Assignment 2: Spatial Analysis and Viz with R (II)

1: Screenshots of at least 2 critical results from each practical exercise. Please label each one properly.

Practical 7: Using R as a GIS

```
> # point in polygon. Gives the points the attributes of the polygons that they are in
> pip <- over(House.Points, OA.Census)
> pip
       OA11CD White_British Low_Occupancy Unemployed Qualification
1   E00004771      46.77419     0.8547009  1.8264840      50.40650
2   E00004357      44.28571     7.6271186  1.3274336      71.25000
3   E00004345      48.54369     1.6528926  0.9090909      55.10204
4   E00004355      57.08812     5.7377049  1.5789474      55.20362
5   E00004495      72.26562     3.9682540  2.0833333      65.19824
6   E00004226      60.15625     1.6806723  1.5625000      69.71154
7   E00004687      52.24490     6.8965517  3.2967033      59.25926
8   E00004342      36.54822     8.5106383  1.8750000      70.00000
9   E00004357      44.28571     7.6271186  1.3274336      71.25000
10  E00004403      68.08511     0.0000000  1.3953488      50.82645
11  E00004712      49.38272     1.9108280  0.4032258      53.06859
12  E00004403      68.08511     0.0000000  1.3953488      50.82645
13  E00004346      58.19398     2.1276596  1.1111111      37.03704
14  E00004241      58.98305     3.0075188  0.4329004      70.56452
15  E00004687      52.24490     6.8965517  3.2967033      59.25926
16  E00004216      66.22074     3.4246575  1.2605042      70.73171
17  E00004358      42.64706     8.7719298  3.0150754      48.88889
18  E00004499      69.46565     4.4444444  1.7793594      66.55518
19  E00004712      49.38272     1.9108280  0.4032258      53.06859
20  E00004379      62.92135     3.7037037  2.9268293      65.81197
21  E00004346      58.19398     2.1276596  1.1111111      37.03704
```

Figure 1. The House.Points Data assigned with attributes of OA.Census Data using Point in Polygon process with over() function.

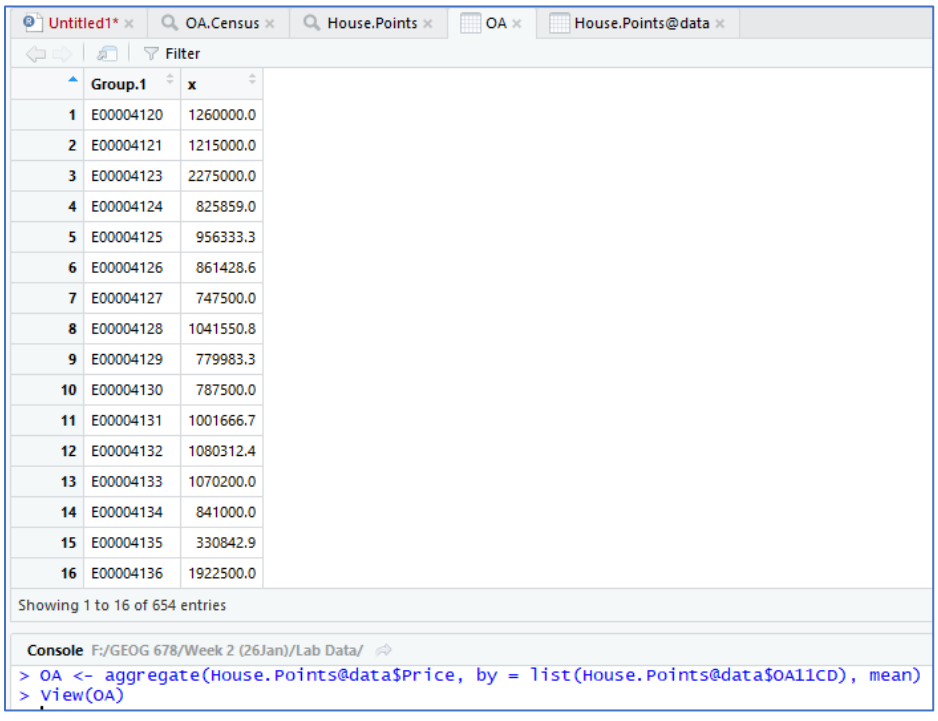

Figure 2. Mean house prices calculated using aggregate() function with parameter ‘mean’.

