Practical 9: Measuring Spatial Autocorrelation in R

An Introduction to Spatial Data Analysis and Visualisation in R - Guy Lansley & James Cheshire (2016)

This practical will cover how to run various measures of spatial autocorrelation in R. We will consider both statistics of global spatial autocorrelation and how to identify spatial clustering across our study area. Data for the practical can be downloaded from the **Introduction to Spatial Data Analysis and Visualisation in R** (https://data.cdrc.ac.uk/tutorial/an-introduction-to-spatial-data-analysis-and-visualisation-in-r) homepage.

In this practical we will:

- Run a global Spatial autocorrelation for a shapefile
- Identify local indicators of spatial autocorrelation
- Run a Getis-Ord

First, we must set the working directory and load the practical data.

```
# Set the working directory
setwd("C:/Users/Guy/Documents/Teaching/CDRC/Practicals")

# Load the data. You may need to alter the file directory
Census.Data <-read.csv("practical_data.csv")
```

We will also need to load the spatial data files from the previous practicals.

```
# load the spatial libraries
library("sp")
library("rgdal")
library("rgeos")

# Load the output area shapefiles
Output.Areas <- readOGR(".", "Camden_oa11")
```

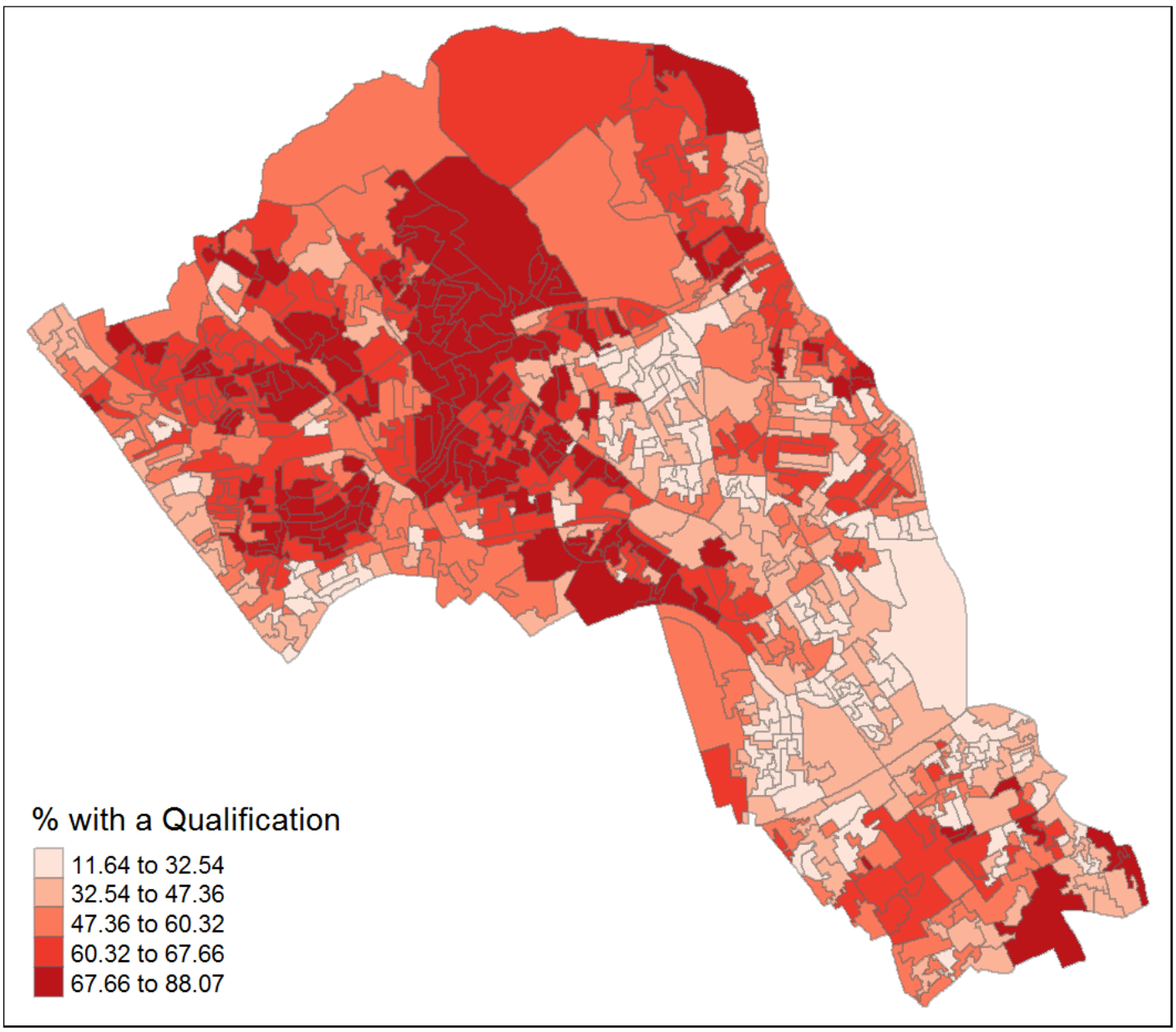
```
## OGR data source with driver: ESRI Shapefile
## Source: ".", layer: "Camden_oa11"
## with 749 features
## It has 1 fields
```

```
# join our census data to the shapefile
OA.Census <- merge(Output.Areas, Census.Data, by.x="OA11CD", by.y="OA")

# load the houses point files
House.Points <- readOGR(".", "Camden_house_sales")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: ".", layer: "Camden_house_sales"
## with 2547 features
## It has 4 fields
```

