NCG603 - Modelling Voter Turnout Assignment

Sean O’Riogain (18145426)

8 May 2019

```
knitr::opts_chunk$set(echo = TRUE)
getwd()                   # Display the current working directory
```

```
## [1] "C:/Users/oriogain/Dropbox/Maynooth/NCG613 - Data Analytics Project/Assignments"
```

```
suppressPackageStartupMessages(library(tidyverse))  # Needed for dplyr functions etc.
suppressPackageStartupMessages(library(GWmodel))    # Needed for GWR functions
suppressPackageStartupMessages(library(car))        # Needed for Boxplot()
suppressPackageStartupMessages(library(classInt))   # Needed for ClassIntervals()
suppressPackageStartupMessages(library(RColorBrewer)) # Needed for Palette() etc.
suppressPackageStartupMessages(library(spdep))      # Needed for moran() etc.
suppressPackageStartupMessages(library(gclus))      # Needed for cpairs() etc.
suppressPackageStartupMessages(library(sqldf))      # Needed for sqldf() etc.
suppressPackageStartupMessages(library(OutlierDetection)) # Needed for OutlierDetection() etc.
suppressPackageStartupMessages(library(leaps))      # Needed for regsubsets()
suppressPackageStartupMessages(library(MASS))       # Needed for stepAIC()
suppressPackageStartupMessages(library(gridExtra))  # Needed for gridArrange()

data(DubVoter)            # Get the relevant spatial dataframe
str(Dub.voter@data)       # Display the structure of the dataframe's data slot
```

```
## 'data.frame':    322 obs. of  12 variables:
##  $ DED_ID   : num  2001 2002 2003 2004 2005 ...
##  $ X        : num  314792 314418 314253 313677 313957 ...
##  $ Y        : num  235244 234934 234163 234507 234737 ...
##  $ DiffAdd  : num  22.1 23.6 33.5 14 13.9 ...
##  $ LARent   : num  16.2 22.4 75.4 37.3 13.6 ...
##  $ SC1      : num  8.56 7.9 10.61 5.47 6.96 ...
##  $ Unempl   : num  8.01 12.13 8.94 15.19 13.02 ...
##  $ LowEduc  : num  0.6 0.586 0.523 0.699 0.58 ...
##  $ Age18_24 : num  24.2 21.9 28 14.7 12 ...
##  $ Age25_44 : num  39.6 45.5 49.7 39.3 38 ...
##  $ Age45_64 : num  12.3 15.12 9.39 16.57 19.47 ...
##  $ GenEl2004: num  48.9 41 45 52.6 57.3 ...
##  - attr(*, "data_types")= chr  "N" "N" "N" "N" ...
```

Introduction

Objective

The aim of this assignment is to predict the variation in voter turnout across the 322 electoral districts (ED) in Dublin for the General Election of 2002.

Context

To enable us to do this we have a spatial polygons dataframe for those areal units which, in its data slot, contains a row for each ED with a value in each of their 12 columns (variables). Refer to the results of the most recent command above for the structure of that dataset.

Those 12 variables can be broken down into the following categories:

- The first variable contains a unique identifier for each row/ED;
- the next 2 contain the coordinates (of their centroids, presumably);
- the next 5 contain socio-economic (percentage) metrics that are considered to be important predictors of voter participation (e.g. percentage unemployed and percentage of low educational attainment);
- the next 3 socio-economic variables contain the percentage value of the ED's population in each of 3 key voting-eligible age groups;
- and the final (response) variable provides the actual percentage of the electorate that turned out to vote in each ED for the election in question.

Approach

As the last variable referred to above contains the values that we are required to predict, we will be using a suitable, supervised machine learning technique to do so.

Before selecting that technique, we will do due diligence on the dataset itself by assessing its level of data quality (e.g. in terms of valid value ranges, completeness etc.), and by looking for the presence of outliers and collinearity and the extent of same.

Data Quality Analysis

In this section, we will look at the dataset, variable by variable, and assess the level of data quality as well as its spatial distribution in each case.

DED_ID (Unique Identifier)

Here, we will assess whether or not this variable provides a unique identifier for each row in the dataset (and for which spatial distibution is not relevant).

```
paste("DED_ID values are unique? =",              # Check for uniqueness
      length(Dub.voter$DED_ID)==length(unique(Dub.voter$DED_ID)))
```

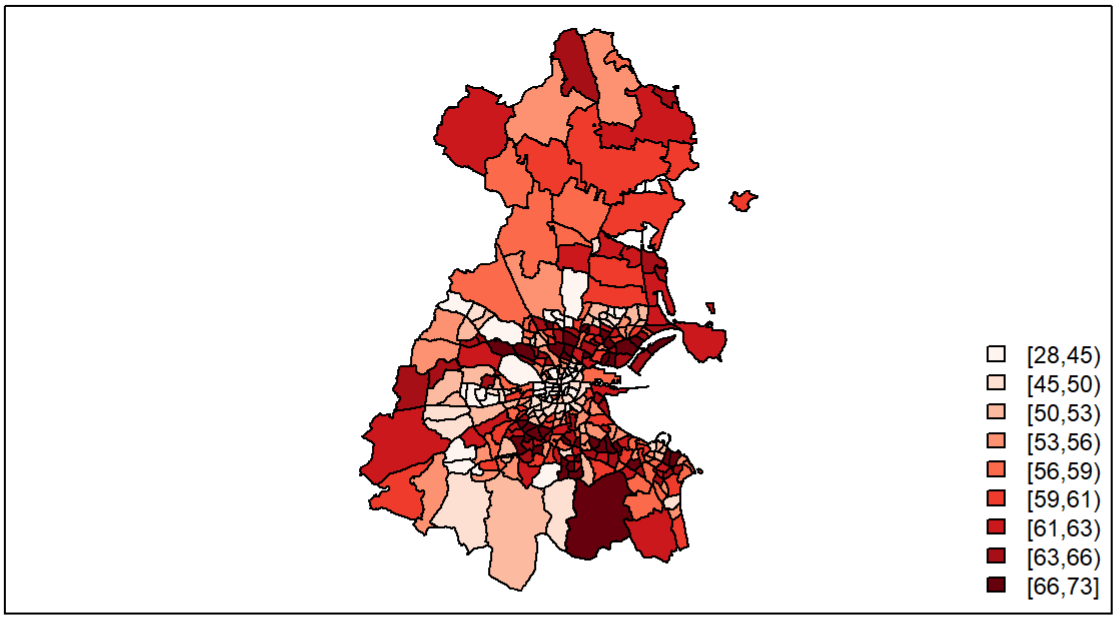
```
## [1] "DED_ID values are unique? = TRUE"
```

X and Y (Coordinates)

We will now attempt to draw a map showing the distribution of the response variable (GenEl2004) to ensure that the dataset refers to EDs in the Dublin area only.

```
quickMap3 <- function(SPDF,Var,nClass=9,dp=0,plotVar=FALSE){
   Classes <- classIntervals(Var,nClass,method="quantile",dataPrecision=dp)
   Palette <- brewer.pal(nClass,"Reds")
   Colours <- findColours(Classes,Palette)
   plot(SPDF,col=Colours)
   legend("bottomright",
      legend=names(attr(Colours,"table")),
      fill=attr(Colours,"palette"),
      cex=0.75,bty="n")
   box()
   if(plotVar) {
      xy <- coordinates(SPDF)
      text(xy[,1],xy[,2],round(Var,dp),col="blue",cex=0.5)
   }
}

quickMap3(Dub.voter,Dub.voter$GenEl2004)
```



The above map confirms that we are dealing with EDs in the Dublin area only - i.e. no ED is an ‘outlier’ in that sense (as specified by its coordinates).

We will now attempt to compare the values of the X and Y variables with those that were used to draw the map above:

```
coords<-data.frame(coordinates(Dub.voter),X2=Dub.voter$X,Y2=Dub.voter$Y)
names(coords)<-c("x_poly","y_poly","x_data","y_data")
head(coords)
```

```
##     x_poly   y_poly   x_data   y_data
## 0 314954.7 235623.6 314791.7 235244.0
## 1 314584.0 235180.6 314417.8 234933.8
## 2 314462.3 234450.2 314253.1 234162.9
## 3 313751.3 234766.3 313676.5 234506.9
## 4 314054.4 235028.6 313956.8 234737.3
## 5 311971.0 237173.7 311883.4 236758.4
```

We can see from the sample values above that both sets of coordinates do not quite match up. Having consulted with the data owner, Martin Charlton, we will now get both sets of coordinates in synch using the following code:

```
coords.new<-data.frame(X=coords$x_poly,Y=coords$y_poly)
head(Dub.voter@data)[1:3]
```

```
##   DED_ID        X        Y
## 0   2001 314791.7 235244.0
## 1   2002 314417.8 234933.8
## 2   2003 314253.1 234162.9
## 3   2004 313676.5 234506.9
## 4   2005 313956.8 234737.3
## 5   2006 311883.4 236758.4
```

```
Dub.voter@data %>%
  mutate(X=coords.new$X, Y=coords.new$Y) ->
  Dub.voter@data
head(Dub.voter@data)[1:3]
```

```
##   DED_ID        X        Y
## 1   2001 314954.7 235623.6
## 2   2002 314584.0 235180.6
## 3   2003 314462.3 234450.2
## 4   2004 313751.3 234766.3
## 5   2005 314054.4 235028.6
## 6   2006 311971.0 237173.7
```

We can see from the final results above that the values of the X and Y coordinates in Dub.voter@data (mailto:Dub.voter@data) slot have been updated to the their equivalents from the Dub.voter@polygons (mailto:Dub.voter@polygons) slot.