Assignment 2: Spatial Analysis and Viz with R (II)

Practical 8: Representing Densities in R

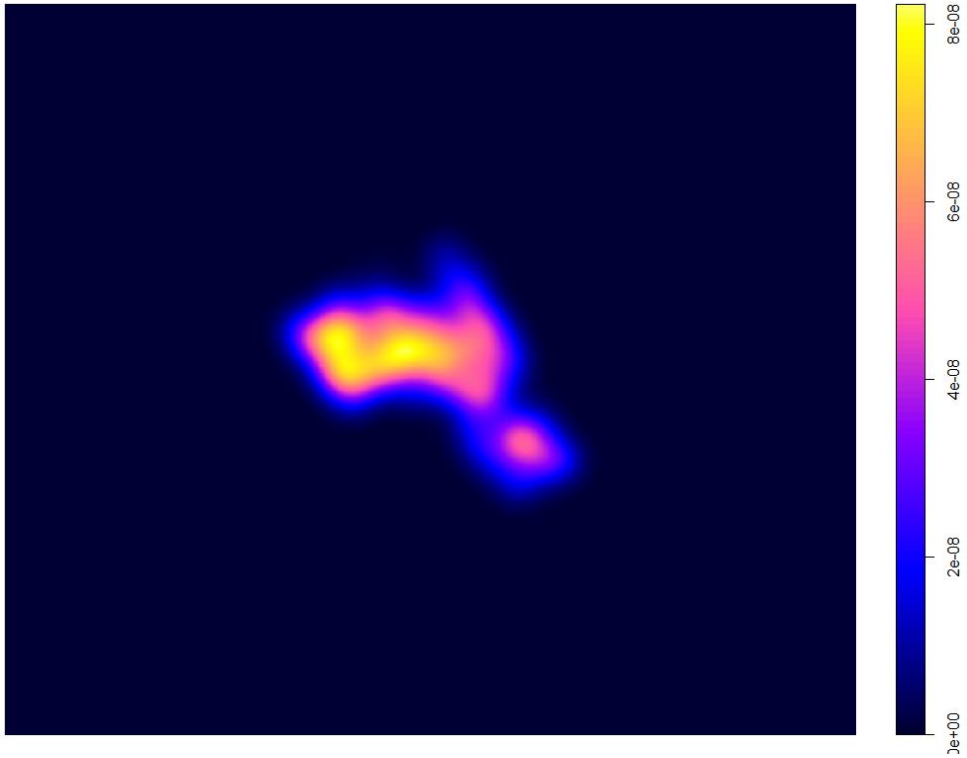

Figure 3. Map showing Kernel Density for House Sale Data in Camden.

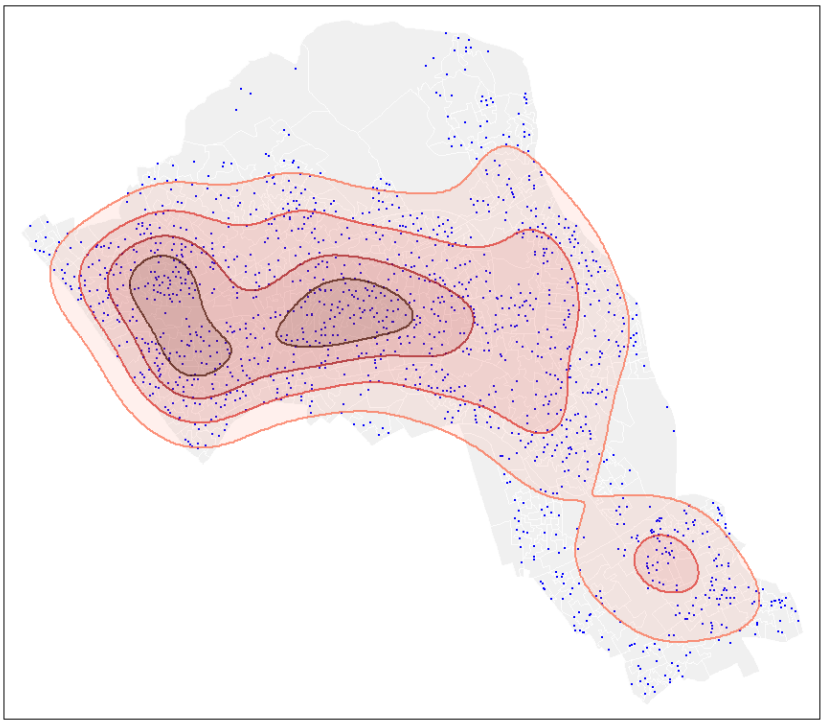

Figure 4. Map with several layers showing location of houses, kernel density with different ranges.