Remember the distribution of our qualification variable? We will be working on that today. We have first mapped it to remind us of its spatial distribution across our study area.

```
library("tmap")

tm_shape(OA.Census) + tm_fill("Qualification", palette = "Reds", style = "quantile
", title = "% with a Qualification") + tm_borders(alpha=.4)
```



Running a spatial autocorrelation

A spatial autocorrelation (http://www.spatialanalysisonline.com/HTML/?spatial_autocorrelation.htm) measures how distance influences a particular variable. In other words, it quantifies the degree of which objects are similar to nearby objects. Variables are said to have a positive spaital autocorrelation when similar values tend to be nearer together than dissimilar values.

Waldo Tober's first law of geography is that *"Everything is related to everything else, but near things are more related than distant things."* so we would expect most geographic phenomena to exert a spatial autocorrelation of some kind. In population data this is often the case as persons of similar characteristics

tend to reside in similar neighbourhoods due to a range of reasons including house prices, proximity to work places and cultural factors.

We will be using the spatial autocorrelation functions available from the spdep package.

```
library(spdep)
```

Finding neighbours

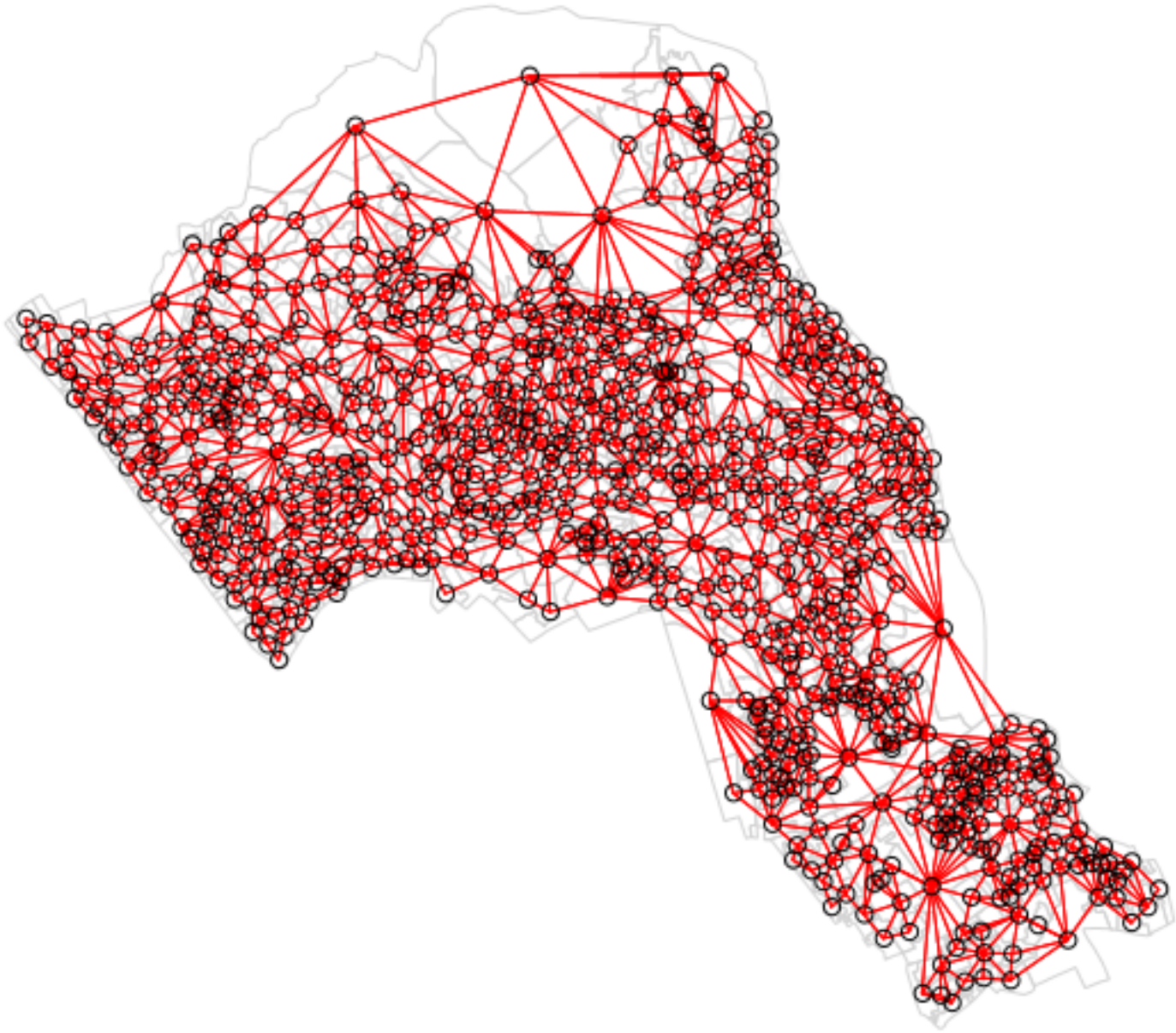
In order for the subsequent model to work, we need to work out what polygons neighbour each other. The following code will calculate neighbours for our *OA.Census* polygon and print out the results below.

```
# Calculate neighbours
neighbours <- poly2nb(OA.Census)
neighbours
```

```
## Neighbour list object:
## Number of regions: 749
## Number of nonzero links: 4342
## Percentage nonzero weights: 0.7739737
## Average number of links: 5.797063
```