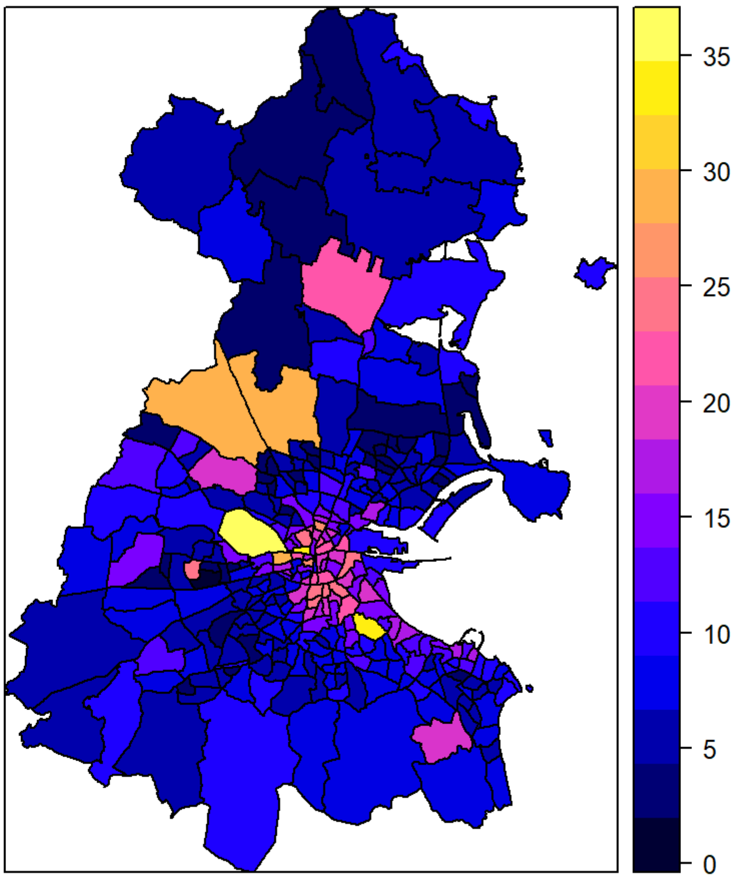
DiffAdd (Percentage who changed address within the past year)

Now let’s take a look at the range of values for this variable.

```
summary(Dub.voter$DiffAdd)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.905   5.485   7.699   9.863  12.603  34.741
```

```
spplot(Dub.voter,"DiffAdd")
```



The above results indicate a reasonable spread of valid percentage values for this variable. The map shows that there are population mobility hotspots in the city centre EDs, in the Dun Laoghaire area and the outer, northwestern suburbs.

LARent (Percentage of local authority renters)

Now let's take a look at the range of values for this variable.

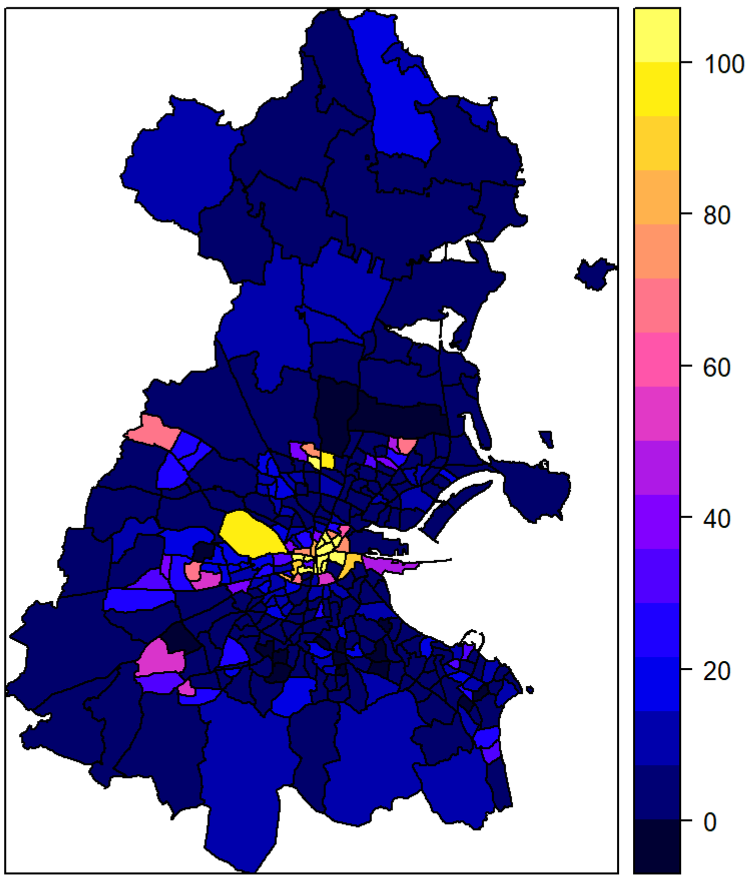
```
summary(Dub.voter$LARent)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   1.113   4.837  15.171  15.997 100.000
```

```
which(Dub.voter$LARent==100)
```

```
##  [1]  18  73  75  88  89 117 119 121 144 145 153 161
```

```
spplot(Dub.voter,"LARent")
```



The above results indicate a wide spread of valid percentage values for this variable. Later in this document we will take a closer look at the observations with the minimum (0) and maximum (100) values to assess the possibility of missing or outlier values.

The map shows that those high-end outlier EDs in relation to public housing are primarily located primarily around the city centre and in the midwestern suburbs.

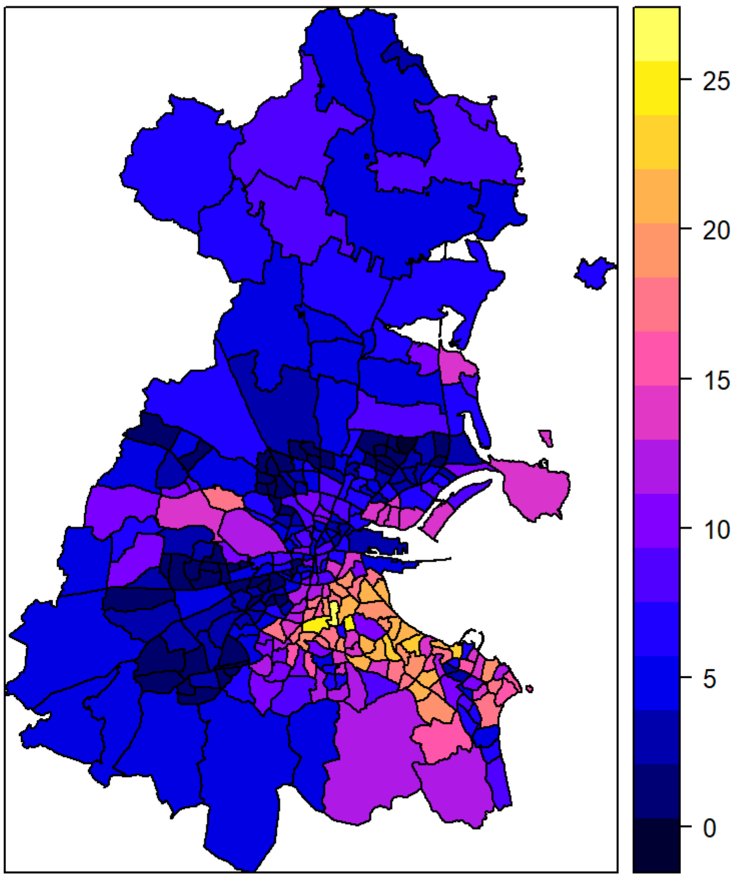
SC1 (Percentage of Social Class 1 people)

Now let's take a look at the range of values for this variable.

```
summary(Dub.voter$SC1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2202  3.2316  6.2096  8.0513 11.6889 25.6292
```

```
spplot(Dub.voter,"SC1")
```



The above results indicate a reasonable spread of valid percentage values for this variable. The maps shows that the affluence hotspots are located around Dublin Bay (with the exception of the central port areas) and in certain midwestern EDs.

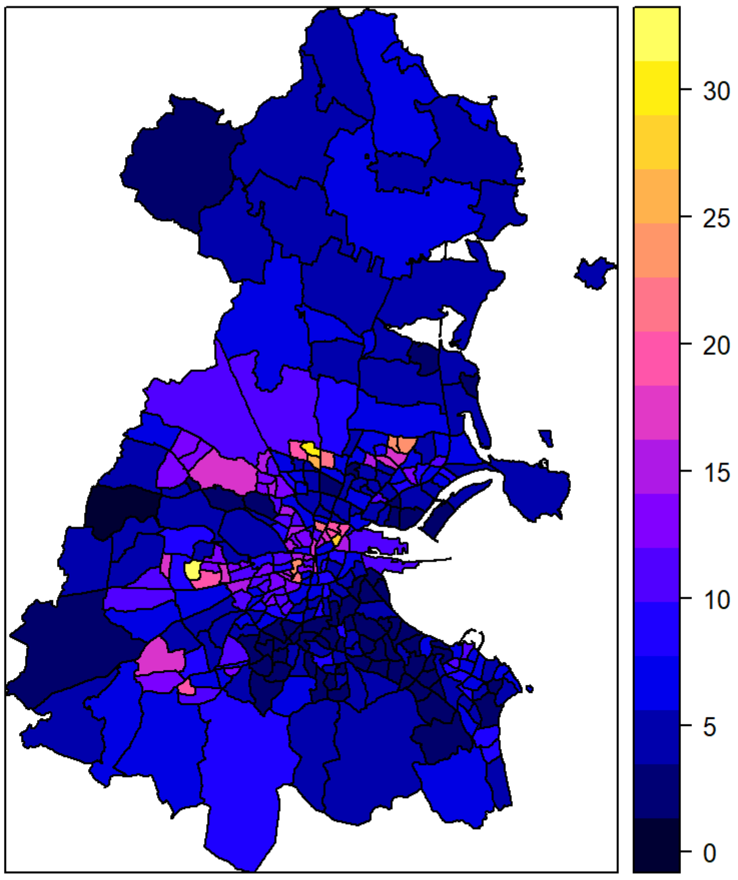
Unempl (Percentage of unemployed people)

Now let's take a look at the range of values for this variable.

```
summary(Dub.voter$Unempl)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.258   3.966   5.753   7.563  10.047  31.136
```

```
spplot(Dub.voter,"Unempl")
```



The above results indicate a reasonable spread of valid percentage values for this variable. The map shows that the unemployment blackspots are sprinkled around some north-central and midwestern EDs.