Assignment 2: Spatial Analysis and Viz with R (II)

Practical 9: Measuring Spatial Autocorrelation in R

```
> # 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
```

Figure 5. Number of neighbors calculated for OA.Census polygon using poly2nb() from spdep package.

```
> # 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
```

Figure 6. Global Moran's I Test Results

Practical 10: Geographically Weighted Regression in R

```
> model <- lm(OA.Census$Qualification ~ OA.Census$Unemployed+OA.Census$White_British)
> summary(model)

Call:
lm(formula = OA.Census$Qualification ~ OA.Census$Unemployed +
    OA.Census$White_British)

Residuals:
    Min      1Q  Median      3Q     Max
-50.311  -8.014   1.006   8.958  38.046

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)             47.86697    2.33574   20.49   <2e-16
OA.Census$Unemployed    -3.29459    0.19027  -17.32   <2e-16
OA.Census$White_British  0.41092    0.04032   10.19   <2e-16

(Intercept)             ***
OA.Census$Unemployed    ***
OA.Census$White_British ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.69 on 746 degrees of freedom
Multiple R-squared:  0.4645,    Adjusted R-squared:  0.463
F-statistic: 323.5 on 2 and 746 DF,  p-value: < 2.2e-16

> # this will plot 4 scatter plots from the linear model
> plot(model)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
```

Figure 7. Linear Regression Analysis Using lm() function in R.

```
> #GWR Model
> library("spgwr")
Loading required package: spData
To access larger datasets in this package, install the spDataLarge package with:
`install.packages('spDataLarge', repos='https://nowosad.github.io/drat/', type='source')`
NOTE: This package does not constitute approval of GWR
as a method of spatial analysis; see example(gwr)
Warning message:
package 'spgwr' was built under R version 4.0.3
> #calculate kernel bandwidth
> GWRbandwidth <- gwr.sel(OA.Census$Qualification ~ OA.Census$Unemployed +
+                         OA.Census$White_British, data=OA.Census, adapt =TRUE)
Adaptive q: 0.381966 CV score: 101420.8
Adaptive q: 0.618034 CV score: 109723.2
Adaptive q: 0.236068 CV score: 96876.06
Adaptive q: 0.145898 CV score: 94192.41
Adaptive q: 0.09016994 CV score: 91099.75
Adaptive q: 0.05572809 CV score: 88242.89
Adaptive q: 0.03444185 CV score: 85633.41
Adaptive q: 0.02128624 CV score: 83790.04
Adaptive q: 0.01315562 CV score: 83096.03
Adaptive q: 0.008130619 CV score: 84177.45
Adaptive q: 0.01535288 CV score: 83014.34
Adaptive q: 0.01515437 CV score: 82957.49
Adaptive q: 0.01436908 CV score: 82857.74
Adaptive q: 0.01440977 CV score: 82852.4
Adaptive q: 0.01457859 CV score: 82833.25
Adaptive q: 0.01479852 CV score: 82855.45
Adaptive q: 0.01461928 CV score: 82829.32
Adaptive q: 0.01468774 CV score: 82823.82
Adaptive q: 0.01473006 CV score: 82835.89
Adaptive q: 0.01468774 CV score: 82823.82
```

