Project 3: Thematic Mapping & Clustering

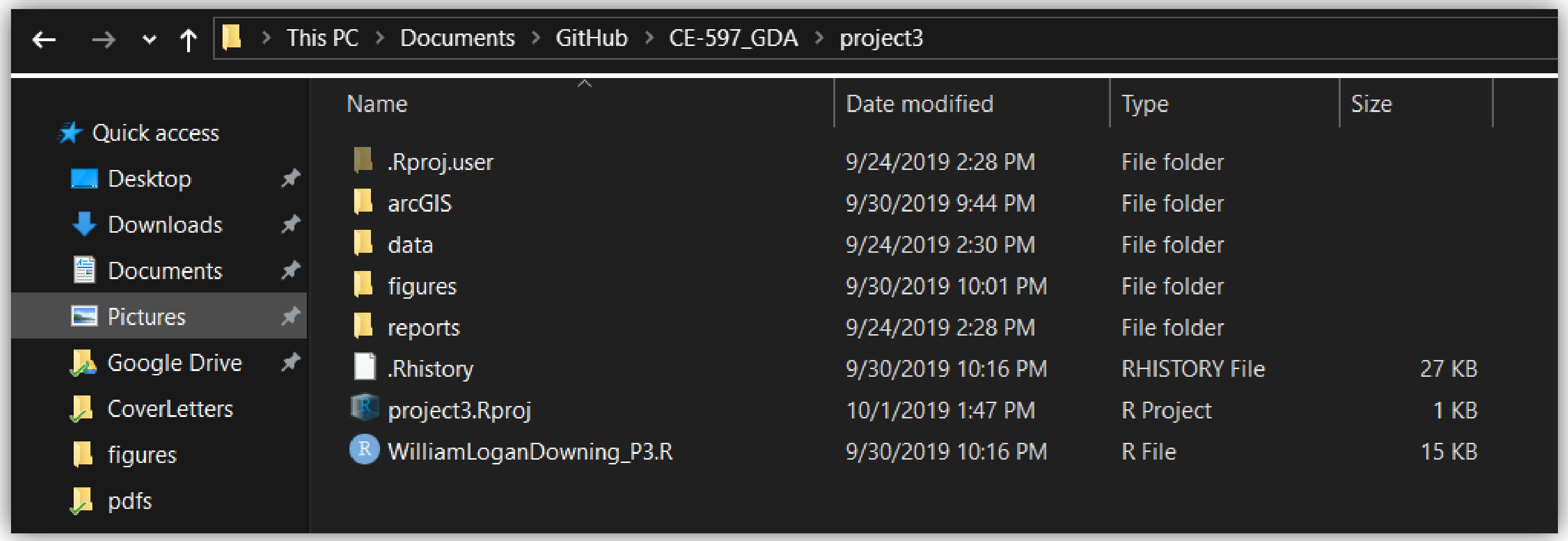
W. Logan Downing

Outline

1. Data Preparation
2. Thematic Mapping of Population Density
3. Develop a Kmeans Clustering Algorithm
4. Kmeans Comparison with ArcGIS

Note

- You will see me using the png() command. This is to save my figures and persist them in the project structure that I've been using to keep myself organized.



An R Project is used to make R Studio use the source file directory as its working directory.

Data Preparation

1. In this section, I opted to work with the tracts data **but** I have also chosen to read in and subset West Lafayette from the tracts for practice.

```r
# Load the block data and clip it first.
blocks <- sf::st_read('./data/raw/tabblock2010_18_pophu', layer='tabblock2010_18_pophu')
#blocks_proj <- sf::st_transform(blocks, crs=32616)
# subset blocks to west lafayette.
wl <- osmdata::getbb('west lafayette, usa')
# need to create a polygon from the bbox.
coords <- matrix(c(wl[1,1], wl[2,1], wl[1,1], wl[2,2], wl[1,2], wl[2,2],
                   wl[1,2], wl[2,1], wl[1,1], wl[2,1]), ncol=2,
                 byrow=T)
geoobj <- st_sfc(st_polygon(list(coords)))
wl_poly <- st_sf(geoobj, crs=st_crs(blocks))

wl_blocks <- st_intersection(blocks, wl_poly)
wl_blocks <- sf::st_transform(wl_blocks, crs=32616)
sf::st_crs(wl_blocks)
wl_blocks$blockArea <- sf::st_area(wl_blocks) %>% set_units(km^2) # set the blockArea
```