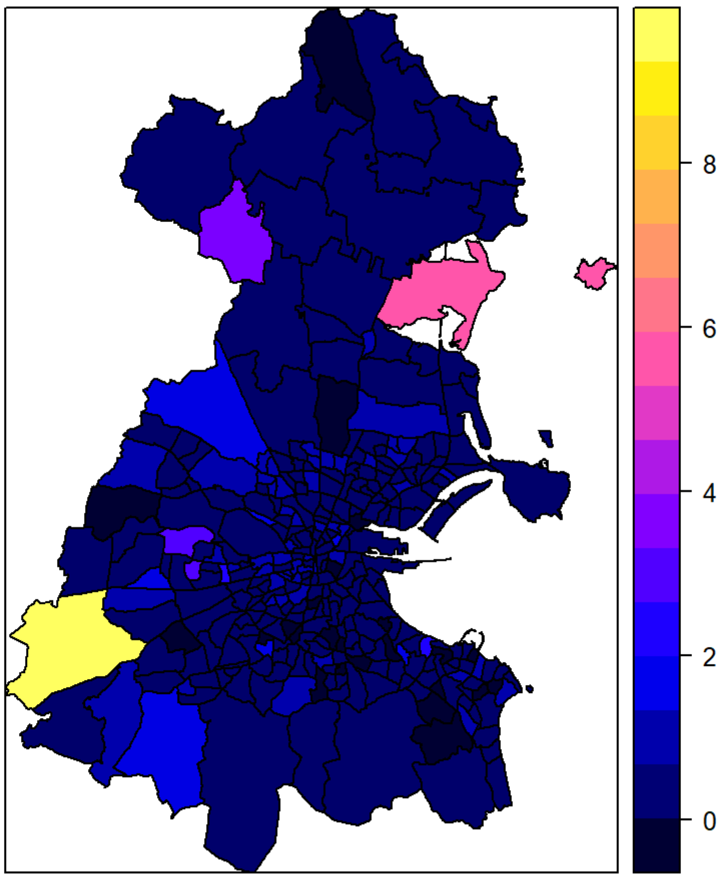
LowEduc (Percentage who changed address within the past year)

Now let's take a look at the range of values for this variable.

```
summary(Dub.voter$LowEduc)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.0000  0.1568  0.3324  0.4883  0.5856  9.2351
```

```
spplot(Dub.voter,"LowEduc")
```



The above results indicate a reasonable spread of valid percentage values for this variable. However, later in this document we will take a closer look at occurences of the minimum value (0) to assess the possibility of them being missing values. Th map shows only 2 main educational attainment hotpots in the outer northeastern and southwestern suburbs, respectively.

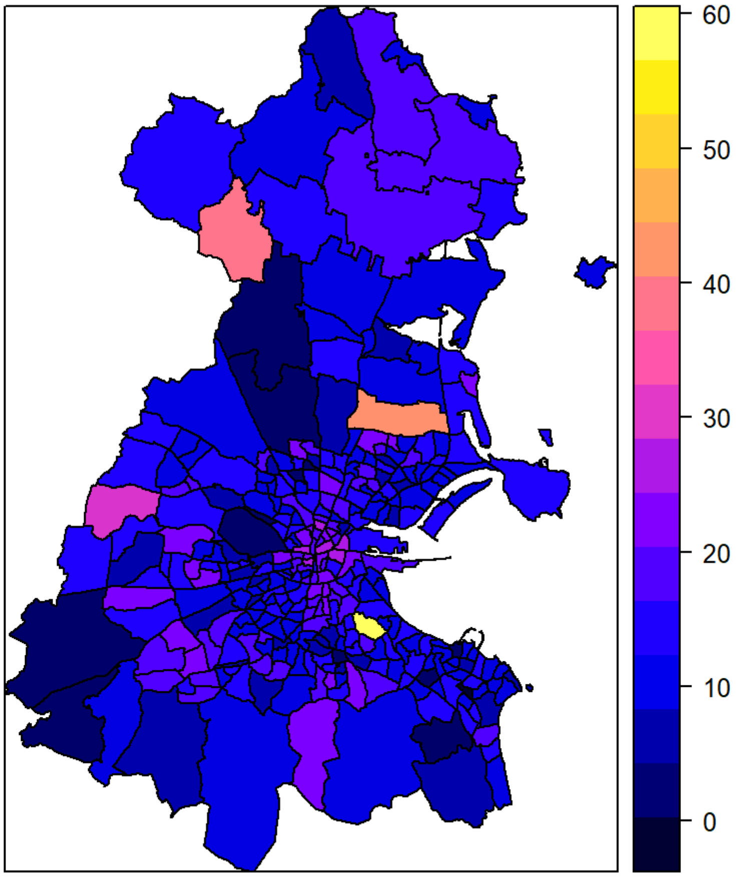
Age18_24 (Percentage in the 18-24 Age Group)

Now let's take a look at the range of values for this variable.

```
summary(Dub.voter$Age18_24)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1441  9.8865 12.7538 13.4328 16.5606 56.5482
```

```
spplot(Dub.voter,"Age18_24")
```



The above results indicate a reasonable spread of valid percentage values for this variable.The maps shows that some youthful hotspots are located in the city centre with a high-end on in the inner southern suburbes and in a small number of isolated outer suburbs.

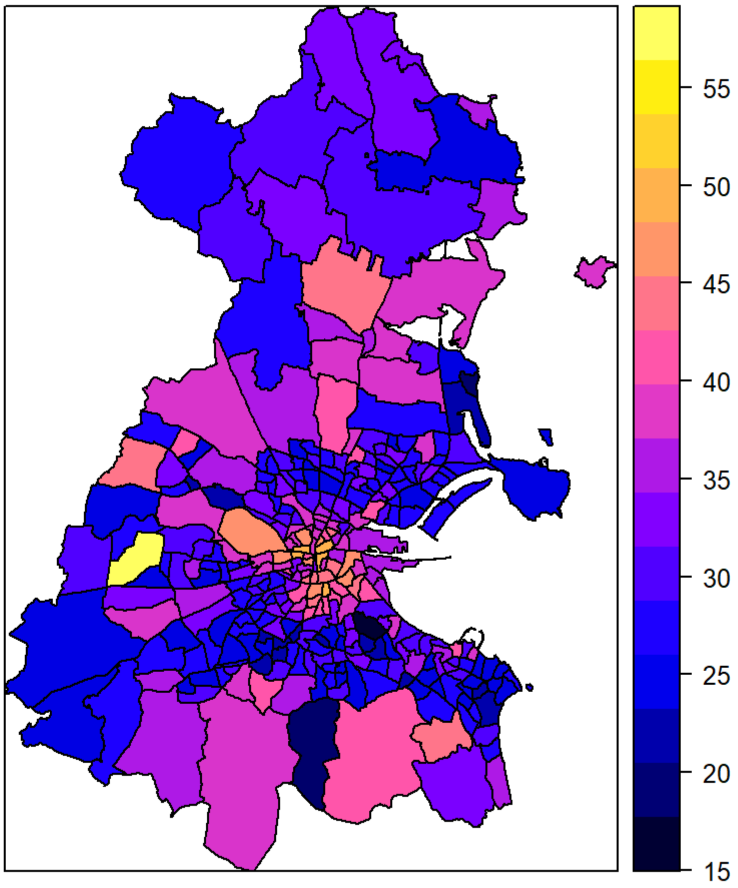
Age25_44 (Percentage in the 25-44 Age Group)

Now let's take a look at the range of values for this variable.

```
summary(Dub.voter$Age25_44)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   17.63   26.69   29.99   31.65   36.13   56.40
```

```
spplot(Dub.voter,"Age25_44")
```



The above results indicate a reasonable spread of valid percentage values for this variable. The map shows that millennials are concentrated in the city centre and in a broad arc in the outer suburbs.

