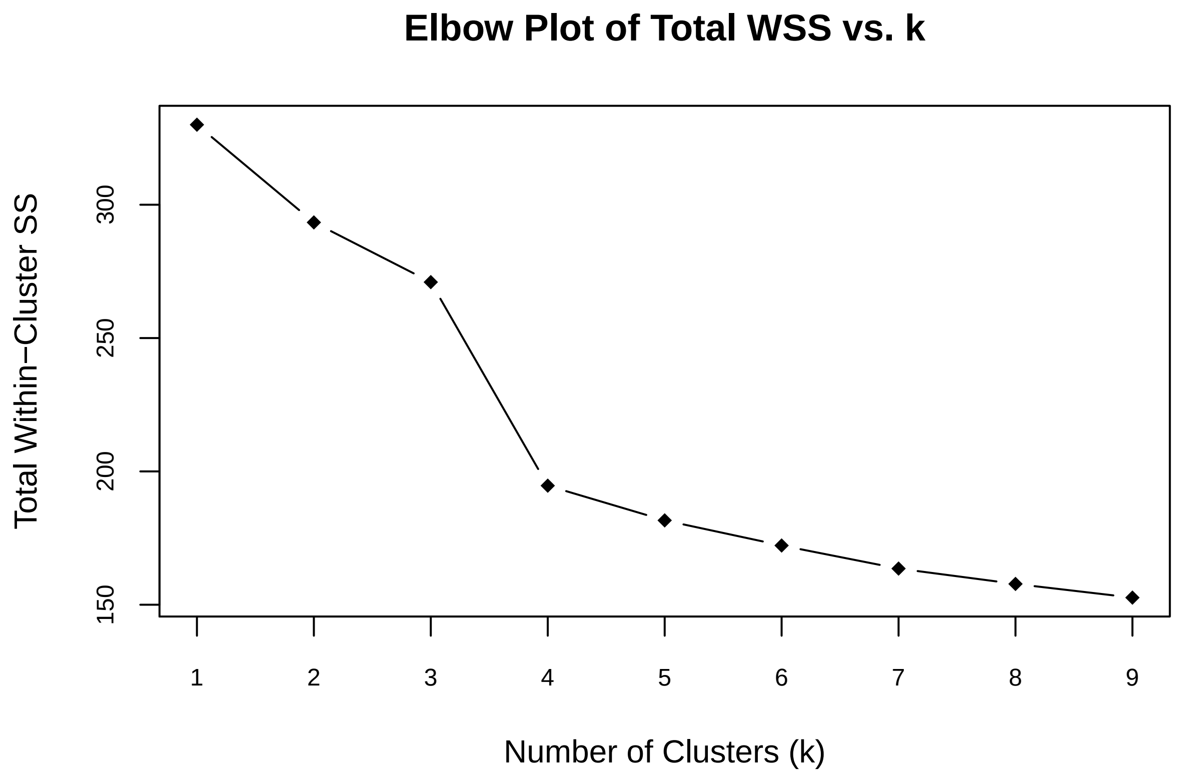
```
cluster.store[1,] <- 1:nrow(data)
###Making a vector of 1's that is the same size as the number of rows in the data to indicate cluster s
cluster.size <- rep(1,nrow(data))
###Making a vector of 0's that is the one less than the number of rows in the data for the dissimilar s
dissim.size <- rep(0,nrow(data)-1)
###Making a matrix of 2 columns of 0's. Used for when merging clusters.
merged <- matrix(rep(0,2*(nrow(data)-1)),ncol=2)
###Makes negative values for the column numbers in cluster.store to use as indices for merging
merge.ind <- -cluster.store[1,]
###the initial matrix of county neighbors
initial.W <- W

update.data <- data
```

```
  ###Loop##

for (i in 1:(nrow(data)-1)){

###Count cluster size###

    for (j in 1:nrow(data)){
      cluster.size[j] <- sum(cluster.store[i,]==j)
    }
    cluster.size


###Induce cluster structure in dissimilarity matrix###
    clusts <- cluster.store[i,][cluster.size>1]
    clusts

    if(length(clusts)>0){
      for (j in 1:length(clusts)){
        num.row <- sum(cluster.store[i,]==clusts[j])
        update.data[cluster.store[i,]==clusts[j],] <- matrix(rep(apply(as.matrix(data[cluster.store[i,]=
      }
    }


    ###distance matrix to indicate how similar county clusters are
    sim.mat <- as.matrix(dist(update.data, diag=TRUE, upper=TRUE))

    ######
    sim.mat[lower.tri(sim.mat)] <- max(sim.mat)+1
    sim.mat[W==0] <- max(sim.mat)
    diag(sim.mat) <- max(sim.mat)

    ###Ensure points in same cluster aren't compared###
    #########
    clust.mat <- as.matrix(dist(cbind(cluster.store[i,], cluster.store[i,]), method="maximum", diag=TRUE
    sim.mat[clust.mat==0] <- max(sim.mat)

    ###Find most similar###
    similar <- which(sim.mat==min(sim.mat), arr.ind=TRUE)
    choice <- sample(x=1:nrow(similar), size=1)
    join.1 <- cluster.store[i,similar[choice,1]]
    join.2 <- cluster.store[i,similar[choice,2]]


    ###Store results###
    cluster.store[(i+1),] <- cluster.store[i,]
    cluster.store[(i+1),][cluster.store[(i+1),]==join.1] <- min(join.1,join.2)