Data Preparation

```r
tract <- sf::st_read('./data/raw/Tract_2010Census_DP1_IN/utm', layer='tract_in_selected_utm')
sf::st_crs(tract)
# epsg code found at https://spatialreference.org/ref/epsg/wgs-84-utm-zone-16n/
tract_proj <- sf::st_transform(tract, crs=32616)

tract_proj$blockArea <- sf::st_area(tract_proj) %>% set_units(km^2)
tract_proj <- tract_proj[,c("GEOID10", "NAMELSAD10", "ALAND10", "AWATER10",
                          "INTPTLAT10", "INTPTLON10", "DP0010001", "blockArea")]
class(tract_proj)
```

```
> tract <- sf::st_read('./data/raw/Tract_2010Census_DP1_IN/utm', layer='tract_in_selected_utm')
Reading layer `tract_in_selected_utm' from data source `C:\Users\Downi\Documents\GitHub\CE-597_GDA\pro
ject3\data\raw\Tract_2010Census_DP1_IN\utm' using driver `ESRI Shapefile'
Simple feature collection with 1511 features and 194 fields
geometry type:  POLYGON
dimension:      XY
bbox:           xmin: 403477.8 ymin: 4180915 xmax: 692181.8 ymax: 4625475
epsg (SRID):    26916
proj4string:    +proj=utm +zone=16 +datum=NAD83 +units=m +no_defs
> sf::st_crs(tract)
Coordinate Reference System:
  EPSG: 26916
  proj4string: "+proj=utm +zone=16 +datum=NAD83 +units=m +no_defs"
> # epsg code found at https://spatialreference.org/ref/epsg/wgs-84-utm-zone-16n/
> tract_proj <- sf::st_transform(tract, crs=32616)
>
> tract_proj$blockArea <- sf::st_area(tract_proj) %>% set_units(km^2)
> tract_proj <- tract_proj[,c("GEOID10", "NAMELSAD10", "ALAND10", "AWATER10",
+                           "INTPTLAT10", "INTPTLON10", "DP0010001", "blockArea")]
> class(tract_proj)
[1] "sf"         "data.frame"
```