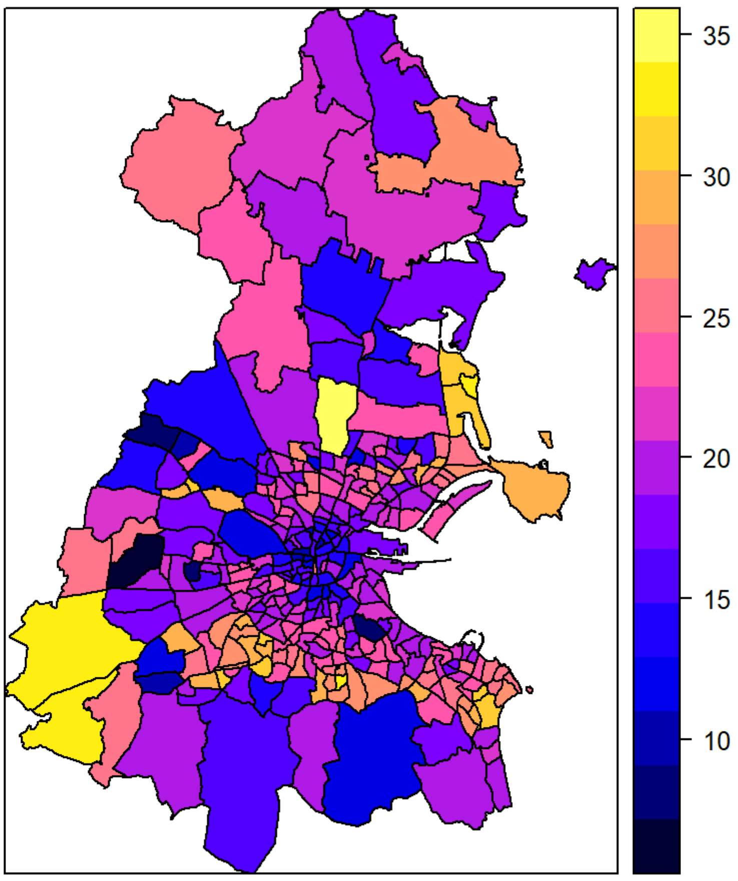
Age45_64 (Percentage in the 45-64 Age Group)

Now let's take a look at the range of values for this variable.

```
summary(Dub.voter$Age45_64)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   7.118  17.686  20.684  20.865  24.124  34.014
```

```
spplot(Dub.voter,"Age45_64")
```



The above results indicate a reasonable spread of valid percentage values for this variable. The map shows that the older generation are distributed across the EDs with some hotspots in the well-to-do suburbs at the north and south ends of Dublin Bay and its hinterland as well as some southwestern EDs.

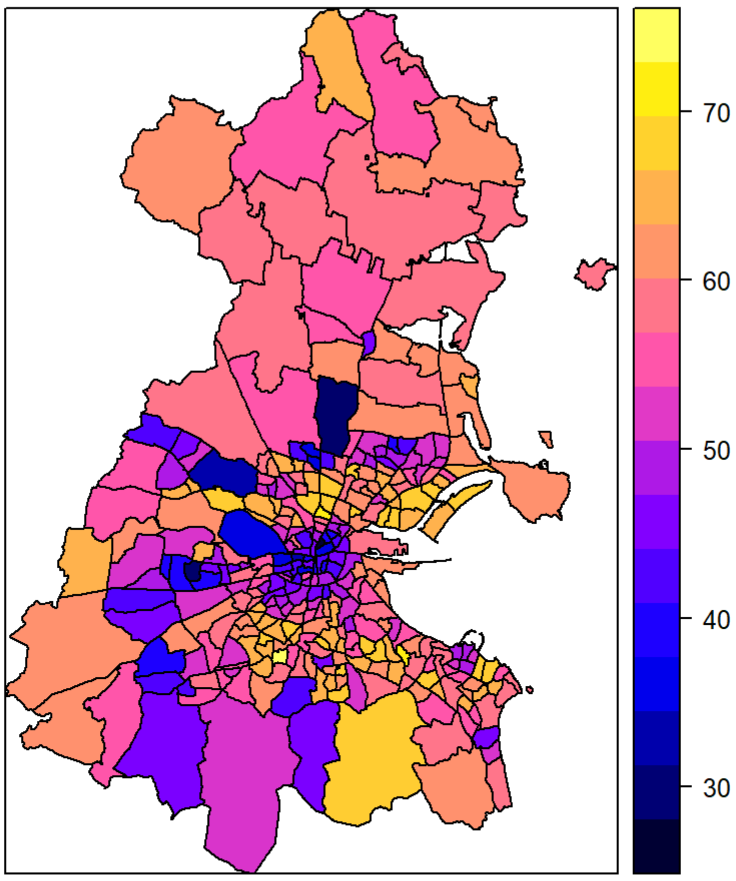
GenEl2004 (Percentage Turnout)

Now let’s take a look at the range of values for this variable.

```
summary(Dub.voter$GenEl2004)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   27.98   50.45   57.05   55.61   61.99   72.91
```

```
spplot(Dub.voter,"GenEl2004")
```



The above results indicate a reasonable spread of valid percentage values for this variable. This map shows a similar pattern to the previous one (re the older population) especially in the affluent suburbs at the southern and northern ends of Dulin Bay and its hinterland.

Missing Data Analysis

We will now look at the presence of missing date (NA) or potentially missing values (0) in this dataset.

```
miss_na<-function(x){sum(is.na(x))}      # Function to count NA values
miss_0<-function(x){sum(x==0)}           # Function to count zero values

apply(Dub.voter@data,2,miss_na)          # Count zero NA for all variables
```

```
##    DED_ID         X         Y   DiffAdd    LARent       SC1    Unempl 
##         0         0         0         0         0         0         0 
##   LowEduc  Age18_24  Age25_44  Age45_64 GenEl2004 
##         0         0         0         0         0
```

```
apply(Dub.voter@data,2,miss_0)           # Count zero values for all variables
```

```
##    DED_ID         X         Y   DiffAdd    LARent       SC1    Unempl 
##         0         0         0         0        12         0         0 
##   LowEduc  Age18_24  Age25_44  Age45_64 GenEl2004 
##        24         0         0         0         0
```

The results above show that NA values are not present for any variable, that the LARent variable has 12 occurences of the zero value, and the LowEduc variable has 24 occurrences of the zero value.

As the the proportion of zero values in both cases is relatively low, and because it is not unreasonable for the more prosperous EDs to have zero values for these 2 variables, we can conclude that those zero values do not represent missing values.

Outlier Analysis

Outlier analysis is not appropriate for the first 3 columns because the first one is just a unique identifier and the other two are coordinates which have already been implicitly checked for outliers by the previous map plotting exercise.

As we have not yet performed any variable (or model) selection, we will have to take a univariate approach to outlier detection (as opposed to a bivariate or multivariate one).

We will now perform univariate outlier detection using the following functions:

1. Boxplot() in the car library;
2. UnivariateOutlierDetection() in the OutlierDetection library.

In the case of the latter function, we will apply all of its available merthods (distance-, density- and depth-based) to identify the observations that are considered to be outliers by all of them.

Finally, we will merge the results of both functions to find the observations which are common to both of them and then take a closer look at all of them, based on the weight of evidence provided by the use of multiple methods performed by both functions.

Using Boxplot()

We will now display the observation (row) numbers where an outlier was detected for each variable:

```
# Get the observation numbers where an outlier was detected for each variable
out.box.list<-apply(Dub.voter@data[,4:12],2,Boxplot,id=list(cex=0.5))
```

```
str(out.box.list)
```

```
## List of 9
##  $ DiffAdd  : int [1:10] 79 3 280 231 152 252 120 66 138 139
##  $ LARent   : int [1:10] 18 73 75 88 89 117 119 121 144 145
##  $ SC1      : int [1:2] 136 282
##  $ Unempl   : int [1:10] 95 16 73 18 154 82 81 124 66 17
##  $ LowEduc  : int [1:10] 180 230 229 95 182 116 269 321 203 7
##  $ Age18_24 : int [1:6] 3 117 216 229 238 280
##  $ Age25_44 : int [1:2] 75 177
##  $ Age45_64 : int [1:2] 177 212
##  $ GenEl2004: int [1:5] 89 95 153 212 219
```

And the total number of outlier observations (overall and for unique observations):

```
# Get the total number of outlier observations
out.box<-unlist(out.box.list)
paste("Total (Overall) =",length(out.box))
```

```
## [1] "Total (Overall) = 57"
```

```
paste("Total (Unique)  =",length(unique(out.box)))
```

```
## [1] "Total (Unique)  = 44"
```

The results above show us that:

- A total of 57 outlier values are present across the entire dataset

    - in 44 of the 322 observations.
- At least 1 outlier value was detected for all of those variables.
- The maximum number of outliers per variable is 10.
- The minimum number of outliers per variable is 2.