    cluster.store[(i+1),][cluster.store[(i+1),]==join.2] <- min(join.1,join.2)

    dissim.size[i] <- min(sim.mat)
    merged[i,] <- c(merge.ind[similar[choice,1]], merge.ind[similar[choice,2]])
    old.merge.ind <- merged[i,]
    merge.ind[merge.ind==old.merge.ind[1]] <- i
    merge.ind[merge.ind==old.merge.ind[2]] <- i



    ###Update the W matrix ###
    new.W.row <- apply((W[cluster.store[(i+1),]==min(join.1,join.2),]),2,sum)
    num.replaced <- nrow(W[cluster.store[(i+1),]==min(join.1,join.2),])
    W[cluster.store[(i+1),]==min(join.1,join.2),] <- matrix(rep(new.W.row,num.replaced),nrow=num.replac
    W[,cluster.store[(i+1),]==min(join.1,join.2)] <- matrix(rep(new.W.row,num.replaced),ncol=num.replac

    W[W>0] <- 1
  }



  W.list <- vector("list", nrow(data))
  for (i in 1:nrow(data)){
    W.list[[i]] <- matrix(data = NA, nrow = nrow(data), ncol = nrow(data),
                          byrow= FALSE, dimnames = NULL)
  }


  for(i in 1:nrow(data)){
    clust.mat <- as.matrix(dist(cbind(cluster.store[i,], cluster.store[i,]), method="maximum", diag=TRUE
    W.list[[i]] <- initial.W
    W.list[[i]][clust.mat>0] <- 0
  }

result <- list(W.list=W.list, cluster.store=cluster.store, merge.ind=merge.ind, dissim.size=dissim.size
```

```
# Calculates the squared distance between two vectors x and y
sq.dist = function(x, y){
  return(sum((x-y)^2))
}

# Calculate the WSS for a single cluster
wss.cluster = function(clustermat){
  temp = apply(clustermat, 2,  mean)
  return(sum(apply(clustermat, 1,  function(row){sq.dist(row, temp)})))
}

# Compute the Total WSS across all the clusters.
wss.total = function(data, labels){
  wss.tot = 0
  k = length(unique(labels))
  for(i in 1:k){
    wss.tot = wss.tot + wss.cluster(subset(data, labels == i))
  }
  return(wss.tot)
}
```

```
see <- result[["cluster.store"]]
hc.wss = NULL
for (k in 59:67) {
    hc.wss[k] = wss.total(data, see[k,])
}
```

```
plot(1:9, hc.wss[67:59], type = "b", pch = 18, xlab = "Number of Clusters (k)", ylab = "Total Within-Clu
axis(1, 1:9, 1:9, cex.axis = .75)
```