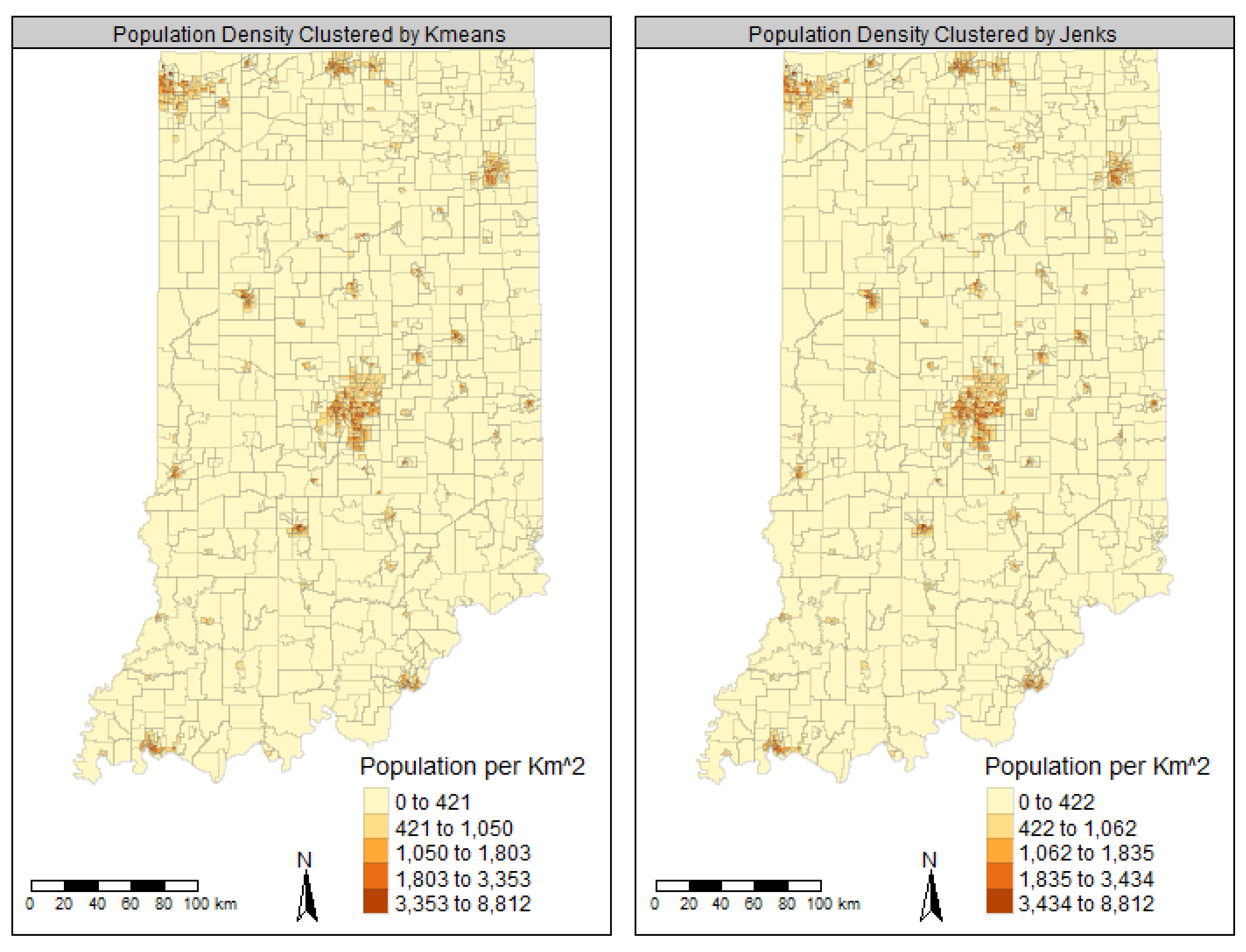
Thematic Mapping of Population Density

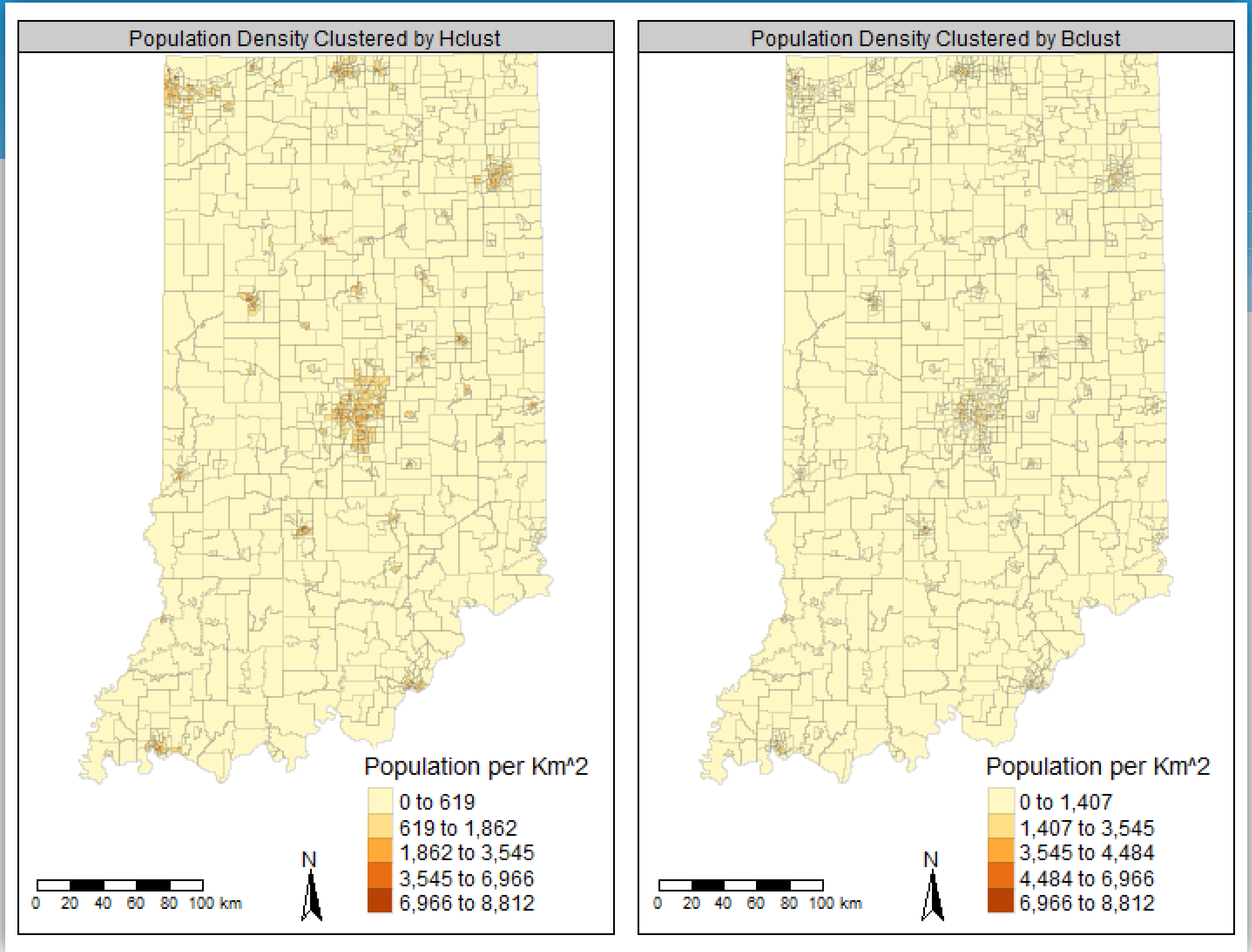
1. For this section, we needed to create thematic maps from the tract population (DP0010001).

