Lab 2

Thomas Weil GIS 3

- Load in Data
- More Data Frame Manipulation
- Load in census table, merge to find density
- Calculate Standardized Illness Rate
- Estimating the optimal number of clusters
- Building the geodemographic
- Load in the clusters premade from another data set
- Examine Clusters
- Map The Clusters

In this lab we will be constructing a clustered index, to show areas that are similar in terms of several demographic, housing and socio economic variables. This will allow us to see the degree to which certain areas in the city are like other areas.

Load in Data

Create a data frame of Numerators and Denominators

```
setwd("/Users/thomasweil/Desktop/YEAR2/ZOOM university/GIS 3/lab 6")
load("census_2011_UK_OA.RData")
Census_2011_Count <- merge(Liverpool,Census_2011_Count_All,by="OA",all.x=TRUE)
head(OAC_Input_Lookup[,])
```

	VariableCode	Type	Denominator	SubDomain	Domain	VariableDescription
1	k001	Count	KS102EW0001	Population Age	Demographic	Age 0 to 4
2	k002	Count	KS102EW0001	Population Age	Demographic	Age 5 to 14
3	k003	Count	KS102EW0001	Population Age	Demographic	Age 25 to 44
4	k004	Count	KS102EW0001	Population Age	Demographic	Age 45 to 64
5	k005	Count	KS102EW0001	Population Age	Demographic	Age 65 to 89
6	k006	Count	KS102EW0001	Population Age	Demographic	Age 90 and over

6 rows | 1-7 of 9 columns

```
#We will now write some code that will calculate all of the numerators:

OAC_Input <- as.data.frame(Census_2011_Count$OA)
colnames(OAC_Input) <- "OA"

#Our loop in order to create the current variables

for (n in 1:nrow(OAC_Input_Lookup)){
    # Get the variables to aggregate for the row specified by n
    select_vars <- OAC_Input_Lookup[n,"England_Wales"]

    # Create a list of the variables to select
    select_vars <- unlist(strsplit(paste(select_vars),","))

    # Create variable name
    vname <- OAC_Input_Lookup[n,"VariableCode"]

    # Create a sum of the census variables for each Output Area
    tmp <- data.frame(rowSums(Census_2011_Count[,select_vars, drop=FALSE]))
    colnames(tmp) <- vname

    # Append new variable to the OAC_Input object
    OAC_Input <- cbind(OAC_Input,tmp)

    # Remove temporary objects
    remove(list = c("vname","tmp"))
}
OAC_Input$k035 <- NULL

OAC_Input_den <- as.data.frame(Census_2011_Count$OA)
colnames(OAC_Input_den) <- "OA"

den_list <- unique(OAC_Input_Lookup[,"Denominator"])
den_list <- paste(den_list[den_list != ""])
# Select denominators
OAC_Input_den <- Census_2011_Count[,c("OA",den_list)]

OAC_Input <- merge(OAC_Input,OAC_Input_den, by="OA")
```

More Data Frame Manipulation

Create numerator denominator list

```
K_Var <- OAC_Input_Lookup[OAC_Input_Lookup$Type == "Count",c(1,3)]

head(K_Var)
```

	VariableCode	Denominator
1	k001	KS102EW0001
2	k002	KS102EW0001
3	k003	KS102EW0001
4	k004	KS102EW0001
5	k005	KS102EW0001
6	k006	KS102EW0001

6 rows

##Run a loop to compare numerator and denominator and create percentages

```
# Create an OA list / data frame
OAC_Input_PCT_RATIO <- subset(OAC_Input, select = "OA")
# Loop
for (n in 1:nrow(K_Var)){

    num <- paste(K_Var[n,"VariableCode"]) # Get numerator name
    den <- paste(K_Var[n,"Denominator"]) # Get denominator name
    tmp <- data.frame(OAC_Input[,num] / OAC_Input[,den] * 100) # Calculate percentages
    colnames(tmp) <- num
    OAC_Input_PCT_RATIO <- cbind(OAC_Input_PCT_RATIO,tmp) # Append the percentages

    # Remove temporary objects
    remove(list = c("tmp","num","den"))
}
```

```
# Create an OA list / data frame
OAC_Input_PCT_RATIO <- subset(OAC_Input, select = "OA")
# Loop
for (n in 1:nrow(K_Var)){

    num <- paste(K_Var[n,"VariableCode"]) # Get numerator name
    den <- paste(K_Var[n,"Denominator"]) # Get denominator name
    tmp <- data.frame(OAC_Input[,num] / OAC_Input[,den] * 100) # Calculate percentages
    colnames(tmp) <- num
    OAC_Input_PCT_RATIO <- cbind(OAC_Input_PCT_RATIO,tmp) # Append the percentages

    # Remove temporary objects
    remove(list = c("tmp","num","den"))
}
```