We see that the elbow is at 4 clusters.

```
table(see[64,])
```

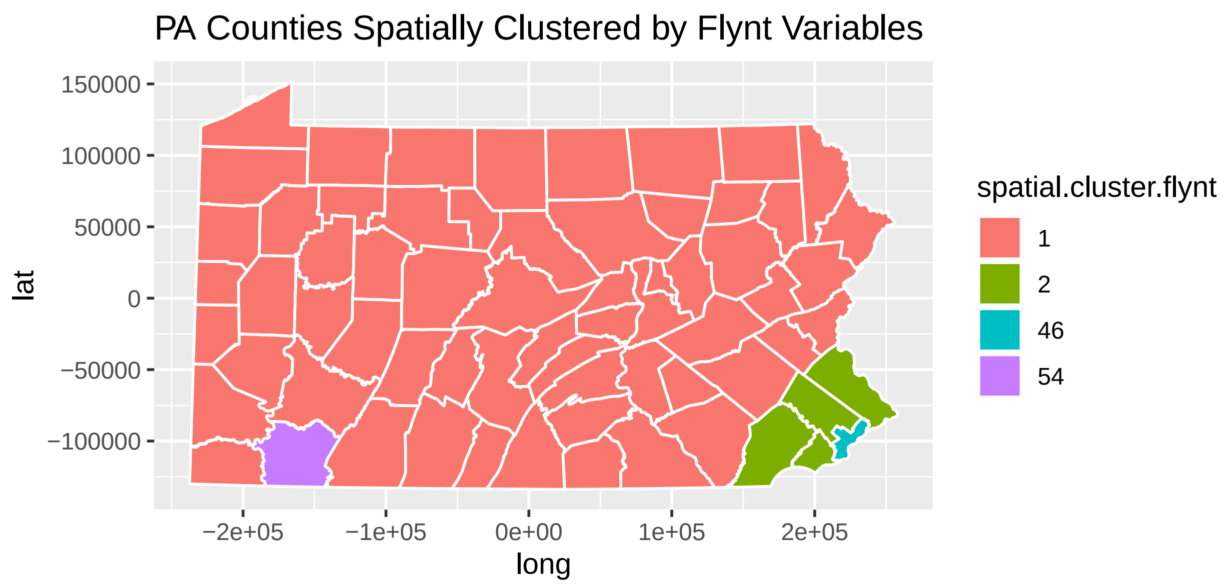
```
##
##  1  2 46 54
## 61  4  1  1
```

We will select row 64 from the cluster result and add it to the data set for mapping purposes.

```
t <- PaCounty@data[1]
t.vec <- as.vector(t[1])
spatial.cluster.flynt <- factor(see[64,])
cluster.flynt <- cbind(t.vec, spatial.cluster.flynt)

region <- match(tolower(cluster.flynt$COUNTY_NAM), tolower(together$County))

region <- factor(region, labels = 0:66)
cluster.flynt <- cbind(cluster.flynt, region)
cluster.flynt <- cluster.flynt[,2:3]
together <- inner_join(together, cluster.flynt, by = "region")
```

```
## Warning: Column `region` joining character vector and factor, coercing into
## character vector
```

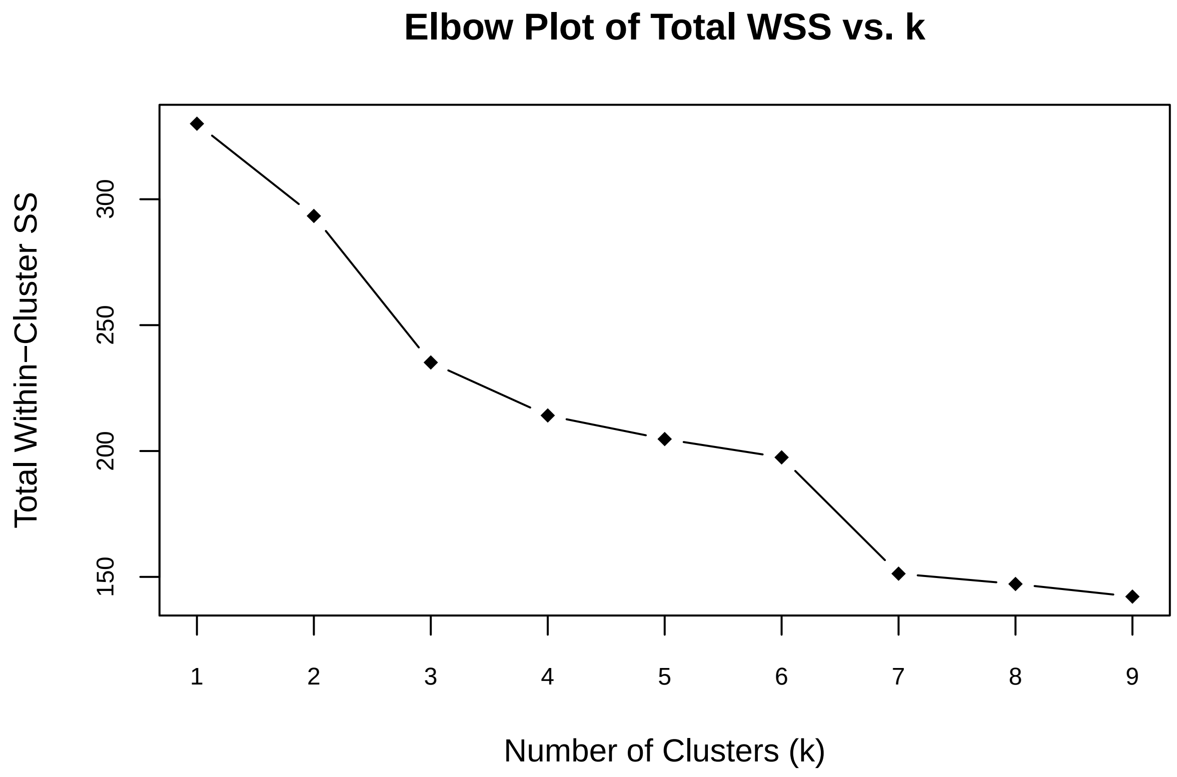
```
ggplot() + geom_polygon(data = together, aes(x = long, y = lat, fill = spatial.cluster.flynt, group=grou
```