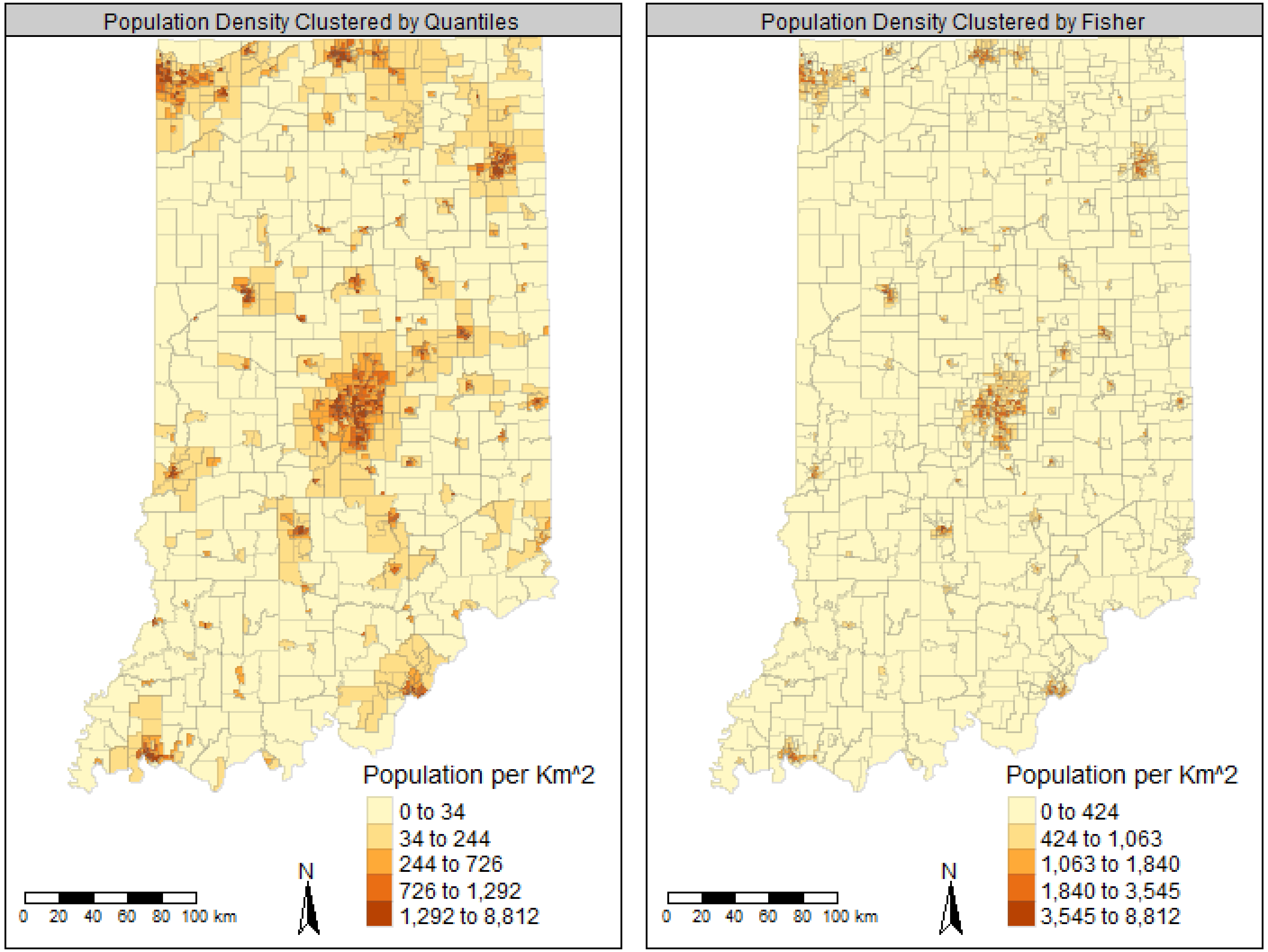
```
k <- tm_shape(tract_proj) + tm_polygons('DP0010001', style='kmeans', title='Block Area (Km^2)') +
  tm_compass(position=c('right', 'bottom'), text.size=1) +
  tm_scale_bar(position=c('left', 'bottom'), text.size=1.5) +
  tm_layout(frame=T, panel.show=T, main.title.position='center', main.title.size=1,
            legend.position=c('right', 'bottom'), legend.text.size=1,
            legend.title.size=1.4, legend.bg.color='white', legend.bg.alpha=.6,
            inner.margins = c(.17, 0, 0, 0),
            panel.labels='Population Clustered by Kmeans')

j <- tm_shape(tract_proj) + tm_polygons('DP0010001', style='jenks', title='Block Area (Km^2)') +
  tm_compass(position=c('right', 'bottom'), text.size=1) +
  tm_scale_bar(position=c('left', 'bottom'), text.size=1.5) +
  tm_layout(frame=T, panel.show=T, main.title.position='center', main.title.size=1,
            legend.position=c('right', 'bottom'), legend.text.size=1,
            legend.title.size=1.4, legend.bg.color='white', legend.bg.alpha=.6,
            inner.margins = c(.17, 0, 0, 0),
            panel.labels='Population Clustered by Jenks')
```

