Lab 6: Geodemographics & Data Reduction

Mark Baker

Overview

For this lab, I am following the Geodemographics & Data Reduction Lab from Urban Analytics (Singleton 2017). Most of the code presented in this lab will be directly pulled from the tutorial with my interpretation of the steps that are being taken. Additionally, at the end of this lab, I aim to improve the final map generated highlighting clustered communities. I have included the data used for this assignment as well.

Loading Data

First, we will load in the main data source for this assignment.

```
load("./data/census_2011_UK_OA.RData")
```

From this, I will then crop the data to highlight Liverpool.

```
Census_2011_Count <- merge(Liverpool,Census_2011_Count_All,by="OA",all.x=TRUE)
```

Beginning Aggregation

Now, looking at the the first 6 rows in the *OAC_Input_Lookup* Dataset, we can see that aggregation is necessary.

```
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
##                           England_Wales
## 1                           KS102EW0002
## 2 KS102EW0003,KS102EW0004,KS102EW0005
## 3             KS102EW0010,KS102EW0011
## 4             KS102EW0012,KS102EW0013
## 5 KS102EW0014,KS102EW0015,KS102EW0016
## 6                           KS102EW0017
```

Preforming this aggregation within a for loop:

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

```r
# Loop through each row in the OAC input table
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"]

      # Creates 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"))

} # END: Loop through each row in the OAC input table
```

We have done this for all variable codes including k035, which we are not necessarily interested in. For this reason, we will set this value to NULL.

```r
OAC_Input$k035 <- NULL
```

Doing the protocol presented above, we generated the numerators of interest. Now we will do the same for the denominators.

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

# Create a list of denominators
den_list <- unique(OAC_Input_Lookup[,"Denominator"])
den_list <- paste(den_list[den_list != ""])

# Selecting denominators
OAC_Input_den <- Census_2011_Count[,c("OA",den_list)]
```

After completing this, we will them merge these two data frames to preform further manipulations.

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

Calculating Percentages

To get the precentages, we are interested in the columns where the type is Count, meaning it is not a ratio.

```r
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
```

Now using K_Var, we will now calculate the precentages.

```
# Create an OA list / data frame
OAC_Input_PCT_RATIO <- subset(OAC_Input, select = "OA")

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"))
}
```

Standardized Illness Rates (SIR)

The goal of this section is to calculate the variable k035 which was the standardized illness rate (SIR) - which needs to be calculated for each subset of the national data (in this case Liverpool):

First, we will 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
```

Next, our goal is to calculate total 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","KS
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
```

Now, to calculate expected rate and ratio.

```
# 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 les

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"
```

With this completed, we will merge the data and remove the undesired objects.

```
# 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_
```

Note: this code is the same as the code above. But, i included it again as this was the only way to get the file to properly knit.

```
# 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","KS
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