Figure 8. Kernel Bandwidth Calculated for given Geographically Weighted Regression.

```
> #print the results of the model
> gwr.model
Call:
gwr(formula = OA.Census$Qualification ~ OA.Census$Unemployed +
    OA.Census$White_British, data = OA.Census, adapt = GWRbandwidth,
    hatmatrix = TRUE, se.fit = TRUE)
Kernel function: gwr.Gauss
Adaptive quantile: 0.01468774 (about 11 of 749 data points)
Summary of GWR coefficient estimates at data points:
                            Min.  1st Qu.   Median  3rd Qu.     Max.  Global
X.Intercept.            11.08183 34.43427 45.76862 59.75372 85.01866 47.8670
OA.Census.Unemployed    -5.45291 -3.28308 -2.55398 -1.79413  0.77019 -3.2946
OA.Census.White_British -0.28046  0.19955  0.37788  0.53216  0.94678  0.4109
Number of data points: 749
Effective number of parameters (residual: 2traceS - traceS'S): 132.6449
Effective degrees of freedom (residual: 2traceS - traceS'S): 616.3551
Sigma (residual: 2traceS - traceS'S): 9.903539
Effective number of parameters (model: traceS): 94.44661
Effective degrees of freedom (model: traceS): 654.5534
Sigma (model: traceS): 9.610221
Sigma (ML): 8.983902
AICc (GWR p. 61, eq 2.33; p. 96, eq. 4.21): 5633.438
AIC (GWR p. 96, eq. 4.22): 5508.777
Residual sum of squares: 60452.16
Quasi-global R2: 0.7303206
>
```

Figure 9. Geographically Weighted Regression Analysis using gwr() in R.

Assignment 2: Spatial Analysis and Viz with R (II)

Practical 11: Interpolating Point Data in R

ERROR

```
> int.Z <- over(dat.pp,House.Points, fn=mean)

Error in get(as.character(FUN), mode = "function", envir = envir) :
  object 'fn' of mode 'function' was not found
```

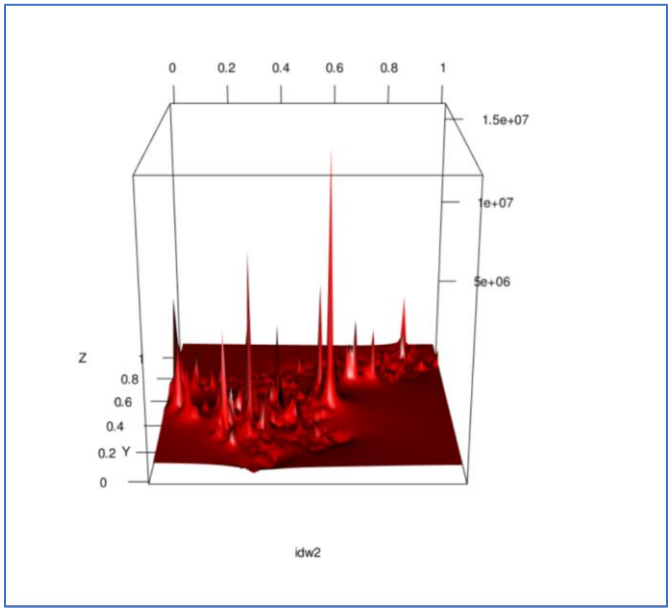

Figure 10. 3-Dimensional Plot of IDW for the Price variable of House.Points data.

```
> library(automap)
Warning message:
package 'automap' was built under R version 4.0.3
> # this is the same grid as before
> grid <-spsample(House.Points, type = 'regular', n = 10000)
> # runs the kriging
> kriging_result = autoKrige(log(Price)~1, House.Points, grid)
Warning in autoKrige(log(Price) ~ 1, House.Points, grid) :
  Removed 2556 duplicate observation(s) in input_data:
          coordinates    UID    Price oseast1m osnrth1m
2     (525813, 185524) 594622 15200000   525813   185524
7     (528756, 182768) 592670  8500000   528756   182768
28    (527973, 187231) 515670  5700000   527973   187231
10    (526633, 187131) 594637  7720000   526633   187131
41    (527094, 183871) 594158  4450000   527094   183871
24    (528707, 183342) 592648  5775000   528707   183342
46    (526549, 184541) 594455  4250000   526549   184541
64    (527838, 184072) 592990  3700000   527838   184072
24.1  (528707, 183342) 592648  5775000   528707   183342
48    (528707, 183342) 592650  4050000   528707   183342
```

Figure 11. Sample points selected using regular sampling method and automatic interpolation performed using kriging technique.