Load in census table, merge to find density

Use merge function

```
#Extract Variable
tmp <- Census_2011_Count[,c("OA","KS101EW0008")]
colnames(tmp) <- c("OA","k007")
#Merge
OAC_Input_PCT_RATIO <- merge(OAC_Input_PCT_RATIO,tmp,by="OA")
```

Calculate Standardized Illness Rate

```
# Calculate rates of ill people 15 or less and greater than or equal to 65
ill_16_64 <- rowSums(Census_2011_Count[,c("KS301EW0005","KS301EW0006")]) # Ill people 16-64
ill_total <- rowSums(Census_2011_Count[,c("KS301EW0002","KS301EW0003")]) # All ill people
ill_L15_G65 <- ill_total - ill_16_64 # Ill people 15 or less and greater than or equal to 65
# Calculate total people 15 or less and greater than or equal to 65
t_pop_16_64 <- rowSums(Census_2011_Count[,c("KS102EW0007","KS102EW0008","KS102EW0009","KS102EW0010","KS102EW0011","KS102EW0012","KS102EW0013")]) # People 16-64
t_pop <- Census_2011_Count$KS101EW0001 # All people
t_pop_L15_G65 <- t_pop - t_pop_16_64 # All people 15 or less and greater than or equal to 65
# Calculate expected rate
ex_ill_16_64 <- t_pop_16_64 * (sum(ill_16_64)/sum(t_pop_16_64)) # Expected ill 16-64
ex_ill_L15_G65 <- t_pop_L15_G65 * (sum(ill_L15_G65)/sum(t_pop_L15_G65)) # Expected ill people 15 or less and greater than or equal to 65
ex_ill <- ex_ill_16_64 + ex_ill_L15_G65 # total expected ill people
# Ratio
SIR <- as.data.frame(ill_total / ex_ill * 100) # ratio between ill people and expected ill people
colnames(SIR) <- "k035"
# Merge data
OAC_Input_PCT_RATIO <- cbind(OAC_Input_PCT_RATIO,SIR)
# Remove unwanted objects
remove(list=c("SIR","ill_16_64","ill_total","ill_L15_G65","t_pop_16_64","t_pop","t_pop_L15_G65","ex_ill_16_64","ex_ill_L15_G65","ex_ill"))
OAC_Input_PCT_RATIO
```

OA	k001	k002	k003	k004	k005	k006	k008	k009
E00032987	6.04804934	12.62626263	41.919192	14.646465	8.08080808	0.0000000	0.0000000	46.94936
E00032988	1.72413793	5.16091954	36.206897	28.448276	9.77011494	1.1494253	16.3670161	73.03030
E00032989	4.20420420	5.10510511	37.237237	27.027027	5.70570571	0.0000000	0.0000000	80.26756
E00032990	4.24242424	2.42424242	46.969697	16.363636	10.6060606	0.3030303	1.2121212	68.40301
E00032991	3.12500000	3.87500000	49.937500	16.250000	1.25000000	0.0000000	0.0000000	81.79470
E00032992	1.25000000	5.83333333	49.583333	18.333333	4.16666667	0.0000000	0.0000000	81.98198
E00032993	2.55148936	6.38297872	46.808511	9.9290780	7.44680851	0.0000000	0.0000000	69.29134
E00032994	7.20720721	16.21621622	19.819820	27.027027	16.2162162	0.0000000	0.0000000	46.80952
E00032995	4.95974627	8.18208955	25.746269	24.253731	23.50746269	0.7462687	0.0000000	50.65302
E00032996	5.99520384	15.82733813	21.103118	27.8177458	18.46522782	0.0000000	0.0000000	42.35669