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_
```

Standardization

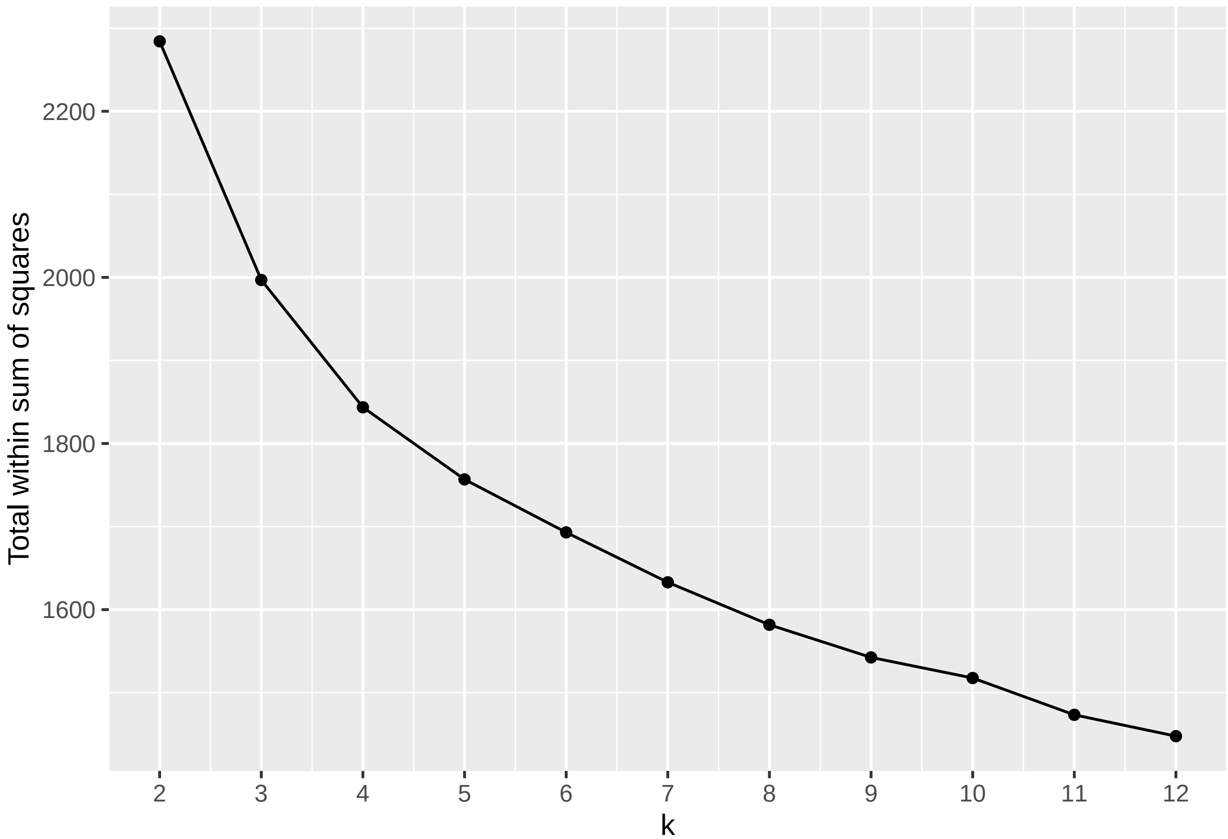
```
# Calculate inverse hyperbolic sine
OAC_Input_PCT_RATIO_IHS <- log(OAC_Input_PCT_RATIO[,2:61]+sqrt(OAC_Input_PCT_RATIO[,2:61]^2+1))

# 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 col

# 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 number of clusters

With the standardized data, I want to now cluster the data for the spatial analysis. To do this, we must estimate the number of clusters.

```
library(ggplot2)

# Create a new empty numeric object to store the wss results
```

```r
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) + lal
```



Based on this, we see thaat the slope of the line begins to tail off between 7 and 8 and for that this this lab choose 7.

Building the geodemographic

Loading the cluster in the clustered data.

```r
load("./data/cluster_7.Rdata")
```

Viewing the data.

```r
str(cluster_7)
```

```
## List of 9
##  $ cluster     : Named int [1:1584] 7 5 7 5 5 7 5 1 1 4 ...
##   ..- attr(*, "names")= chr [1:1584] "E00032987" "E00032988" "E00032989" "E00032990" ...
```

```
##  $ centers     : num [1:7, 1:60] 0.553 0.584 0.677 0.666 0.391 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:7] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:60] "k001" "k002" "k003" "k004" ...
##  $ totss       : num 2827
##  $ withinss    : num [1:7] 286 308 250 255 159 ...
##  $ tot.withinss: num 1635
##  $ betweenss   : num 1192
##  $ size        : int [1:7] 259 340 279 334 109 73 190
##  $ iter        : int 6
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

With this, we can access the data as follows.

```
# 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]
```

Mapping the clusters as presented in the tutorial

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

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