###Centroid Hierarchical Clustering

Reset fullData and together for each new cluster.

```r
hc.centroid = hclust(dist(data), method="centroid")
hc.wss = NULL
for (k in 1:9) {
    hc.wss[k] = wss.total(data, cutree(hc.centroid, k))
}

plot(1:9, hc.wss, type = "b", pch = 18, xlab = "Number of Clusters (k)", ylab = "Total Within-Cluster SS
axis(1, 1:9, 1:9, cex.axis = .75)
```

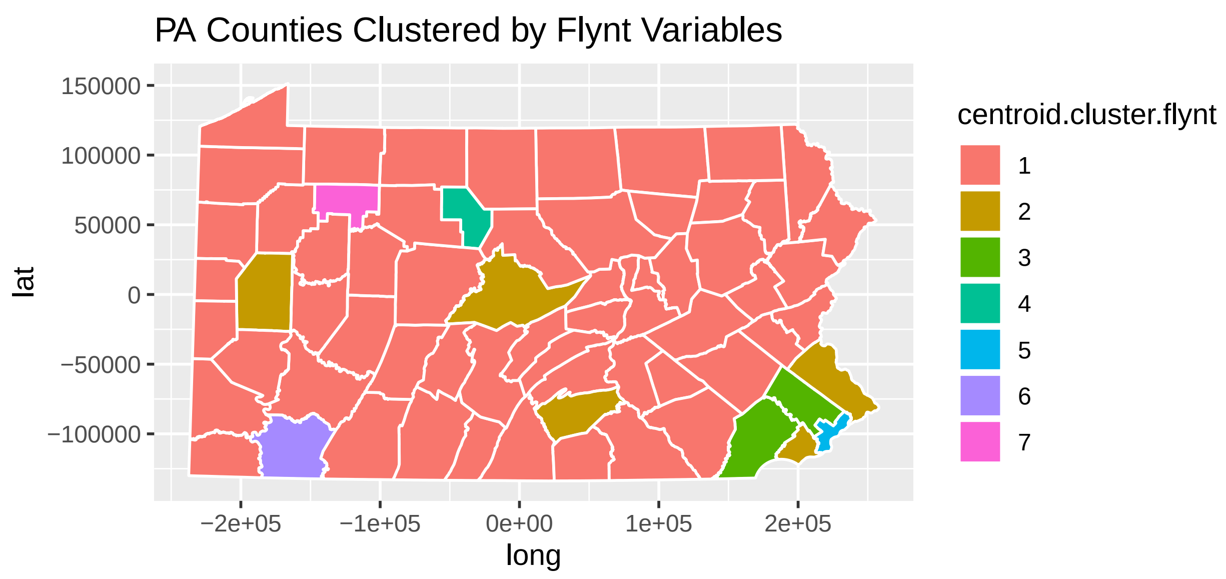


We see that the elbow is at 7 clusters. We will cut the tree at 7 and add this to our data set for mapping purposes.

```r
t <- PaCounty@data[1]
t.vec <- as.vector(t[1])
centroid.cluster <- cutree(hc.centroid, 7)
centroid.cluster.flynt <- factor(centroid.cluster)
cluster.flynt <- cbind(t.vec, centroid.cluster.flynt)

region <- match(tolower(cluster.flynt$COUNTY_NAM), tolower(together$County))

region <- factor(region, labels = 0:66)
cluster.flynt <- cbind(cluster.flynt, region)
cluster.flynt <- cluster.flynt[,2:3]
together <- inner_join(together, cluster.flynt, by = "region")
```

```
## Warning: Column `region` joining character vector and factor, coercing into
## character vector
```

```
ggplot() + geom_polygon(data = together, aes(x = long, y = lat, fill = centroid.cluster.flynt, group=gr
```



PA Counties Clustered by Flynt Variables
150000
100000
50000
0
−50000
−100000
lat
−2e+05
−1e+05
0e+00
1e+05
2e+05
long
centroid.cluster.flynt
1
2
3
4
5
6
7