Using UnivariateOutlierDetection()

We will now use this function to apply all of its methods to find all of outliers for each of the same variables as previously. Those methods are as follows:

- Robust Kernal-based Outlier Factor(RKOF) algorithm
- depthTools package
- Generalised Dispersion (using LOO dispersion matrix)
- Mahalanobis Distance (an observation and)based on the Chi square cutoff)
- k Nearest Neighbours Distance
- kth Nearest Neighbour Distance

Only the numbers of observations that are identified as outliers by all of those methods is returned to provide the required weight of evidence.

```
# This function runs all methods of UnivariateOutlierDetection() for the specified variable (x)
UOD<-function(x){
  obs<-try(UnivariateOutlierDetection(x,dist=T,depth=T,dens=T)$`Location of Outlier`)

  if (class(obs)=='try-error') {        # If no outliers found for a variable
    return(as.integer(NA))              #   return NA
  } else {                              # Otherwise
    return(obs)                         #   return their observation numbers
  }
}

out.uod.list<-apply(Dub.voter@data[,4:12],2,UOD)  # Invoke the function
```

```
## Error in quantile.default(bootdata, cutoff) :
##   missing values and NaN's not allowed if 'na.rm' is FALSE
## Error in quantile.default(bootdata, cutoff) :
##   missing values and NaN's not allowed if 'na.rm' is FALSE
```

```
str(out.uod.list)                                 # Display a summary of its results
```

```
## List of 9
##  $ DiffAdd  : int 79
##  $ LARent   : int NA
##  $ SC1      : int [1:4] 133 136 263 282
##  $ Unempl   : int [1:2] 16 95
##  $ LowEduc  : int NA
##  $ Age18_24 : int [1:8] 3 75 117 120 216 229 238 280
##  $ Age25_44 : int [1:3] 153 177 280
##  $ Age45_64 : int [1:3] 212 242 255
##  $ GenEl2004: int 184
```

And the total number of outlier observations (overall and for unique observations):

```
# Get the total number of outlier observations
out.uod<-unlist(out.uod.list)
paste("Total (Overall) =",length(out.uod))
```

```
## [1] "Total (Overall) = 24"
```

```
paste("Total (Unique)  =",length(unique(out.uod)))
```

```
## [1] "Total (Unique)  = 22"
```

The results above show us that, when all methods of UnivariateOutlierDetection() were applied:

- A total of 24 outlier values are present across the entire dataset
    - in 22 of the 322 observations.
- No outliers were detected for 2 variables.
- The maximum number of outliers per variable is 8.
- The minimum number of outliers per variable is 1.

We will now find the observation numbers that are common to both lists of outliers (as provided Boxplot() and UnivariateOutlierDetection(), respectively):

```
# This function matches the contents of two lists (l1 and l2) and returns the indexes of matching value
s for each element in the form of another list (l)
match.list<-function(l1,l2){
  l<-vector("list",length(l1))              # Initialise the output list
  names(l)<-names(l1)                       #  and name its elements (based on l1)

  for(i in c(1:length(l))){                 # Construct the output list
    v<-intersect(unlist(l1[i]),unlist(l2[i]))   # Find the matching elements of l1 & l2
    if(length(v)==0){v<-as.integer(NA)}     # If no matching element, use NA
    l[[i]]<-v                               # Load the current list element
  }

return(l)                                   # Return the output list
}

out.both.list<-match.list(out.box.list,out.uod.list)# Invoke the function
str(out.both.list)                                  # Display a summary of its results
```

```
## List of 9
##  $ DiffAdd  : int 79
##  $ LARent   : int NA
##  $ SC1      : int [1:2] 136 282
##  $ Unempl   : int [1:2] 95 16
##  $ LowEduc  : int NA
##  $ Age18_24 : int [1:6] 3 117 216 229 238 280
##  $ Age25_44 : int 177
##  $ Age45_64 : int 212
##  $ GenEl2004: int NA
```

And the total number of outlier observations (overall and for unique observations):

```
# Get the total number of outlier observations
out.both<-unlist(out.both.list)
paste("Total (Overall) =",sum(!is.na(out.both)))
```

```
## [1] "Total (Overall) = 13"
```

```
paste("Total (Unique)  =",sum(!is.na(unique(out.both))))
```

```
## [1] "Total (Unique)  = 13"
```

The results above show us that both functions have identified a total of 13 distinct outliers in common across the 322 observations. Let’s take a look at their values now:

```
# This function uses a list of index values (list) to provide the values of the corresponding cells of
 the specified data frame (df)
match.list.val<-function(list,df){
  l<-vector("list",length(list))                # Initialise the output list
  names(l)<-names(list)                         #   and name its elements based on list

  for(i in c(1:length(list))){                  # Construct the output list (l)
    v<-unlist(list[i])                          # Unlist the current index list element
    l[[i]]<-df[,i][v]                           # Load the output list
  }

  return(l)                                     # Return the output list
}

out.both.list.val<-match.list.val(out.both.list,# Invoke the function
                                  Dub.voter@data[,4:12])
str(out.both.list.val)                          # Display a summary of its results
```

```
## List of 9
##  $ DiffAdd  : num 34.7
##  $ LARent   : num NA
##  $ SC1      : num [1:2] 25.6 25.5
##  $ Unempl   : num [1:2] 31.1 29.2
##  $ LowEduc  : num NA
##  $ Age18_24 : num [1:6] 28 28.2 41.9 37.9 30.2 ...
##  $ Age25_44 : num 56.4
##  $ Age45_64 : num 34
##  $ GenEl2004: num NA
```

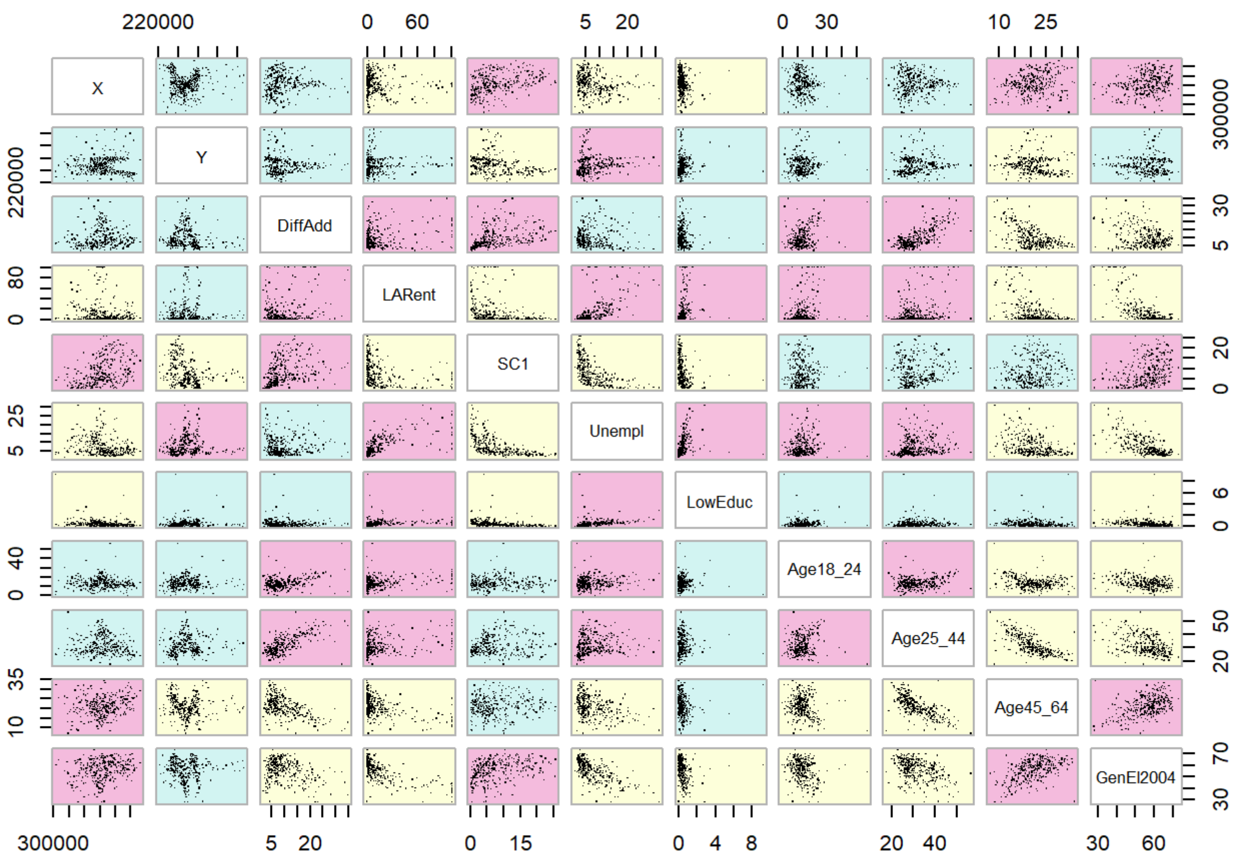
As of the values of the 13 (common) outliers appear to be unreasonable and they are rellatively small in number no further action is considered necessary.

Collinearity Analysis

Global Collinearity

Firstly, let’s see if we can detect any evidence of collinearity at the global level:

```
data.cor<-cor(Dub.voter@data[,2:12])
data.cor %>%
  dmat.color()->cols
cpairs(Dub.voter@data[,2:12],panel.colors=cols,pch=".",gap=.5)
```



The cpairs variant of the standard scatter plot matris (pair) provides a kind of heat map of the level of correlation exists between all variable pairings in the data set. It does this by cutting the correlation values of the variable pairings into 3 equal intervals (breaks): red for noteworthy positive correlation, cream for noteworthy negative correlation and blue for weak or no correlation.

As can be seen from its output above, there is evidence of quite a bit of multicollinearity in the dataset under review, which also applies to its spatial (x-Y coordinate) variables.

The following code will display the value intervals for the cream, blue and red colour codes above:

```
cormat<-cor(Dub.voter@data[,2:12])                   # Create the correlation matrix

n<-nrow(cormat)                                      # Get number of rows

for(i in c(1:n)){                                    # 'Blank' the redundant cells
  for(j in c(i:n)){
    cormat[i,j]<-NA
  }
}

cut<-cut_number(cormat,3)                            # Cut cormat into 3 ranges
levels(cut)                                          # Display the intervals/breaks
```

```
## [1] "[-0.693,-0.117]" "(-0.117,0.107]"  "(0.107,0.703]"
```

The above results tell us that the cream (noteworthy negative) interval covers Pearson r correlation values in the range -0.1 and below, the red (noteworthy positive) interval covers values in the range 0.1 and above, while the blue (weak or none) covers the range from 0.1 to -0.1.

As we will be looking at local collinearity in the next section of the document, for now we will focus solely on the non-spatial variables.