```
## rgdal: version: 1.4-8, (SVN revision 845)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.4.2, released 2019/06/28
##  Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/3.6/Resources/library/rgdal/gdal
##  GDAL binary built with GEOS: FALSE
##  Loaded PROJ.4 runtime: Rel. 5.2.0, September 15th, 2018, [PJ_VERSION: 520]
##  Path to PROJ.4 shared files: /Library/Frameworks/R.framework/Versions/3.6/Resources/library/rgdal/pr
##  Linking to sp version: 1.3-2
```

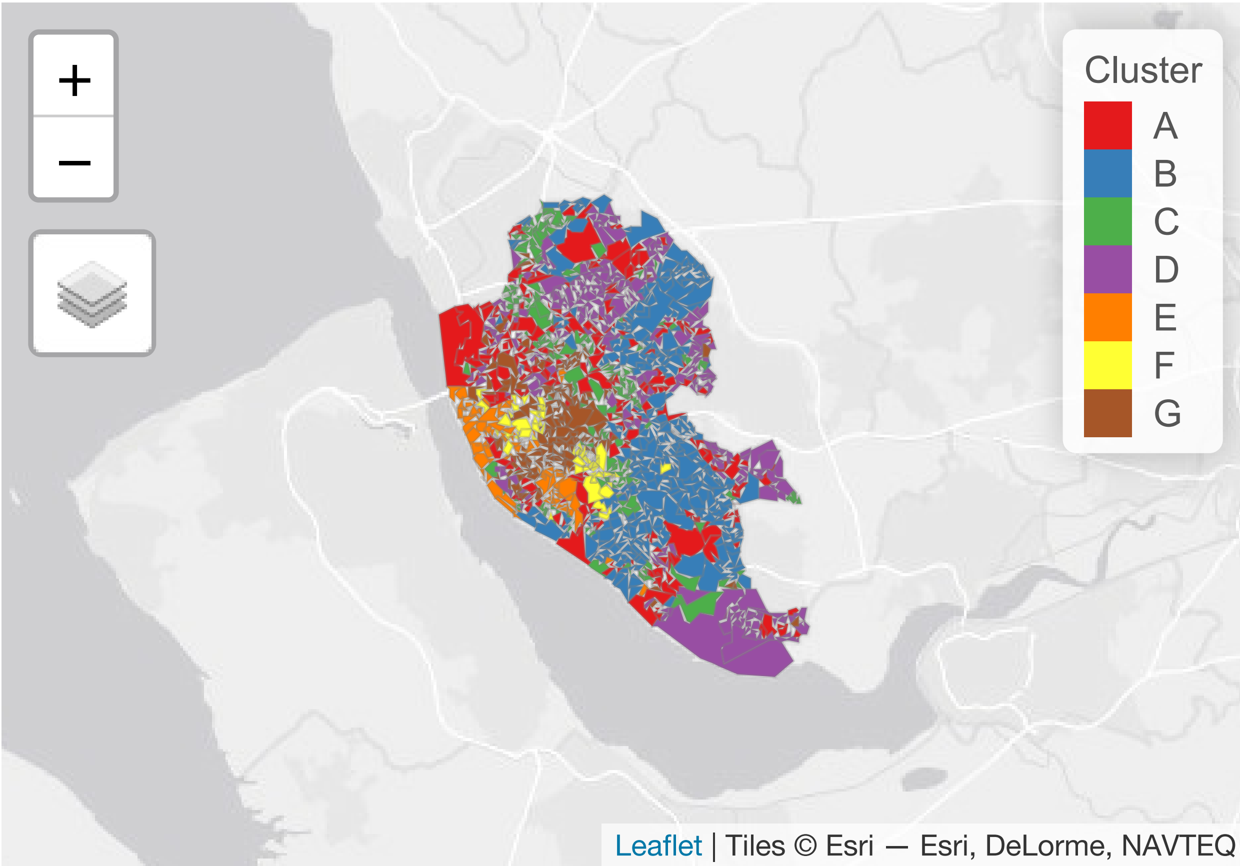
```
library(tmap)

# Import OA boundaries
liverpool_SP <- readOGR("./data/Liverpool_OA_2011.geojson")
```

```
## OGR data source with driver: GeoJSON
## Source: "/Users/markbaker/Downloads/urban_analytics-master/10_Data_Reduction_Geodemographics/data/Liv
## with 1584 features
## It has 1 fields
```

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

m <- tm_shape(liverpool_SP, projection=27700) +
    tm_polygons(col="SUPER", border.col = "grey50",   palette="Set1",border.alpha = .3, title="Cluster"
  tm_layout(legend.position = c("left", "bottom"), frame = FALSE)
```

```r
#Create leaflet plot
tmap_leaflet(m)
```

```
## Warning: The shape liverpool_SP is invalid. See sf::st_is_valid

## Linking to GEOS 3.7.2, GDAL 2.4.2, PROJ 5.2.0

## legend.postion is used for plot mode. Use view.legend.position in tm_view to set the legend position
```