We can plot the links between neighbours to visualise their distribution across space.

```
plot(OA.Census, border = 'lightgrey')
plot(neighbours, coordinates(OA.Census), add=TRUE, col='red')
```

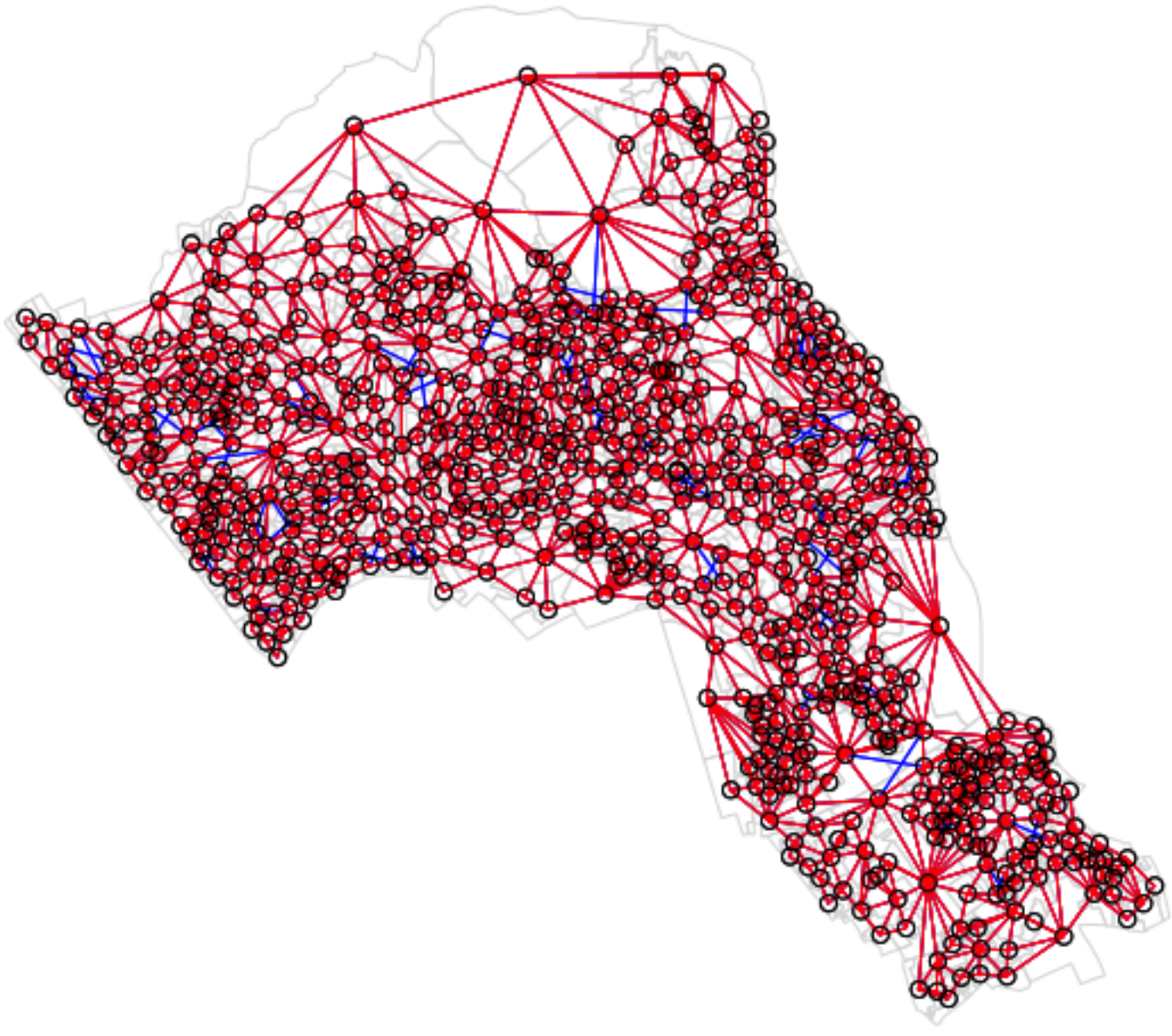


```
# Calculate the Rook's case neighbours
neighbours2 <- poly2nb(OA.Census, queen = FALSE)
neighbours2
```

```
## Neighbour list object:
## Number of regions: 749
## Number of nonzero links: 4176
## Percentage nonzero weights: 0.7443837
## Average number of links: 5.575434
```

We can already see that this approach has identified fewer links between neighbours. By plotting both neighbour outputs we can interpret their differences.

```
# compares different types of neighbours
plot(OA.Census, border = 'lightgrey')
plot(neighbours, coordinates(OA.Census), add=TRUE, col='blue')
plot(neighbours2, coordinates(OA.Census), add=TRUE, col='red')
```



We can represent spatial autocorrelation in two ways; globally or locally. Global models (http://www.spatialanalysisonline.com/HTML/?global_spatial_autocorrelation.htm) will create a single measure which represent the entire data whilst local models (http://www.spatialanalysisonline.com/HTML/?local_indicators_of_spatial_as.htm) let us explore spatial clustering across space.