We will now take a closer look at the noteworthy correlations (both positive and negative) by producing a correlation matrix with a view to quantifying it in descending order of importance (as measured by the Pearson r values).

```
cormat<-cor(Dub.voter@data[,4:12])                   # Create the correlation matrix

n<-nrow(cormat)                                      # Get number of rows

for(i in c(1:n)){                                    # 'Blank' the redundant cells
  for(j in c(i:n)){
    cormat[i,j]<-NA
  }
}

cut<-cut_number(cormat,3)                            # Cut cormat into 3 ranges

corind<-which(as.numeric(cut) %in% c(1,3))           # Find cells with highest cor value

corrow<-corind %% n                                  # Convert cell index to row no.
corcol<-corind %/% n + 1                             # Convert cell index to col. no.

fix<-which(corrow==0)                                # corrow/corcol elements to fix
corrow[fix]<-n                                       # Fix corrow
corcol[fix]<-corcol[fix]-1                           # Fix corcol

rowname<-attr(cormat,"dimnames")[[1]][corrow]        # Convert row number to name
colname<-attr(cormat,"dimnames")[[2]][corcol]        # Convert column number to name
val<-cormat[corind]                                  # Get r value

cordf<-data.frame(rowname,colname,val)               # Create correlation data frame

arrange(cordf,desc(abs(val)))                        # Display correlation data frame
```

```
##       rowname  colname        val
## 1    Age25_44  DiffAdd  0.7030624
## 2    Age45_64 Age25_44 -0.6932300
## 3   GenEl2004   Unempl -0.6822627
## 4   GenEl2004   LARent -0.6806665
## 5      Unempl   LARent  0.6687762
## 6      Unempl      SC1 -0.5915679
## 7    Age45_64  DiffAdd -0.5612839
## 8   GenEl2004 Age45_64  0.4836208
## 9    Age45_64   LARent -0.4626929
## 10 GenEl2004 Age25_44 -0.4268253
## 11  Age45_64   Unempl -0.3742686
## 12       SC1  DiffAdd  0.3722988
## 13 GenEl2004      SC1  0.3517265
## 14  Age18_24  DiffAdd  0.3353004
## 15  Age25_44   LARent  0.3124497
## 16 GenEl2004  DiffAdd -0.3082693
## 17       SC1   LARent -0.2922633
## 18   LowEduc   Unempl  0.2828183
## 19    LARent  DiffAdd  0.2757630
## 20   LowEduc      SC1 -0.2727821
## 21 GenEl2004 Age18_24 -0.2597295
## 22  Age18_24   LARent  0.2524328
## 23   LowEduc   LARent  0.1675897
## 24  Age25_44   Unempl  0.1317413
```

```
                                                # (largest absolute r values first)
```

Now let's evaluate the top 10 (in terms of strength of correlation) for plausibility:

1. People in the 25-44 age group in an ED are the most mobile in terms of changing address (plausible).
2. EDs with the more people in the 45-64 age group have less in the 25-44 age bracket (plausible).
3. EDs with a high proportion of unemployed people have a lower turnout rate (plausible).
4. EDs with a high proportion of local authority renters have a lower turnout rate (plausible).
5. EDs with a high proportion of unemployed people also have a high level of local authority renters (plausible).
6. EDs with a high proportion of unemployed people have a lower proportion of well-off (SC1) people (plausible).
7. EDs with a high proportion of people in the 45-64 age group have a lower proportion of people who have changed their address within the past 12 months (plausible).
8. EDs with a high proportion of people in the 45-64 age group have a high turnout rate (plausible).
9. EDs with a high proportion of people in the 45-64 age group have low proportions of local authority renters (not so obvious).
10. EDs with a high proportion of people in the 25-44 age group have a low turnout rate (plausible).

Local Collinearity

We will now perform a Moran’s I test to see if the probability of local collinearity being present in the data is statistically significant for the variables of interest.

```
nbl<-poly2nb(Dub.voter)              # Create a neighbours list for the SPDF
print(nbl)
```

```
## Neighbour list object:
## Number of regions: 322
## Number of nonzero links: 1902
## Percentage nonzero weights: 1.83442
## Average number of links: 5.906832
```

```
wts<-nb2listw(nbl)                   # Create a weights matrix for the neighbours list
print(wts)
```

```
## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 322
## Number of nonzero links: 1902
## Percentage nonzero weights: 1.83442
## Average number of links: 5.906832
##
## Weights style: W
## Weights constants summary:
##     n     nn  S0       S1       S2
## W 322 103684 322 116.5416 1320.861
```

```
mt<-apply(Dub.voter@data[,4:12],      # Perform the moran test for the required vars.
          2,moran.test,wts)

v<-vector("double",length(mt))              # Initialise the output vector
names(v)<-names(Dub.voter@data[,4:12])      #   and name its elements based on vector

for(i in c(1:length(mt))){                  # Load the ouput vector
  v[i]<-mt[[i]][2]
}

str(v)                                      # Display a summary of the results
```

```
## List of 9
##  $ DiffAdd  : num 5.54e-56
##  $ LARent   : num 5.79e-56
##  $ SC1      : num 3.25e-100
##  $ Unempl   : num 5.02e-54
##  $ LowEduc  : num 0.202
##  $ Age18_24 : num 4.59e-09
##  $ Age25_44 : num 9.18e-47
##  $ Age45_64 : num 4.87e-31
##  $ GenEl2004: num 3.85e-47
```

```
v[which(v>0.05)]                            # Any high p-values?
```

```
## $LowEduc
## [1] 0.202077
```

The p-values resulting from the above test indicate that only one variable (LowEduc) has a p-value greater than 0.05 and, therefore, we cannot reject the null hypothesis for it (i.e. zero spatial autocorrelation present).

Corollarily, the null hypothesis can be rejected for the other variables which indicates that spatial autocorrelation is present for them.

Variable (Feature) Selection

Before we can begin modelling, we must first identify the set of predictors that are most likely to predict the response variable most accurately. In so doing, we must ensure that the model is not overfitted because its predictive power for other similar datasets (e.g. the same type voter data in other areas for the same election or in the same area for other elections) could be impaired.

For linear regression two of the most commonly used measures for assessing a model's goodness of fit are Mallow's C_p and the Akaike Information Criterion (AIC) which a designed to avoid overfitting.

Stepwise linear regression is the process of systemmatically evaluating models with different combinations of predictors by calculating the relevant goodness of fit metric for each one and choosing the combination the results in the lowest value.These techniques can typically be configured to run in a forward (starting with one predictor and systematically add others) or a backward direction (starting with all predictors and systematically remove predictors) or in both firections (sometimes referred to as an exhaustive approach).

R provides a number of different functions for performing stepwise regression of which we will now use the following ones (in both directions):

- regsubsets() in the leaps package (which uses C_p as its metric);
- steps() in the base stats package (which uses AIC as its metric);
- stepAIC() in the MASS package (which also uses AIC as its metric).

We will also use Geographically Weighted Principle Component Analysis (PCA) to sanity-check the outcome of the stepwise regression approach.

Using lm() and p-Values

But before we perform stepwise regression, let's fit a linear regression model for all predictors and look at the p-values for the predictors in the resultant fit which it considers to be the most significant variables statistically.

```
# Fit a model with all (9) predictors
f.lm <- lm(GenEl2004 ~ .,data = Dub.voter@data[,4:12])
summary(f.lm)
```

```
##
## Call:
## lm(formula = GenEl2004 ~ ., data = Dub.voter@data[, 4:12])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.9343  -3.3500   0.4952   3.4707  13.4373
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 77.70467    3.93928  19.726  < 2e-16 ***
## DiffAdd     -0.08583    0.08594  -0.999   0.3187
## LARent      -0.09402    0.01765  -5.326 1.92e-07 ***
## SC1          0.08637    0.07085   1.219   0.2238
## Unempl      -0.72162    0.09387  -7.687 1.96e-13 ***
## LowEduc     -0.13073    0.43022  -0.304   0.7614
## Age18_24    -0.13992    0.05480  -2.554   0.0111 *
## Age25_44    -0.35365    0.07450  -4.747 3.15e-06 ***
## Age45_64    -0.09202    0.09023  -1.020   0.3086
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.304 on 313 degrees of freedom
## Multiple R-squared:  0.6383, Adjusted R-squared:  0.629
## F-statistic: 69.03 on 8 and 313 DF,  p-value: < 2.2e-16
```

The results above indicate the following predictors are significant in descending order of probability: Unempl, LARent, Age25_44 and Age18_24. Interestingly, their coefficient estimate values are all negative which indicates that the higher their percentage values are in a particular ED the lower the turnout is likely to be there. Intuitively, this seems to make sense because turnout in deprived areas (where Unempl and LARent values can be relatively high) is expected to be lower and turnout is known to be higher among senior citizens relative to the younger age groups (as measured by the Age25_44 and Age45_64 predictors).

Let's go ahead and perform the types of stepwise regression referred to above to see how the combination of those 4 predictors compares with many other predictor combinations.