Creating cluster descriptions and profiles

To further understand the classification, we can examine the rates for input attributes within each cluster compared to the Liverpool average.
To to this, we can created indices.

```r
# Merge Original Data (inc. denominators)
LiVOAC_Lookup_Input <- merge(lookup,OAC_Input,by="OA",all.x=TRUE)

# Remove Ratio Variables
LiVOAC_Lookup_Input$k007 <- NULL
LiVOAC_Lookup_Input$k035 <- NULL

# Create Aggregations by SuperGroup
SuperGroup <-aggregate(LiVOAC_Lookup_Input[,4:78], by=list(LiVOAC_Lookup_Input$SUPER),  FUN=sum)

# Create a data frame that will be used to append the index scores
```

```r
G_Index <- data.frame(SUPER=LETTERS[1:7])

# 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(round((SuperGroup[,num] / SuperGroup[,den]) / (sum(SuperGroup[,num])/sum(SuperGroup
  colnames(tmp) <- num

  G_Index <- cbind(G_Index,tmp) # Append the index calculations

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

# View the index scores
G_Index
```

```
##   SUPER k001 k002 k003 k004 k005 k006 k008 k009 k010 k011 k012 k013 k014 k015
## 1     A   83   91   91  114  147  237   90   92   84  144  106   81   38   40
## 2     B   90  109   84  129  124  121   23   65  164   71  106   60   97   92
## 3     C  125  104  115   98   87   66    8   98  105  102  106   78   71   56
## 4     D  121  129   92  104  108   75   31   95   98  117  107   63   59   23
## 5     E   45   21  184   59   33   41   98  152   42   80   89  171  210  197
## 6     F   35   31   62   32   30   37  933  171   29   33   84  137  280  348
## 7     G  129  113  120   83   73   50   20  112   76  125   77  238  139  195
##   k016 k017 k018 k019 k020 k021 k022 k023 k024 k025 k026 k027 k028 k029 k030
## 1   41   43   59   46  105   60   73   51   73   95    6   51   89   82  155
## 2   48   73   25   32  105   68   29   44  142  130   11  317  221   24   30
## 3   44   47   48   38  104   81   75   55  115  100   58   30   45  185   43
## 4   29   39   49   22  106   41   76   57   79  140    4   66  119  147   11
## 5   79  171  146  275   86  432  186  136  144   14  283   14    9    8  375
## 6  393  415  131  143   86  199  116  148   71   42 1040   31   26  101  199
## 7  305  186  408  408   84  134  291  357   68   70  128   57   59  109  143
##   k031 k032 k033 k034 k036 k037 k038 k039 k040 k041 k042 k043 k044 k045 k046
## 1   70  183   61   92  109  100   62   64   44   57   97   83   83  128   98
## 2  181   11   43   23  123  110   84  143   54  236   80  157   61   53   91
## 3  127   31  128   59   98  113   94  105   64   93  128  115   98   97   90
## 4   87  164   48   69  113  117   68   45   54   65  104   87   89  132  111
## 5   39   67  263  302   53   57  112  227  148   61  105   87  260   79   64
## 6   54   71  231  261   42   47  322   97  480   77   72   42  114   38  178
## 7   48  152  140  159   82   93   84   88  101   39  111   65  117  158  112
##   k047 k048 k049 k050 k051 k052 k053 k054 k055 k056 k057 k058 k059 k060
## 1  101   56  114  112  107  103  126   90   76   93  120   99   84  102
## 2  104   73  115  106  104   87   94   58  119  121   70  125  124   96
## 3  104   98  100  100  104   98  106   80   97  107   92  114  104  104
## 4   95  110  120  121  123  113  126   93   54   82  136   87   68  110
## 5  117   98   48   67   61   78   58  132  209  117   78   82  121   90
## 6   64  295   37   43   25  143   44  258  102   60   76   54  105   75
## 7   95  108   81   90  102  105   89  165   82   80  137   71   86  103
```

To assist with spotting trends within the grand index table we can visualize create a plot of shaded cells.

```r
library(reshape2)
```

```r
# Convert from wide to narrow format
G_Index_Melt <- melt(G_Index, id.vars="SUPER")
# View the top of the new narrow formatted data frame
head(G_Index_Melt)
```

```
##   SUPER variable value
## 1     A     k001    83
## 2     B     k001    90
## 3     C     k001   125
## 4     D     k001   121
## 5     E     k001    45
## 6     F     k001    35
```

Creating shaded plot

```r
# Recode the index scores into aggregate groupings
G_Index_Melt$band <- ifelse(G_Index_Melt$value <= 80,"< 80",ifelse(G_Index_Melt$value > 80 & G_Index_Mel

# Add a column with short descriptions of the variables
short <- read.csv("./data/OAC_Input_Lookup_short_labels.csv")
G_Index_Melt <- merge(G_Index_Melt,short,by.x="variable",by.y="VariableCode",all.x=TRUE)

# Order the created factors appropriately - needed to ensure the legend and axis make sense in ggolot2
G_Index_Melt$band <- factor(G_Index_Melt$band, levels = c("< 80","80-120",">120"))
G_Index_Melt$VariableDescription <- factor(G_Index_Melt$VariableDescription, levels = short$VariableDesc
```