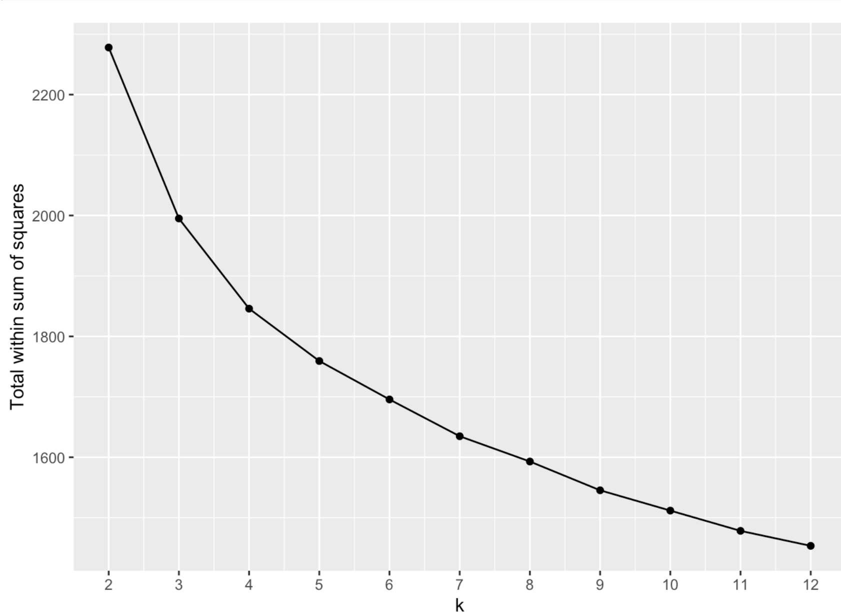
1-10 of 1,584 rows | 1-9 of 61 columns Previous 1 2 3 4 5 6 ... 159 Next

```
# Calculate inverse hyperbolic sine
OAC_Input_PCT_RATIO_IHS <- log(OAC_Input_PCT_RATIO[,2:61]+sqrt(OAC_Input_PCT_RATIO[,2:61]^2+1))
# Calculate Range
range_01 <- function(x){(x-min(x))/(max(x)-min(x))} # range function
OAC_Input_PCT_RATIO_IHS_01 <- apply(OAC_Input_PCT_RATIO_IHS, 2, range_01) # apply range function to columns
# Add the OA codes back onto the data frame as row names
rownames(OAC_Input_PCT_RATIO_IHS_01) <- OAC_Input_PCT_RATIO$OA
```

Estimating the optimal number of clusters

Use sum of squares technique to calculate optimal number of clusters

```
library(ggplot2)
# Create a new empty numeric object to store the wss results
wss <- numeric()
# Run k means for 2-12 clusters and store the wss results
for (i in 2:12) wss[i] <- sum(kmeans(OAC_Input_PCT_RATIO_IHS_01, centers=i,nstart=20)$withinss)
# Create a data frame with the results, adding a further column for the cluster number
wss <- data.frame(2:12,wss[-1])
# Plot the results
names(wss) <- c("k","Twss")
ggplot(data=wss, aes(x= k, y=Twss)) + geom_path() + geom_point() + scale_x_continuous(breaks=2:12) + labs(y = "Total within sum of squares")
```



We will chose 7 clusters

Building the geodemographic

No need to run this, it simple creates optomized clusters.

```
cluster_7 <- kmeans(x=OAC_Input_PCT_RATIO_IHS_01, centers=7, iter.max=1000000, nstart=10000)
OAC_Input_PCT_RATIO_IHS_01
```

Load in the clusters premade from another data set

```
setwd("/Users/thomasweil/Desktop/YEAR2/ZOOM university/GIS 3/lab 6")
# Load cluster object
load("cluster_7.Rdata")
```

Examine Clusters

```
# Lookup Table
lookup <- data.frame(cluster_7$cluster)
# Add OA codes
lookup$OA <- rownames(lookup)
colnames(lookup) <- c("K_7","OA")
# Recode clusters as letter
lookup$SUPER <- LETTERS[lookup$K_7]
```

We will also look at the distribution of these clusters:

```
table(lookup$K_7)
```

```
##
##   1   2   3   4   5   6   7
## 259 340 279 334 109  73 190
```