Running a global spatial autocorrelation

With the neighbours defined. We can now run a model. First we need to convert the data types of the neighbours object. This file will be used to determine how the neighbours are weighted

```
# Convert the neighbour data to a listw object
listw <- nb2listw(neighbours2)
listw
```

```
## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 749
## Number of nonzero links: 4176
## Percentage nonzero weights: 0.7443837
## Average number of links: 5.575434
##
## Weights style: W
## Weights constants summary:
##     n     nn  S0       S1       S2
## W 749 561001 749 285.3793 3113.982
```

We can now run the model. This type of model is known as a Moran's test. This will create a correlation score between -1 and 1. Much like a correlation coefficient, 1 determines perfect postive spatial autocorrelation (so our data is clustered), 0 identifies the data is randomly distributed and -1 represents negative spatial autocorrelation (so dissimilar values are next to each other).

```
# global spatial autocorrelation
moran.test(OA.Census$Qualification, listw)
```

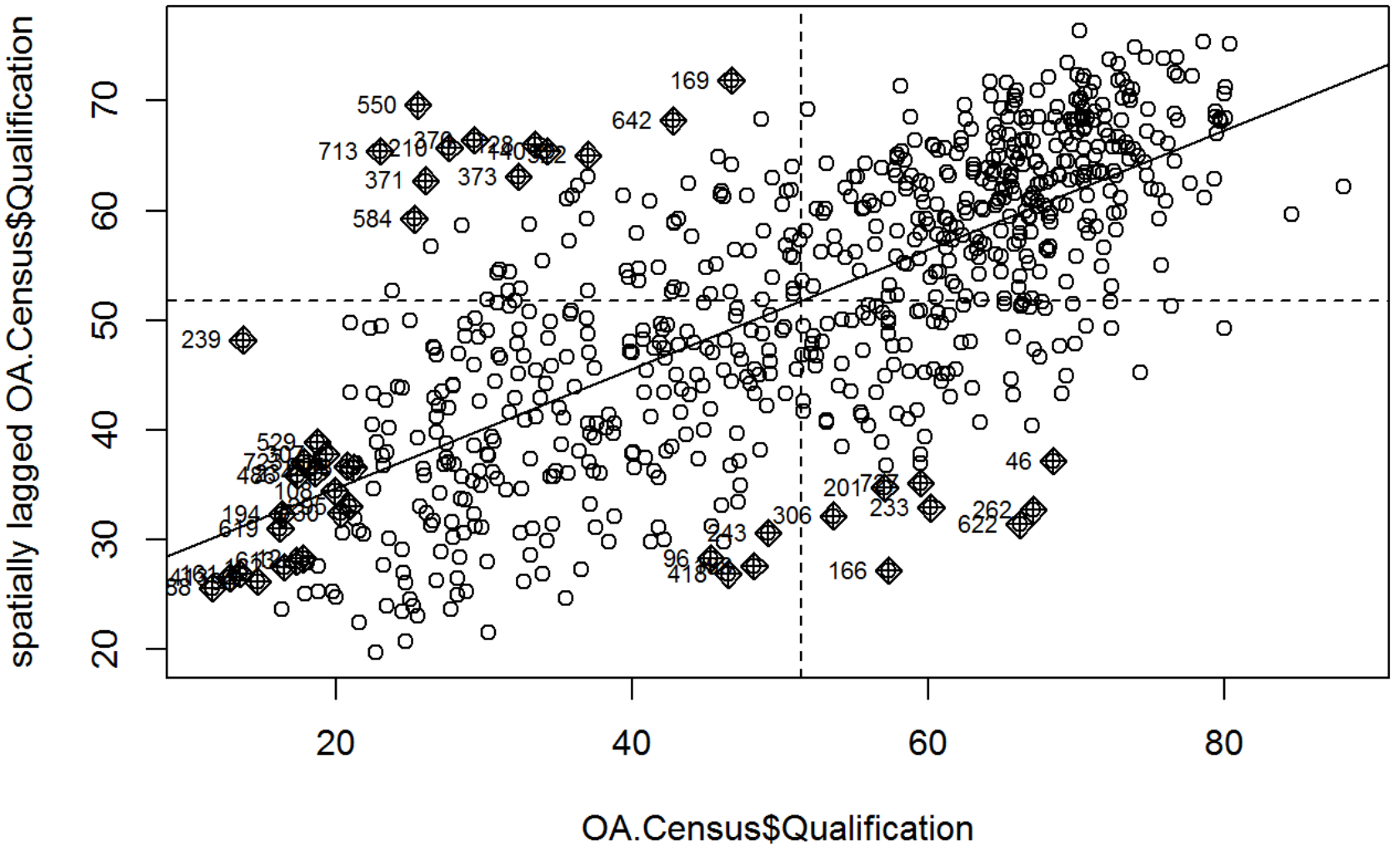
```
##
##  Moran I test under randomisation
##
## data:  OA.Census$Qualification
## weights: listw
##
## Moran I statistic standard deviate = 24.292, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      0.5448699398     -0.0013368984      0.0005055733
```

The Moran I statistic is 0.54, we can therefore determine that there our qualification variable is positively autocorrelated in Camden. In other words, the data does spatially cluster. We can also consider the p-value as a measure of the statistical significance of the model.