Using ggplot2 we can now create a shaded table which you can use to come up with descriptions of the clusters and creative labels.

```r
library(ggplot2)
p <- ggplot(G_Index_Melt, aes(x=SUPER, y=VariableDescription, label=value, fill=band)) +
  scale_fill_manual(name = "Band",values = c("#EB753B","#F7D865","#B3D09F")) +
  scale_x_discrete(position = "top") +
  geom_tile(alpha=0.8) +
  geom_text(colour="black")
p
```

SUPER

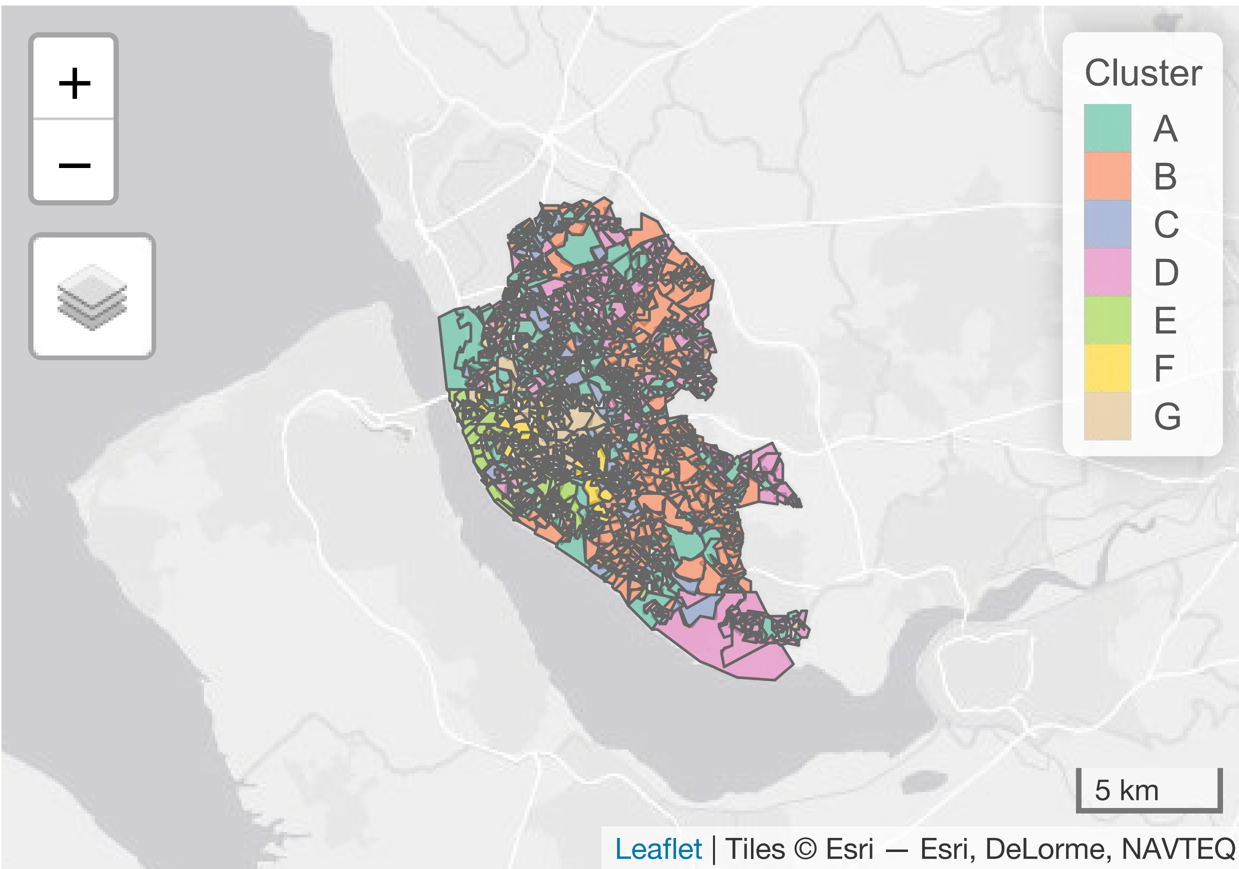
VariableDescription	A	B	C	D	E	F	G
Health	102	96	104	110	90	75	103
Education	84	124	104	68	121	105	86
Public sector	99	125	114	87	82	54	71
Admin	120	70	92	136	78	76	137
Finance	93	121	107	82	117	60	80
IT	76	119	97	54	209	102	82
Accom. and food	90	58	80	93	132	258	165
Haulage / Warehouse	126	94	106	126	58	44	89
Garage	103	87	98	113	78	143	105
Utilities	107	104	104	123	61	25	102
Manufacturing	112	106	100	121	67	43	90
Mining / construction	114	115	100	120	48	37	81
Agriculture	56	73	98	110	98	295	108
Full−time	101	104	104	95	117	64	95
Part−time	98	91	90	111	64	178	112
Unemployed	128	53	97	132	79	38	158
Foot / Bicycle	83	61	98	89	260	114	117
Private Transport	83	157	115	87	87	42	65
Public Transport	97	80	128	104	105	72	111
2+ cars	57	236	93	65	61	77	39
School and FT students	44	54	64	54	148	480	101
Qual L4+	64	143	105	45	227	97	88
Qual L3	62	84	94	68	112	322	84
Qual L1/2	100	110	113	117	57	47	93
Provides unpaid care	109	123	98	113	53	42	82
Occupancy room <=1	92	23	59	69	302	261	159
Private rented	61	43	128	48	263	231	140
Social rented	183	11	31	164	67	71	152
Owned	70	181	127	87	39	54	48
Flats	155	30	43	11	375	199	143
Terrace	82	24	185	147	8	101	109
Semi−detached	89	221	45	119	9	26	59
Detached	51	317	30	66	14	31	57
FT student household	6	11	58	4	283	1040	128
Non−dependent children household	95	130	100	140	14	42	70
No children household	73	142	115	79	144	71	68
Limited English	51	44	55	57	136	148	357
Other EU − post 2001	73	29	75	76	186	116	291
Other EU − 2001	60	68	81	41	432	199	134
UK and Ireland	105	105	104	106	86	86	84
Other ethnic groups	46	32	38	22	275	143	408
Black	59	25	48	49	146	131	408
Chinese and Other	43	73	47	39	171	415	186
Bangladeshi	41	48	44	29	79	393	305
Pakistani	40	92	56	23	197	348	195
Indian	38	97	71	59	210	280	139
Mixed/multiple ethnic group	81	60	78	63	171	137	238
White	106	106	106	107	89	84	77
Divorced or Separated	144	71	102	117	80	33	125
Married or civil partnership	84	164	105	98	42	29	76
Single	92	65	98	95	152	171	112
Communal establishment	90	23	8	31	98	933	20
Age 90 and over	237	121	66	75	41	37	50
Age 65 to 89	147	124	87	108	33	30	73
Age 45 to 64	114	129	98	104	59	32	83
Age 25 to 44	91	84	115	92	184	62	120
Age 5 to 14	91	109	104	129	21	31	113
Age 0 to 4	83	90	125	121	45	35	129