Using regsubsets() and C_p Values

Firstly, we run the stepwise regression function and store its key results:

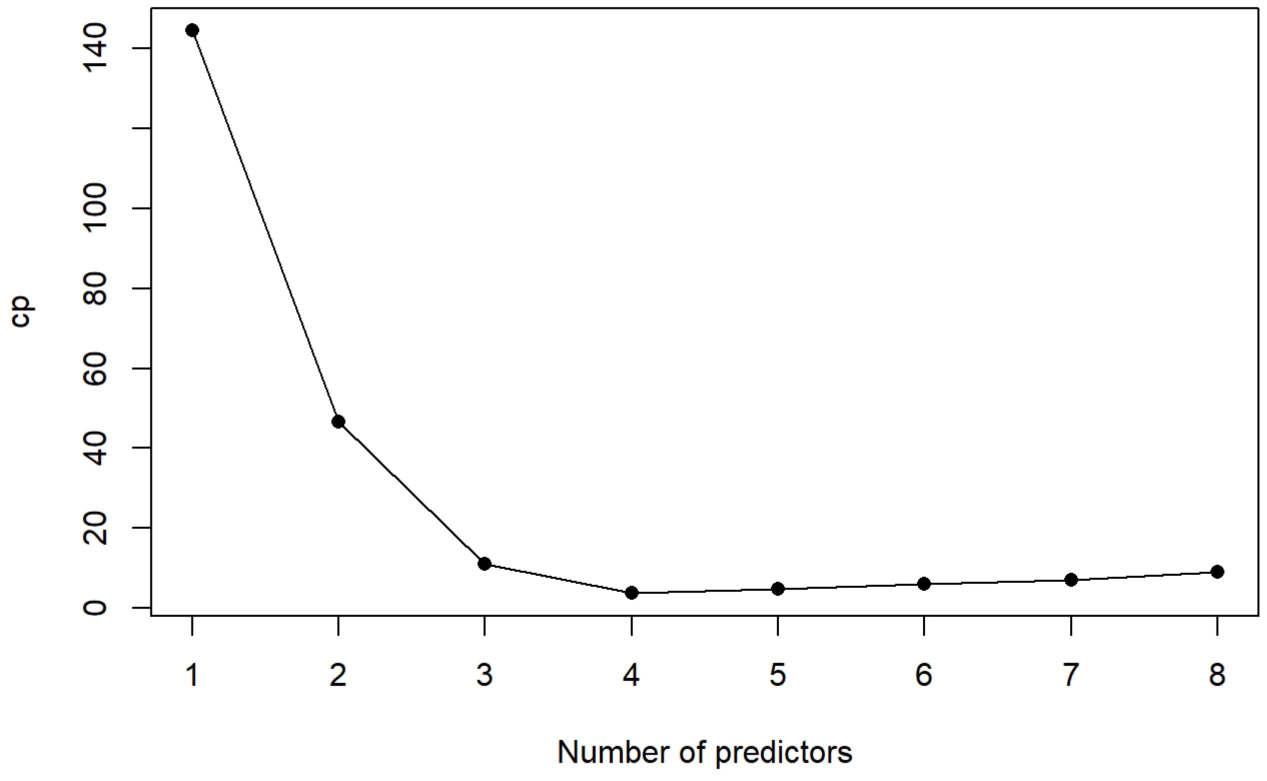
```
f.regs<-regsubsets(GenEl2004 ~ .,# Run the stepwise regression function
                   data=Dub.voter@data[,4:12])

f.regs.sum<-summary(f.regs)      # Store the results

cp<-f.regs.sum$cp                # Extract the Cp values for all stepwise models
np<-rowSums(f.regs.sum$which)-1  # Extract the number of predictors for each too
```

We now plot the C_p value achieved versus the number of predictors for the favoured predictor combination for each number of predictors.

```
# Plot the cp value against the number of predictors
plot(np, cp, pch=16, xlab="Number of predictors")
lines(np, cp)
```



The above plot indicates that best 4-predictor combination achieved the lowest C_p value.

Finally, let’s get the names of the predictors in that combination and display them:

```
# Extract the recommended variable names
terms<-attr(which(f.regs.sum$which[4,]),"names")
terms[2:length(terms)]
```

```
## [1] "LARent"   "Unempl"   "Age18_24" "Age25_44"
```

The results above agree with those achieved during the preliminary study using lm() and p-values.

So let’s see what lm()’s p-values look like now if we fit a model with only those 4 predictors:

```
# Fit a model with all (9) predictors
f.lm.4 <- lm(GenEl2004 ~ LARent + Unempl + Age18_24 + Age25_44,
             data = Dub.voter@data[,4:12])
summary(f.lm.4)
```

```
##
## Call:
## lm(formula = GenEl2004 ~ LARent + Unempl + Age18_24 + Age25_44,
##     data = Dub.voter@data[, 4:12])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -24.9394  -3.1431   0.6102   3.4391  12.9620
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 75.89190    1.66021  45.712  < 2e-16 ***
## LARent      -0.09229    0.01731  -5.332 1.85e-07 ***
## Unempl      -0.75748    0.07590  -9.980  < 2e-16 ***
## Age18_24    -0.15423    0.05055  -3.051  0.00247 **
## Age25_44    -0.35002    0.04645  -7.535 5.16e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.293 on 317 degrees of freedom
## Multiple R-squared:  0.6351, Adjusted R-squared:  0.6305
## F-statistic: 137.9 on 4 and 317 DF,  p-value: < 2.2e-16
```

The p-values in the results above show (not unexpectedly) that the statistical significance of those 4 predictors has increased, especially for Age18_24, although their relative significance has changed slightly in terms of descending order of p-values: Unempl, Age25-44, LARent and Age18_24 (LARrent has swapped position with Age25_44). All of their coefficient estimates remain in negative territory.

Using Step() and AIC Values

Now let's run this stepwise regression function, which uses AIC instead of C_p as its goodness of fit measurement and the all-predictor regression model as its starting point, and display its results:

```
# Perform stepwise model fitting (direction=backward by default)
f.step<-step(f.lm, direction="both", trace=F)

# Summarise the predictors that are recommended to be dropped
f.step$anova
```

```
##         Step Df  Deviance Resid. Df Resid. Dev      AIC
## 1            NA        NA       313   8805.251 1083.354
## 2  - LowEduc  1  2.597484       314   8807.849 1081.449
## 3  - DiffAdd  1 28.981394       315   8836.830 1080.506
## 4 - Age45_64  1 21.719449       316   8858.549 1079.297
## 5      - SC1  1 23.936140       317   8882.486 1078.166
```

The results above recommend dropping the following predictors: LowEduc, DiffAdd, Age45_64 and SC1, leaving us with LARent, Unempl, Age18_24 and Age25_44 as the recommended predictors yet again.

Using StepAIC() aand AIC Values

Now let's try anotherAIC-based stepwise regression function to double-check the previous results.

```
# Doublechecking using the stepAIC function in the MASS library....
f.step.AIC<-stepAIC(f.lm, direction="both", trace=F)

# Summarise the predictors that are recommended to be dropped
f.step.AIC$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## GenEl2004 ~ DiffAdd + LARent + SC1 + Unempl + LowEduc + Age18_24 +
##     Age25_44 + Age45_64
##
## Final Model:
## GenEl2004 ~ LARent + Unempl + Age18_24 + Age25_44
##
##
##         Step Df  Deviance Resid. Df Resid. Dev      AIC
## 1                               313   8805.251 1083.354
## 2  - LowEduc  1  2.597484       314   8807.849 1081.449
## 3  - DiffAdd  1 28.981394       315   8836.830 1080.506
## 4 - Age45_64  1 21.719449       316   8858.549 1079.297
## 5      - SC1  1 23.936140       317   8882.486 1078.166
```

The results above also recommend dropping the following predictors: LowEduc, DiffAdd, Age45_64 and SC1, leaving us with LARent, Unempl, Age18_24 and Age25_44 as the recommended predictors yet again.