Running a local spatial autocorrelation

We will first create a moran plot which looks at each of the values plotted against their spatially lagged values. It basically explores the relationship between the data and their neighbours as a scatter plot. The style refers to how the weights are coded. "W" weights are row standardised (sums over all links to n).

```
# creates a moran plot
moran <- moran.plot(OA.Census$Qualification, listw = nb2listw(neighbours2, style =
"W"))
```



Is it possible to determine a positive relationship from observing the scatter plot?

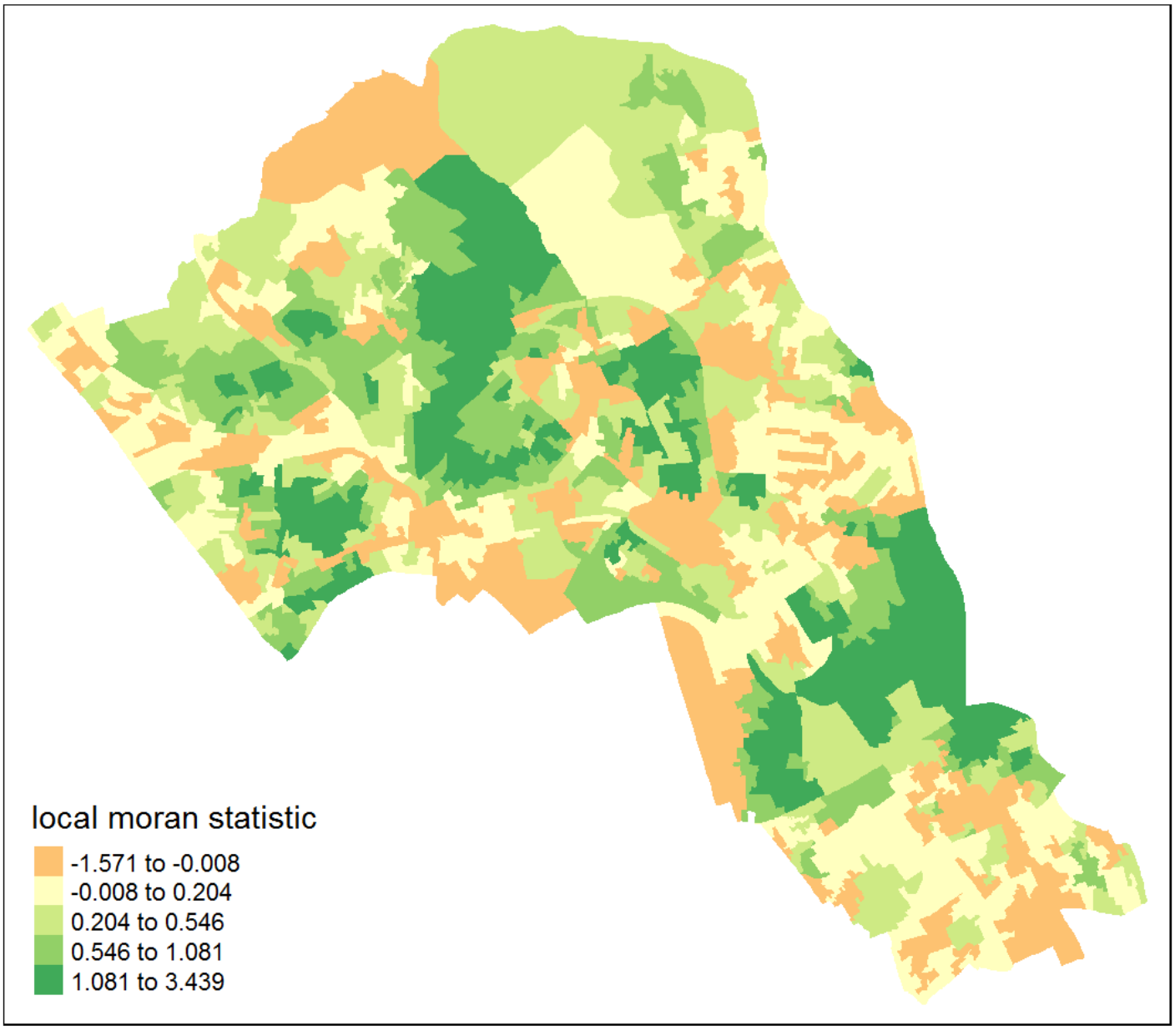
```
# creates a local moran output
local <- localmoran(x = OA.Census$Qualification, listw = nb2listw(neighbours2, sty
le = "W"))
```

By considering the help page for the localmoran function (run `?localmoran` in R) we can observe the arguments and outputs. We get a number of useful statistics from the model which are as defined:

Name	Description
Ii	local moran statistic
E.Ii	expectation of local moran statistic
Var.Ii	variance of local moran statistic
Z.Ii	standard deviate of local moran statistic
Pr()	p-value of local moran statistic

First we will map the local moran statistic (*Ii*). A positive value for *Ii* indicates that the unit is surrounded by units with similar values.

```
# binds results to our polygon shapefile
moran.map <- cbind(OA.Census, local)


# maps the results
tm_shape(moran.map) + tm_fill(col = "Ii", style = "quantile", title = "local moran
statistic")
```