The following clustering methods were used Kmeans, Jenks, Hiearchical, Bayesian, Quantile, and Fisher

```
png('./figures/builtInClustering1.png', width=713, height=545)
tmap_arrange(k,j, nrow=1)
dev.off()
png('./figures/builtInClustering2.png', width=713, height=545)
tmap_arrange(h,b, nrow=1)
dev.off()
png('./figures/builtInClustering3.png', width=713, height=545)
tmap_arrange(q,f, nrow=1)
dev.off()
```


Population Density Clustered by Kmeans
Population per Km^2
0 to 421
421 to 1,050
1,050 to 1,803
1,803 to 3,353
3,353 to 8,812
N
0 20 40 60 80 100 km
Population Density Clustered by Jenks
Population per Km^2
0 to 422
422 to 1,062
1,062 to 1,835
1,835 to 3,434
3,434 to 8,812
N
0 20 40 60 80 100 km


Population Density Clustered by Hclust
Population per Km^2
0 to 619
619 to 1,862
1,862 to 3,545
3,545 to 6,966
6,966 to 8,812
0 20 40 60 80 100 km
N
Population Density Clustered by Bclust
Population per Km^2
0 to 1,407
1,407 to 3,545
3,545 to 4,484
4,484 to 6,966
6,966 to 8,812
0 20 40 60 80 100 km
N



Population Density Clustered by Quantiles
Population per Km^2
0 to 34
34 to 244
244 to 726
726 to 1,292
1,292 to 8,812
0 20 40 60 80 100 km
N
Population Density Clustered by Fisher
Population per Km^2
0 to 424
424 to 1,063
1,063 to 1,840
1,840 to 3,545
3,545 to 8,812
0 20 40 60 80 100 km
N

Developing a Kmeans Clustering Algorithm

```r
# the data value to be provided should be the item you want classes for
my_kmeans <- function(data, k) {
  data <- as.numeric(data) # sanitize the data in the event that it has units
  # first sort the data and get only the unique values
  # you probably don't want the unique values
  # unique values only will mess with the variance probably.
  sorted_vals <- sort(data, decreasing=F)
  # choose k elements at random, this will only happen once to initialize
  centers <- sample(unique(sorted_vals), k) # use unique to guarantee no overlap
  F_val.old <- -9999 # initializing the old F value to determine when to terminate the loop
  F.all <- vector() # put in place to print an iterations graphic
  rank.old <- 0 # initializing a container for previous iteration rankings.

  # In my testing, 100 iterations seemed quick and suffient
  for (j in seq(1, 100)) {


    # need the distances for each k class
    # line below will get the distance from centers and each value
    center_dists <- t(sapply(sorted_vals, function(x) abs(centers-x)))

    # find all values for each center that are less than evertyhing else
    rank <- matrix(apply(center_dists, 1, which.min),ncol=1)
    rank <- cbind(sorted_vals, rank)
    colnames(rank) <- c('val', 'class')

    # now, each row has a class from 1 to k assigned to it.
    # group by the class and calculate variances
    inClass.var = list()
```