Assignment 2: Spatial Analysis and Viz with R (II)

Practical 12: Functions and Loops in R

```
> myfunction <- function(x){
+   z <- log(x)
+   y <- round(z,4)
+   return(y)
+ }
> # resets the newdata object so we can run it again
> newdata <- Census.Data
> # runs the function
> newdata$Qualification_2 <- myfunction(Census.Data$Qualification)
> newdata$Qualification_2
  [1] 4.2990 4.2471 4.2133 4.1072 4.1894 4.3068 4.1343 4.1002 4.2496 4.1997 4.1997 4.1663 4.2972 4.1803 4.2890 4.3151 4.2998
 [18] 4.2351 4.0633 3.1340 4.2764 4.1730 4.2553 4.2556 4.3319 4.1886 4.2523 4.2824 4.1931 4.1789 4.3820 4.3736 4.2522 4.1829
 [35] 4.2322 4.1844 4.2881 4.2600 3.5924 3.2402 3.4762 4.2824 4.1926 4.0580 3.3800 3.2602 3.2737 4.1780 4.3332 4.0194 4.0695
 [52] 3.5615 3.2824 3.4313 3.5401 3.4815 4.0408 3.4573 3.8986 3.4695 3.5119 3.7950 3.9435 4.0441 3.8554 4.1617 3.9304 4.2061
 [69] 4.1372 4.2222 4.1376 3.8439 4.1057 4.0466 4.2894 3.3195 4.1828 4.2316 4.1364 3.4578 3.2558 3.2635 3.6497 3.0383 3.8042
 [86] 4.0597 3.6096 3.6960 4.1931 4.2589 4.1270 3.7579 4.0186 3.5347 4.3365 4.0646 3.6889 4.2737 3.8346 4.2444 4.2018 4.3422
[103] 4.2187 3.8539 4.2848 4.0484 4.1008 4.3653 3.9264 3.8852 3.7696 4.1262 3.8438 3.8020 4.2565 4.3130 4.2063 3.2798 3.7639
[120] 3.4358 4.1197 3.8985 4.1947 4.1411 4.1971 4.0335 3.5804 3.3797 3.8498 3.7406 4.0444 4.0855 3.3011 3.3077 3.6124 3.1978
[137] 3.4225 4.0727 3.8716 3.6852 3.6850 3.7693 3.6794 4.0848 4.1477 4.1921 4.1010 4.0484 3.8503 4.0263 4.1501 3.7355 4.0855
[154] 4.1674 4.0316 3.9416 3.8887 4.1750 3.9267 4.2174 3.9157 4.1740 3.9157 3.4319 3.2189 3.9342 4.1164 4.0924 3.6102 4.0667
[171] 4.0701 4.1851 3.7679 3.8331 3.8425 4.2401 4.1444 4.1711 4.1814 4.1599 4.2547 4.2565 4.2117 4.2137 4.1307 4.2555 4.0246
[188] 4.2434 4.0085 4.1878 4.1858 4.1915 4.2576 4.0589 4.1714 3.3496 4.1997 4.1176 4.0205 4.2703 4.1062 4.0794 4.3391 4.1953
[205] 4.1418 4.2804 4.1549 4.2559 4.2499 4.0755 4.3746 4.1122 4.0878 4.2540 4.2485 4.2305 4.2240 4.0092 3.6119 4.0751 4.1033
[222] 4.2193 4.1740 4.0837 4.2867 3.9822 4.1139 4.0110 4.2661 4.2662 3.8896 4.2141 4.2240 3.8339 3.2980 3.0532 3.4795 4.2811
[239] 3.2060 3.5732 4.2754 3.2950 3.1534 2.9840 3.6997 3.1611 3.3406 4.2390 3.6253 2.8944 4.0166 4.1868 3.3792 4.1042 4.2104
[256] 4.1664 4.0591 4.2026 4.1771 4.2081 4.2893 4.3279 4.0680 4.2479 4.0910 3.7563 3.8664 4.1197 3.5369 3.3200 3.7777 3.1437
[273] 3.3417 4.3016 4.3419 3.9284 4.2573 4.1932 4.2707 4.0174 4.2192 3.9491 4.1440 4.3409 4.3160 4.2368 4.2230 4.2909 4.2915
[290] 4.2644 4.2867 4.1518 4.3391 4.2558 4.1902 4.2457 4.2407 4.3245 4.2502 4.2884 4.3246 4.2550 3.6119 4.2657 4.2927 4.3024
[307] 3.8283 4.2217 4.2498 4.2094 4.3833 4.2923 4.2260 3.3894 3.3768 3.8574 3.6564 3.7579 3.6108 3.8305 3.8298 3.9568 3.3508
[324] 3.3906 3.4125 3.1505 4.2581 3.8145 3.3234 3.3927 3.3142 4.0977 2.9244 2.7951 3.6598 4.2984 3.3484 3.1617 4.2590 3.8211
[341] 4.3531 3.7114 4.1997 4.0720 3.4321 2.9629 4.1532 4.0158 4.0030 3.9887 3.4983 4.1936 4.2842 4.1675 3.9208 4.0972 4.0551
[358] 3.0419 4.0077 3.7130 3.4742 4.1543 3.9737 4.2485 4.1774 4.0898 3.5261 3.7069 4.1980 4.1647 4.1136 3.5669 3.8245 4.1963
[375] 3.7253 3.7365 4.2023 4.0551 3.9457 4.1841 4.2031 3.9687 4.2668 4.2102 4.0929 4.2326 3.6733 3.6838 4.2348 3.6227 4.2079
[392] 4.0338 3.7536 4.2323 3.7474 3.1400 3.5782 3.9812 3.9194 3.9304 3.7942 4.2151 3.2805 4.1184 4.3088 3.9731 3.4565 3.7773
```