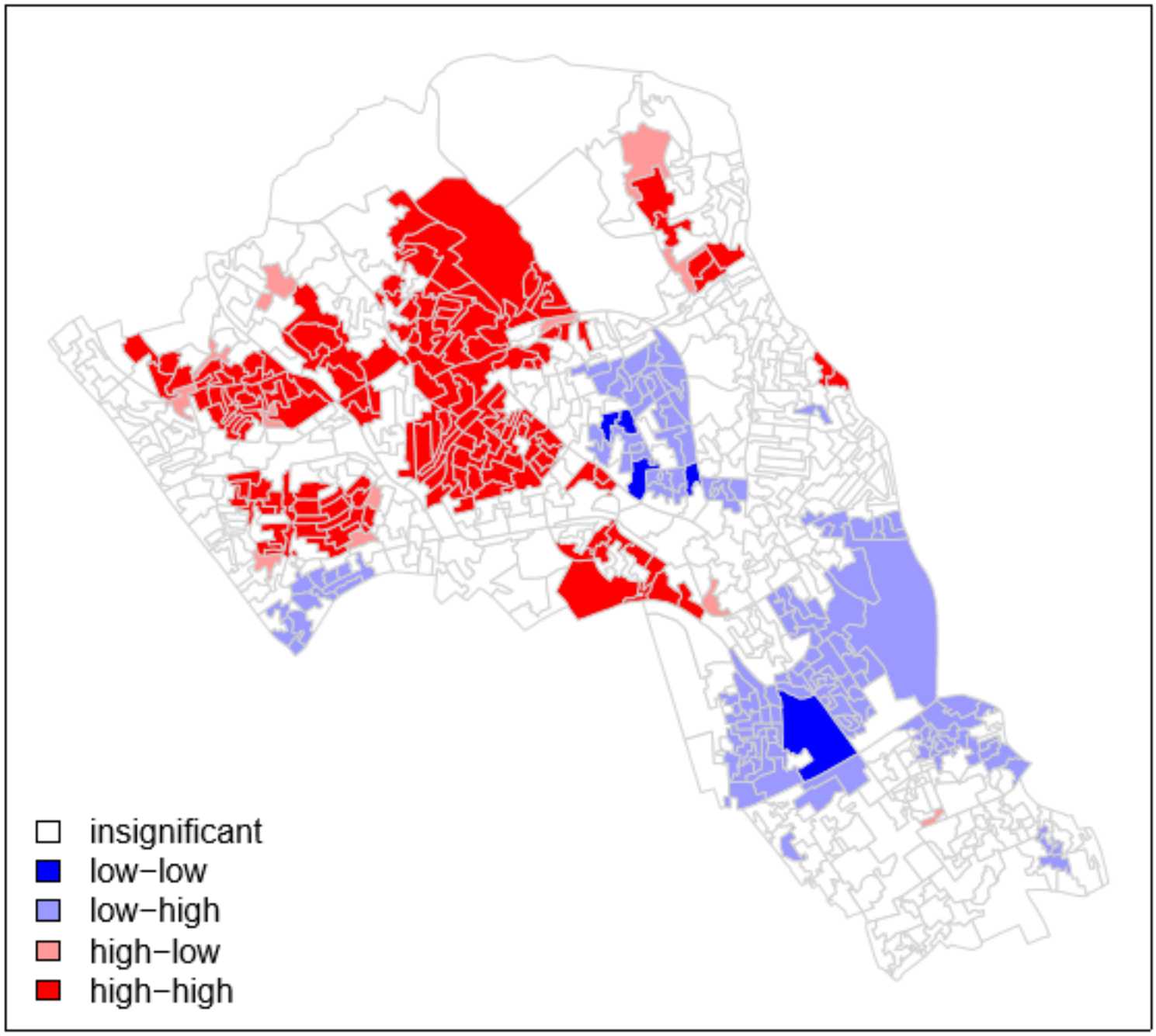


From the map it is possible to observe the variations in autocorrelation across space. We can interpret that there seems to be a geographic pattern to the autocorrelation. However, it is not possible to understand if these are clusters of high or low values.

Why not try to make a map of the P-value to observe variances in significance across Camden? Use `names(moran.map@data)` to find the column headers.

One thing we could try to do is to create a map which labels the features based on the types of relationships they share with their neighbours (i.e. high and high, low and low, insigificant, etc…). The following code will run this for you. *Source: Brunsdon and Comber (2015) (https://uk.sagepub.com/en-gb/eur/an-introduction-to-r-for-spatial-analysis-and-mapping/book241031)*

```
### to create LISA cluster map ###
quadrant <- vector(mode="numeric",length=nrow(local))

# centers the variable of interest around its mean
m.qualification <- OA.Census$Qualification - mean(OA.Census$Qualification)

# centers the local Moran's around the mean
m.local <- local[,1] - mean(local[,1])

# significance threshold
signif <- 0.1

# builds a data quadrant
quadrant[m.qualification >0 & m.local>0] <- 4
quadrant[m.qualification <0 & m.local<0] <- 1
quadrant[m.qualification <0 & m.local>0] <- 2
quadrant[m.qualification >0 & m.local<0] <- 3
quadrant[local[,5]>signif] <- 0

# plot in r
brks <- c(0,1,2,3,4)
colors <- c("white","blue",rgb(0,0,1,alpha=0.4),rgb(1,0,0,alpha=0.4),"red")
plot(OA.Census,border="lightgray",col=colors[findInterval(quadrant,brks,all.inside
=FALSE)])
box()
legend("bottomleft",legend=c("insignificant","low-low","low-high","high-low","high
-high"),
       fill=colors,bty="n")
```



It is apparent that there is a statistically significant geographic pattern to the clustering of our qualification variable in Camden.