Developing a Kmeans Clustering Algorithm

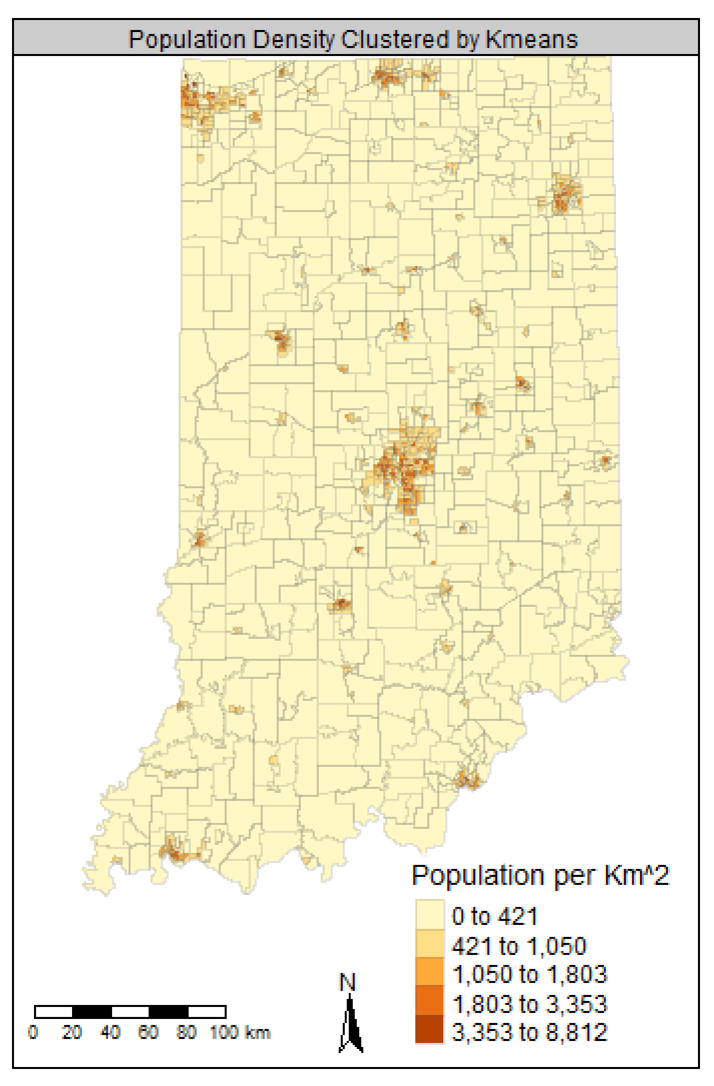
```
    for (i in 1:length(centers)) {
      # grab all class values of a single type
      inClass <- data.frame(rank) %>% dplyr::filter(class == i)
      # find the variance in class
      inClass.var[[i]] = var(inClass$val)

      # adjust the class center
      centers[i] = mean(inClass$val)
    }

    inClass.var <- sum(unlist(inClass.var))
    betweenClass.var = var(centers)
    F_val <-  betweenClass.var/inClass.var

    F.all <- append(F.all, F_val)

    # Save the largest version of F and the rankings.
    # i.e. the jenk's criterion
    if (F_val > F_val.old) {
      iteration_max <- j
      F_val.old <- F_val
      rank.old <- rank
    }

  }
```