Using gwpca() and Loading Values

We will firstly use the bw.gwpca() for establish the optimum bandwidth value (k) and then run the gwpca() itself to perform the PCA, after which we will display its results.

```
bw.choice<-bw.gwpca(Dub.voter,vars=names(Dub.voter)[4:11],k=2)
```

```
## Fixed bandwidth: 27043.27 CV score: 1243801
## Fixed bandwidth: 16717 CV score: 1243038
## Fixed bandwidth: 10335.02 CV score: 1435256
## Fixed bandwidth: 20661.29 CV score: 1225259
## Fixed bandwidth: 23098.99 CV score: 1232203
## Fixed bandwidth: 19154.7 CV score: 1226099
## Fixed bandwidth: 21592.41 CV score: 1226277
## Fixed bandwidth: 20085.82 CV score: 1225945
## Fixed bandwidth: 21016.94 CV score: 1225343
## Fixed bandwidth: 20441.48 CV score: 1225422
## Fixed bandwidth: 20797.13 CV score: 1225241
## Fixed bandwidth: 20881.09 CV score: 1225262
## Fixed bandwidth: 20745.25 CV score: 1225241
## Fixed bandwidth: 20713.18 CV score: 1225245
## Fixed bandwidth: 20765.07 CV score: 1225240
## Fixed bandwidth: 20777.31 CV score: 1225240
## Fixed bandwidth: 20757.49 CV score: 1225240
## Fixed bandwidth: 20769.74 CV score: 1225240
## Fixed bandwidth: 20762.17 CV score: 1225240
## Fixed bandwidth: 20766.85 CV score: 1225240
## Fixed bandwidth: 20763.96 CV score: 1225240
## Fixed bandwidth: 20765.75 CV score: 1225240
## Fixed bandwidth: 20766.17 CV score: 1225240
## Fixed bandwidth: 20765.49 CV score: 1225240
```

```
pca.gw.auto<-gwpca(Dub.voter,vars=names(Dub.voter)[4:11],bw=bw.choice,k=2,scores=T)
pca.gw.auto
```

```
##    ***********************************************************************
##    *                       Package   GWmodel                             *
##    ***********************************************************************
##    Program starts at: 2019-05-09 12:07:56
##    Call:
##
##    Variables concerned:  DiffAdd LARent SC1 Unempl LowEduc Age18_24 Age25_44 Age45_64
##    The number of retained components:  2
##    Number of data points: 322
##    ***********************************************************************
##    *                Results of Principal Components Analysis               *
##    ***********************************************************************
## Importance of components:
##                           Comp.1    Comp.2    Comp.3    Comp.4     Comp.5
## Standard deviation     1.6990453 1.4307196 0.9765114 0.9178398 0.74245890
## Proportion of Variance 0.3608444 0.2558698 0.1191968 0.1053037 0.06890565
## Cumulative Proportion  0.3608444 0.6167142 0.7359110 0.8412147 0.91012038
##                            Comp.6     Comp.7     Comp.8
## Standard deviation     0.54257250 0.49891474 0.41920892
## Proportion of Variance 0.03679812 0.03111449 0.02196701
## Cumulative Proportion  0.94691850 0.97803299 1.00000000
##
## Loadings:
##          Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
## DiffAdd   0.389  0.444  0.004  0.149  0.123  0.293  0.445  0.575
## LARent    0.441 -0.226 -0.144 -0.172  0.612  0.149 -0.539  0.132
## SC1      -0.130  0.576  0.030  0.135  0.590 -0.343  0.076 -0.401
## Unempl    0.361 -0.462 -0.022 -0.189  0.197 -0.085  0.670 -0.355
## LowEduc   0.131 -0.308  0.362  0.861  0.079 -0.062 -0.065 -0.011
## Age18_24  0.237  0.080 -0.845  0.359 -0.224 -0.051 -0.045 -0.200
## Age25_44  0.436  0.302  0.317 -0.053 -0.291  0.448 -0.177 -0.546
## Age45_64 -0.493 -0.118 -0.179  0.144  0.289  0.748  0.142 -0.164
##
##    ***********************************************************************
##    *   Results of Geographically Weighted Principal Components Analysis  *
##    ***********************************************************************
##
##    *********************Model calibration information*********************
##    Kernel function for geographically weighting: bisquare
##    Fixed bandwidth for geographically and temporally weighting:  20765.49
##    Distance metric for geographically weighting: A distance matrix is specified for this model calib
ration.
##
##    ****************     Summary of GWPCA information:    *****************
##    Local variance:
##                Min.   1st Qu.    Median   3rd Qu.    Max.
##    Comp.1    2.5510  790.5018 1052.8927 1191.1831 1278.34
##    Comp.2    1.7216   95.8119  125.8093  140.9405  150.68
##    Local Proportion of Variance:
##                  Min. 1st Qu.  Median 3rd Qu.   Max.
##    Comp.1     37.2657 79.5913 80.4900 81.0288 81.915
##    Comp.2      8.6540  9.3803  9.6280  9.8821 32.497
##    Cumulative 69.7630 89.4877 90.0839 90.3740 90.986
##
##    ***********************************************************************
##    Program stops at: 2019-05-09 12:07:57
```

The results above consider the first two components to be the most important because they collectively explain almost 62% of the global variance in the voter data and about 90% of the local variance (on average). The loadings for the first component put its focus on the young (Age18_24 and Age25_44), mobile (DiffAdd) and less well off (LARent and Unempl) while down-weighting the well off (SC1) and older people (Age45_64).

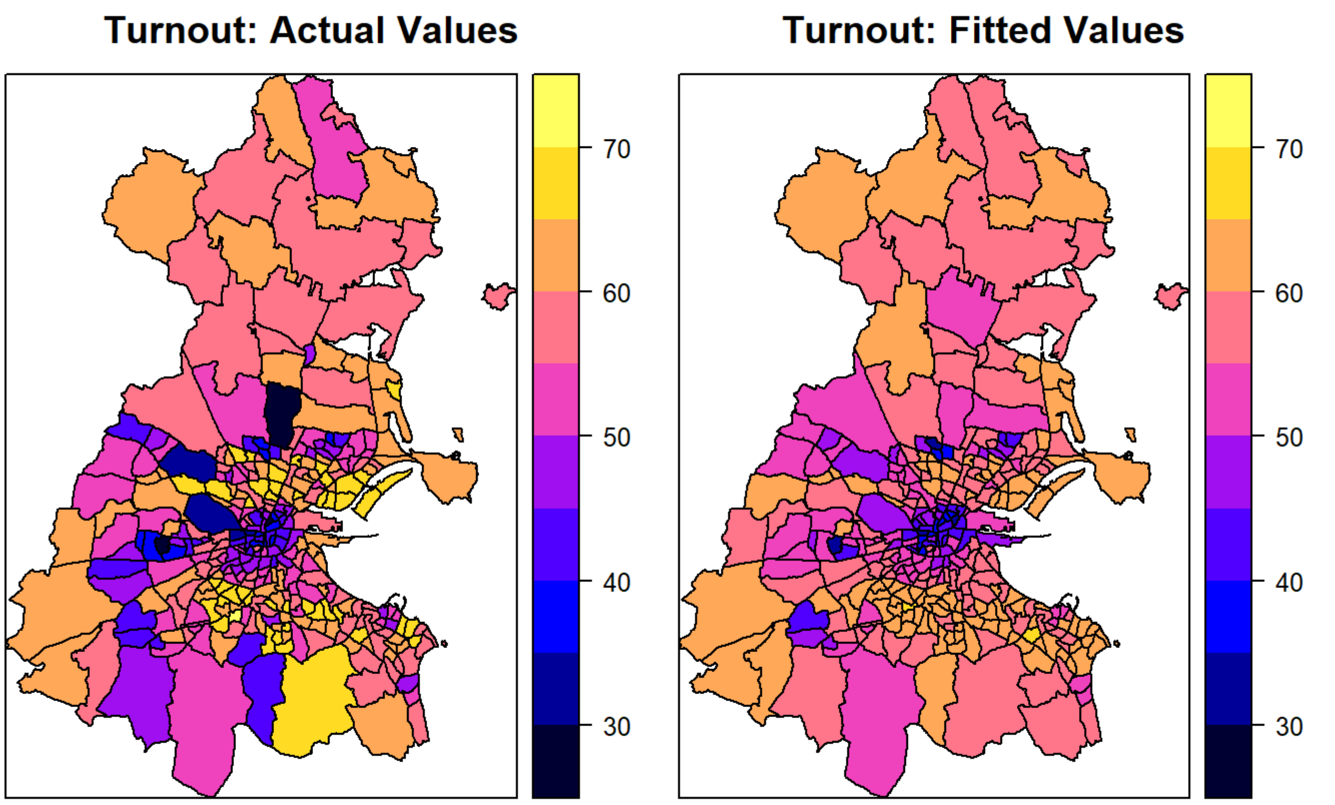
Age18_24 and Age25_44 are the only predictors with a positive weight for both components so that seems to support its inclusion in the predictor set recommended by our stepwise regression exercises above. Age45_64 is the only predictor that has negative weightings for both components so that appears to support its elimination by stepwise regression. DiffAdd has sizable positive weightings for both components but the fact that it has the highest correlation (0.70) of any variable (with Age25_44) could explain why it was eliminated when Age24_45 was included. It is unclear what justification has to offer for the selection of the Unempl and LARent apart from the fact that LARent has the second highest positive weighting for the first component and, after DiffAdd has been discounted, Unempl has the third highest.

On balance, we will go with the recommended predictor set of LARent, Unempl, Age18_24 and Age25_44.

Model Evaluation

As a prelude to performing a proper cross-valisation test of the recommended model (which is outside the scope of this document), we will perform a graphical comparison of its predicted (fitted) values with their actual equivalents (using the same scale) as follows:

```
Dub.voter@data$Fitted<-fitted(f.lm.4)
p1<-spplot(Dub.voter,"GenEl2004",main="Turnout: Actual Values",
           at=seq(25,75,5))
p2<-spplot(Dub.voter,"Fitted",main="Turnout: Fitted Values",
           at=seq(25,75,5))
grid.arrange(p1,p2,nrow=1,ncol=2)
```



The maps above show that our oLS-based linear regression model has not done a particularly good job at predicting, so it certainly cannot be accused of overfitting. Whle it does appear to have done a fair job in some EDs in the north and southwest of the county, the relative lack of blue areas in the fitted map suggests that it has particular difficulty with predicting where very low turnout rates will occur. For instance, it completely missed that dark blue (very low) outlier in an ED just north of the city centre in the actual turnout map. It also missed that yellow (very high) outlier in an ED in the southeast corner of the county, so it seems to have problems at that end too.

Let's now do a very high-level comparison of the numbers undelying both of the graphs above:

```
summary(Dub.voter$GenEl2004)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   27.98   50.45   57.05   55.61   61.99   72.91
```

```
summary(Dub.voter$Fitted)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   30.21   53.13   57.25   55.61   60.61   65.31
```

Even though the median and mean values for both the actual and fitted turnout value sets are exactly the same (to all intents and purposes), as expect, the results above confirm that the model is overestimating at the low end and underestimating at the top end.

Here are some approaches that could be taken toward creating an improved model (which are outside the scope of this document):

1. Take another look at the classification of outliers and at what might be done to lessen any influence that they might be having on the model;
2. Try using Geographically Weighted Regression (the robust version to cope with outliers) instead of Multiple Linear Regression (for both predictor selection and model fitting);
3. Try a different (e.g. non-linear) predictor selection method like Random Forest;
4. Try a modelling technique that does not rely on linearity such as K Nearest Neighbours (KNN).