Getis-Ord

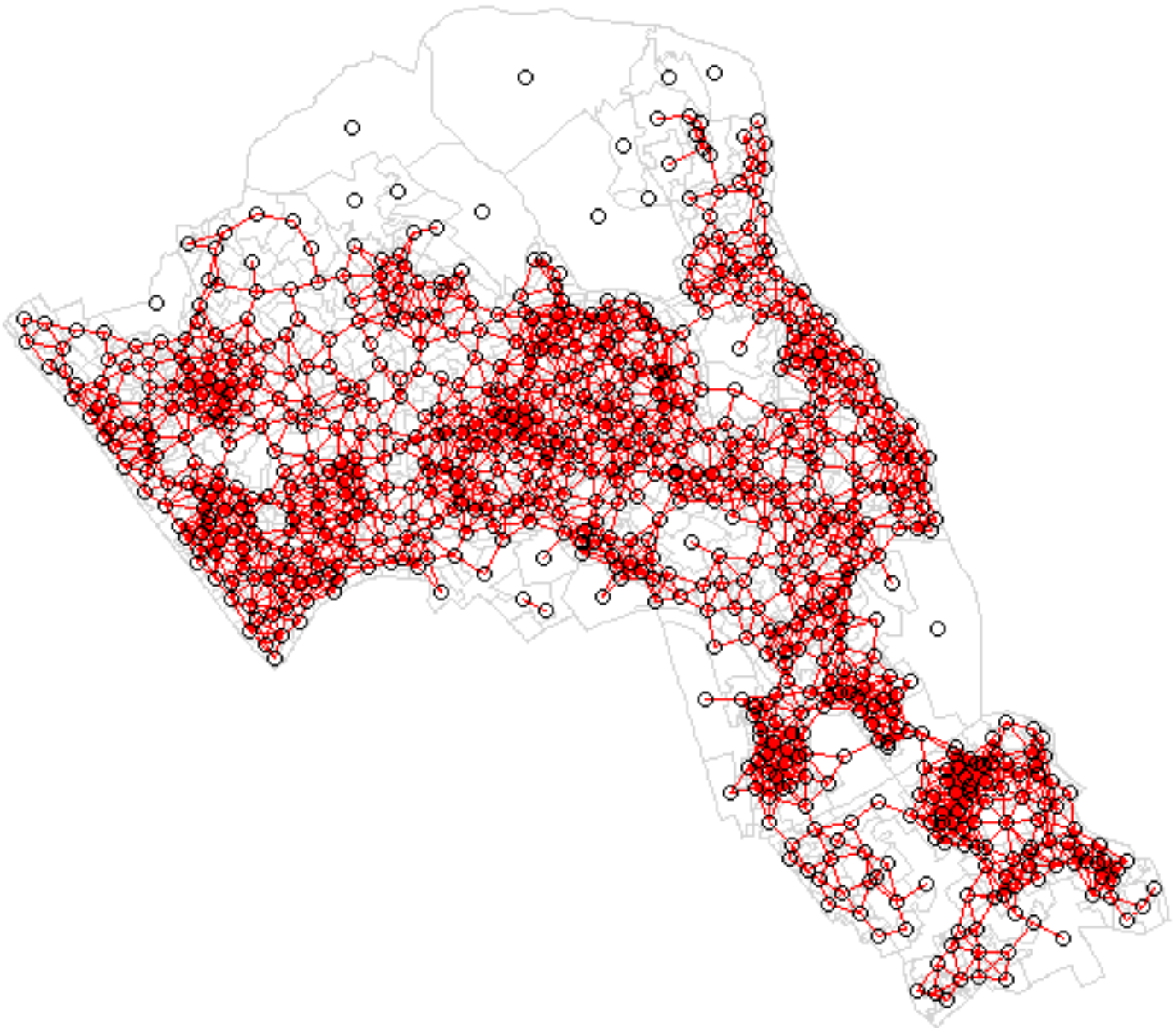
Another approach we can take is hot-spot analysis. The Getis-Ord Gi Statistic looks at neighbours within a defined proximity to identify where either high or low values cluster spatially. Here statistically significant hot-spots are recognised as areas of high values where other areas within a neighbourhood range also share high values too.