Band
- < 80
- 80−120
- >120

Enchancing the plot generated in this tutorial:

```
map <- tm_shape(liverpool_SP, projection=27700) + #Ploting Liverpool using tmap
  tm_polygons(col="SUPER", alpha = .70, border.col = "grey40",  border.lwd=.3, palette="Set2", title="Cl
  tm_layout (frame = FALSE, bg.color = "transparent")+ #adjusting the legend style
  tm_scale_bar() +  #Addign a scale bar
  tm_compass()  # Adding a compass

#Create leaflet plot
tmap_leaflet(map)
```

```
## Compass not supported in view mode.

## Warning: The shape liverpool_SP is invalid. See sf::st_is_valid
```



Interpretation of the Clusters

Based on the map above, we can see that there are 7 distinct clusters, which I will refer to as the letter they are designated as in the map above. First looking at cluster A, this cluster is most closely associated with older individuals and unemployment. Cluster B are mostly associated with owned detached homes for older individuals (ages 45 plus). CLuster C is associated with individuals that either own or rent there home and are most likely newer family with young children. Cluster D is associated younger individuals 0-14, indentifying as unemployed, lving in socially rented property. Note that clusters A, B, C, and D represent areas with few people of color. On the other hand, cLusters E, G, and F are predominately populated by people of color and have privately rented rooms in flats. These clusters are most clearly differentiate by the persons employment tyoe.