Map The Clusters

Use tmap

```
# Load packages
library(rgdal)
```

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

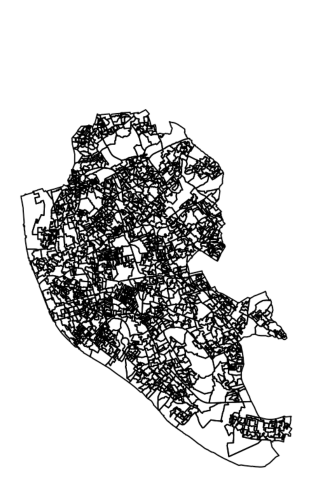
```
## rgdal: version: 1.3-6, (SVN revision 773)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.1.3, released 2017/20/01
##  Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/3.5/Resources/library/rgdal/gdal
##  GDAL binary built with GEOS: FALSE
##  Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
##  Path to PROJ.4 shared files: /Library/Frameworks/R.framework/Versions/3.5/Resources/library/rgdal/pro
j
##  Linking to sp version: 1.3-1
```

```
library(tmap)
# Import OA boundaries
liverpool_SP <- readOGR("Liverpool_OA_2011.geojson", "OGRGeoJSON")
```

```
## OGR data source with driver: GeoJSON
## Source: "/Users/thomasweil/Desktop/YEAR2/ZOOM university/GIS 3/lab 6/Liverpool_OA_2011.geojson", layer
: "OGRGeoJSON"
## with 1584 features
## It has 1 fields
```

```
# Merge lookup

plot(liverpool_SP)
```



```
liverpool_SP@data
```

	oa_code
	<fctr>
0	E00033384
1	E00033305
2	E00033833
3	E00176725
4	E00034020
5	E00176675
6	E00176733
7	E00034034
8	E00176718
9	E00165724

1-10 of 1,584 rows Previous 1 2 3 4 5 6 ... 159 Next

```
liverpool_SP <- merge(liverpool_SP, lookup, by.x="oa_code",by.y="OA")

tmap_mode("plot")
```

```
## tmap mode set to plotting
```

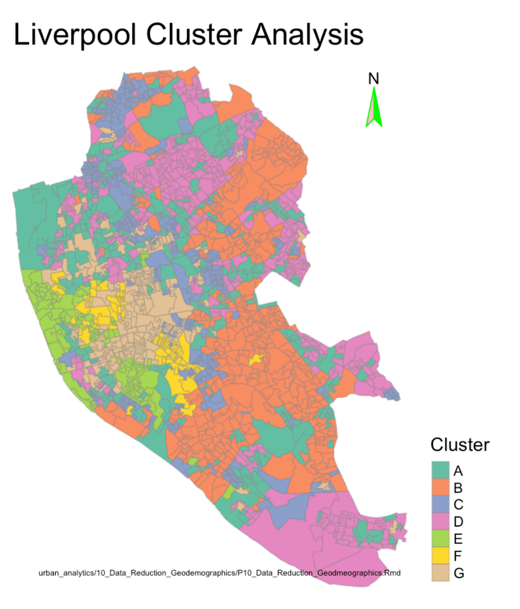
```
tmap_style("beaver")
```

```
## tmap style set to "beaver"
```

```
## other available styles are: "white", "gray", "natural", "cobalt", "col_blind", "albatross", "bw", "cla
ssic", "watercolor"
```

```
m <- tm_shape(liverpool_SP, projection=27700) +
    tm_polygons(col="SUPER", border.col = "grey50",   palette="Set2",border.alpha = .3, title="Cluster",
showNA=FALSE) +
  tm_layout(legend.outside=TRUE,legend.position = c("left", "bottom"), frame = FALSE, main.title= "Liverp
ool Cluster Analysis", bg.color="white")+tm_compass(position= c("right", "top"), color.dark="green", colo
r.light="pink", text.color= "red", size=2)+tm_credits("urban_analytics/10_Data_Reduction_Geodemographics/
P10_Data_Reduction_Geodmeographics.Rmd
")+tm_basemap("Stamen.Watercolor")

m
```




```
#Create leaflet plot
```

This map, showing the areas that belong to each data cluster, shows some clear spatial clustering as well. The downtown area has many areas in the G category, then there seem to be a few rings, descending down to f, e, C, and then the rest. The d tend to be farther out. There is an interesting cluster of the E cluster around the F and and G but out near the water. This is porbbably due to specific enviormental conditions/job oppurtunities specific to those groups of people (fishing?). These are my current take aways from the map.