First, we need to define a new set of neighbours. Whilst the spatial autocorrection considered units which shared borders, for Getis-Ord we are defining neighbours based on proximity.

```
# creates centroid and joins neighbours within 0 and x units
nb <- dnearneigh(coordinates(OA.Census),0,800)
# creates listw
nb_lw <- nb2listw(nb, style = 'B')
```

```
# plot the data and neighbours
plot(OA.Census, border = 'lightgrey')
plot(nb, coordinates(OA.Census), add=TRUE, col = 'red')
```

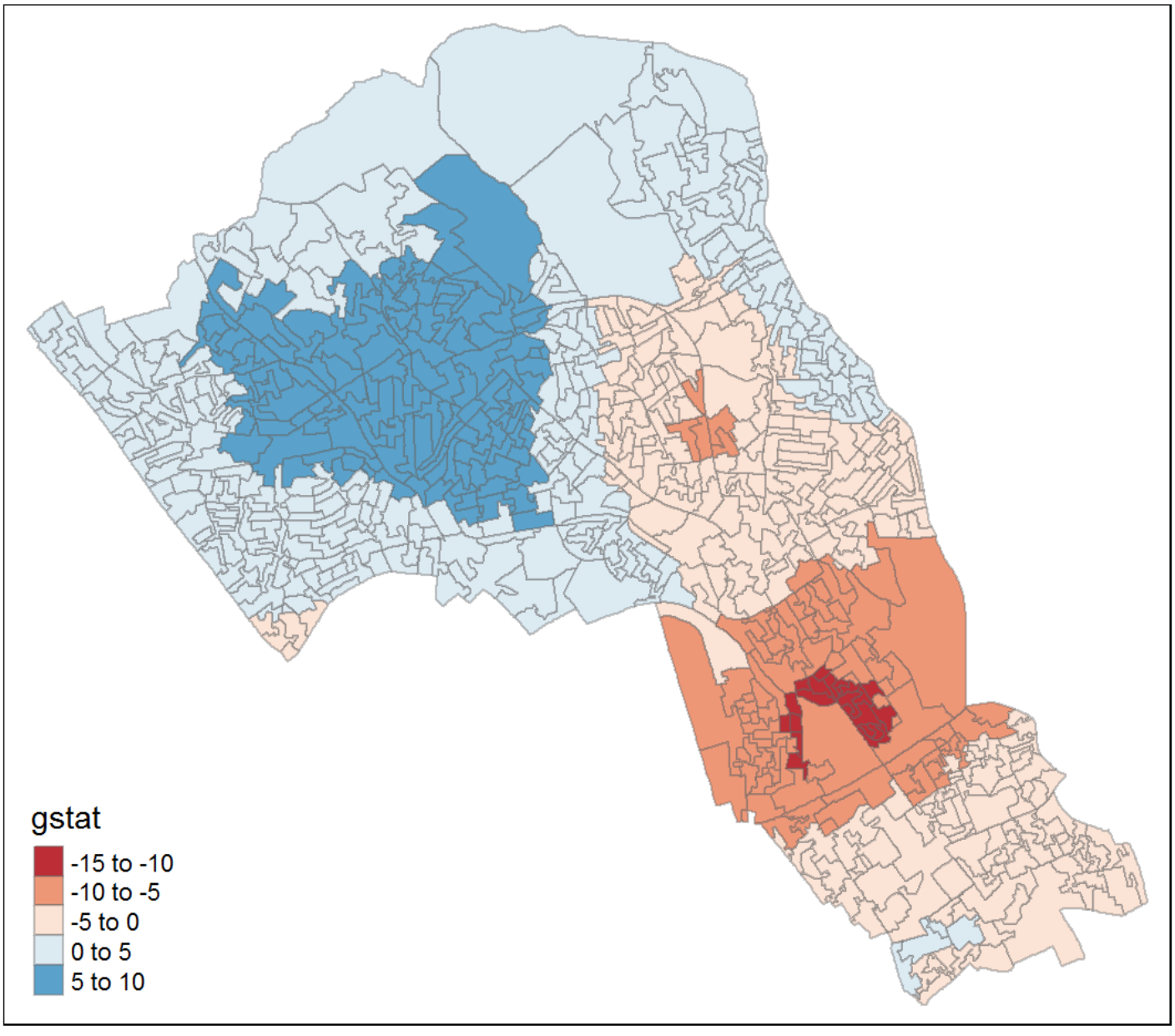
Note in this example map below we only had a search radius of 250 metres to demonstrate the function. However, it means that some areas do not have any defined nearest neighbours so we will need to search 800 meters or more for our model in Camden.



With a set of neighbourhoods estabilished we can now run the test and bind the results to our polygon file.

On some machines the `cbind` may not work with a spatial data file, in this case you will need to change *OA.Census* to *OA.Census@data (mailto:OA.Census@data)* so that R knows which part of the spatial data file to join. If you take this approach the subsequent column ordering may be different to what is shown in the example below.

```
# compute Getis-Ord Gi statistic
local_g <- localG(OA.Census$Qualification, nb_lw)
local_g <- cbind(OA.Census, as.matrix(local_g))
names(local_g)[6] <- "gstat"

# map the results
tm_shape(local_g) + tm_fill("gstat", palette = "RdBu", style = "pretty") + tm_bord
ers(alpha=.4)
```



The Gi Statistic is represented as a Z-score. Greater values represent a greater intensity of clustering and the direction (positive or negative) indicates high or low clusters. The final map should indicate the location of hot-spots across Camden. Repeat this for another variable.

The rest of the online tutorials in this series can be found at: https://data.cdrc.ac.uk/tutorial/an-introduction-to-spatial-data-analysis-and-visualisation-in-r (https://data.cdrc.ac.uk/tutorial/an-introduction-to-spatial-data-analysis-and-visualisation-in-r)