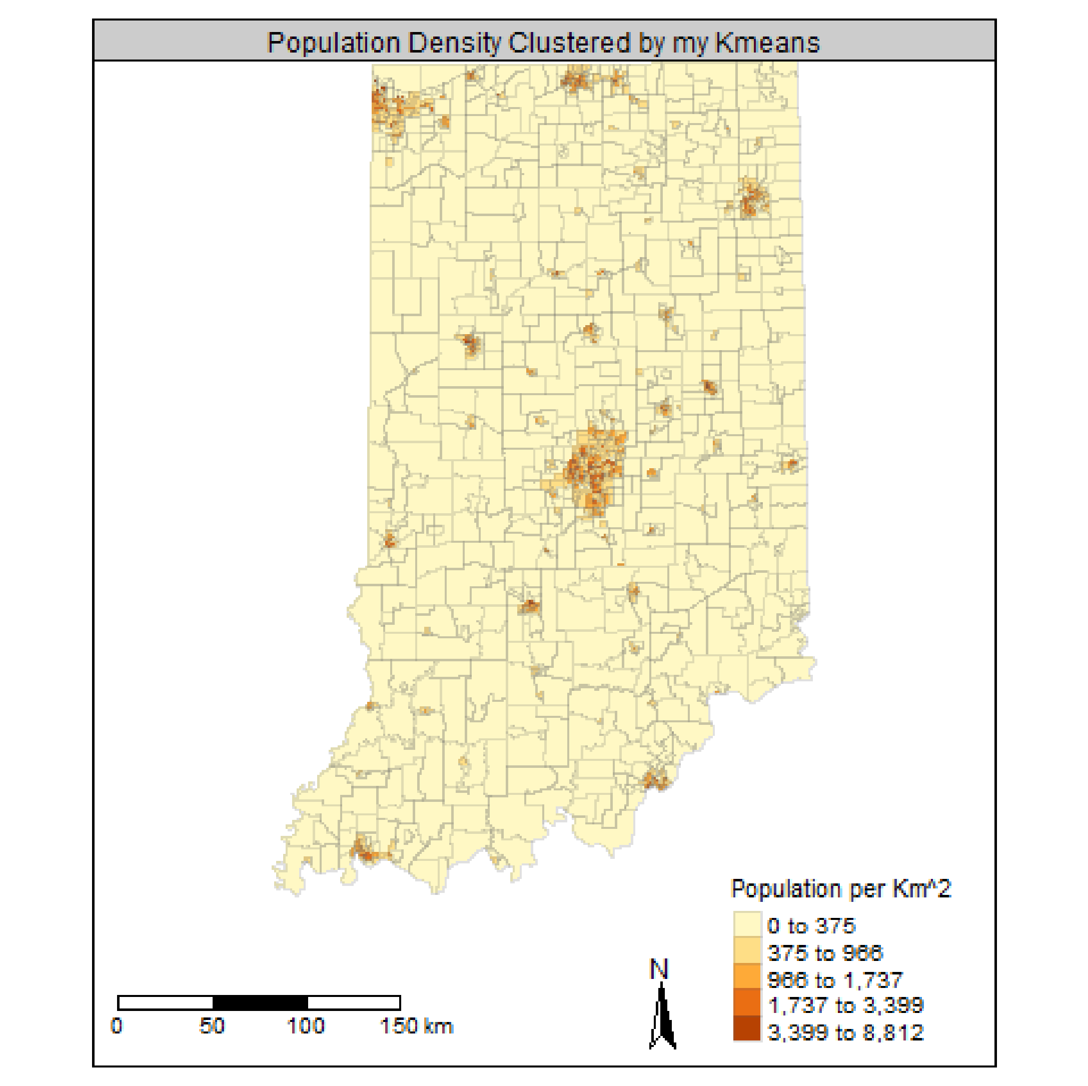
```
  # rank.old should have the maximal F rankings
  # break into a 2 column set with min and max values for each
  classBreaks <- data.frame(rank.old) %>% group_by(class) %>%
    summarize(min_val=min(val), max_val=max(val))

  classBreaks <- as.vector(list(sort(c(classBreaks$min_val, max(classBreaks$max_val))))[[1]])

  png('./figures/F_graph.png')
  plot(F.all, xlab='Iterations', ylab='F Value', main='Variation of F for each Iteration',
       pch=16, col='deepskyblue3')
  points(iteration_max, F_val.old, pch=16, col='red')
  dev.off()

  return(classBreaks)
}
```

Developing a Kmeans Clustering Algorithm

```
# Tm will automatically fill in the gaps between the breaks
# since my algorithm will not ensure that each segment butts up against each other.
png('./figures/myKmeans.png')
tm_shape(tract_proj) +
  tm_polygons('density', style='fixed', breaks=my_breaks,
              title='Population per Km^2', border.alpha=.2 ) +
  tm_compass(position=c('right', 'bottom'), text.size=1) +
  tm_scale_bar(position=c('left', 'bottom'), text.size=.8) +
  tm_layout(frame=T, panel.show=T, panel.labels='Population Density Clustered by my Kmeans',
            legend.bg.color='white', legend.text.size=.8,
            legend.bg.alpha=.6, inner.margins = c(.17, .2, 0, .2))
dev.off()
```

Developing a Kmeans Clustering Algorithm





My Kmeans implementation vs ArcGIS