Figure 12. User-defined function named *myfunction* created to get the log of number and display results with numbers rounded to 4 decimal points.

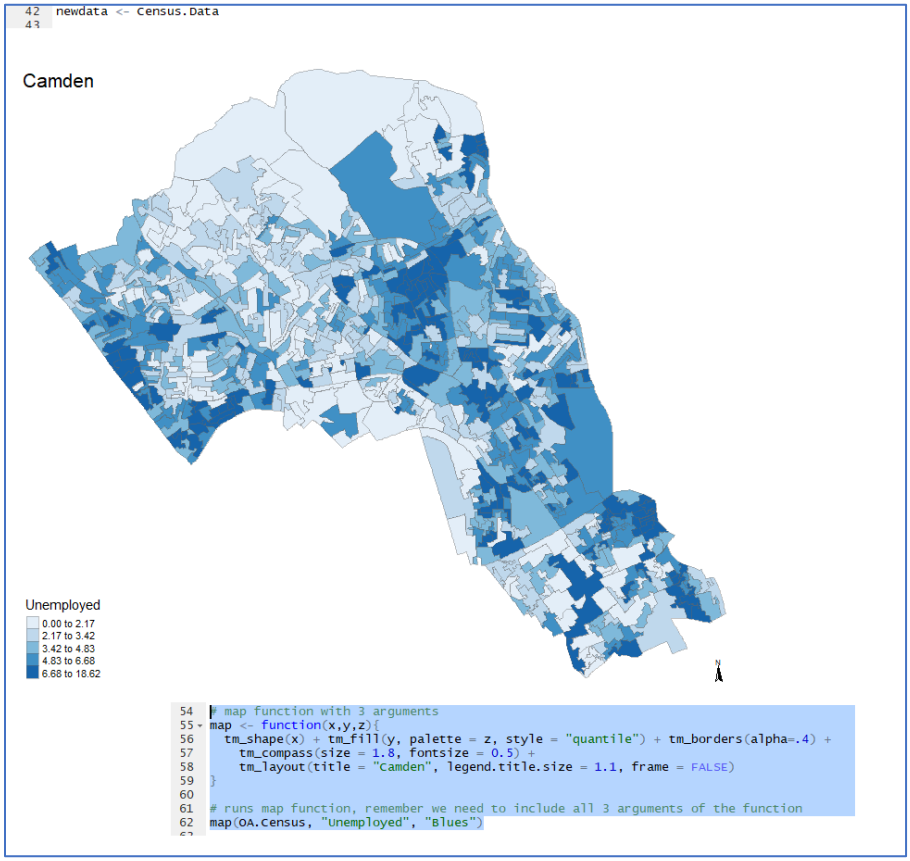

Figure 13. User-defined function used to create map.

2: Screenshots of all plots, graphics, and maps. Please label each one properly.

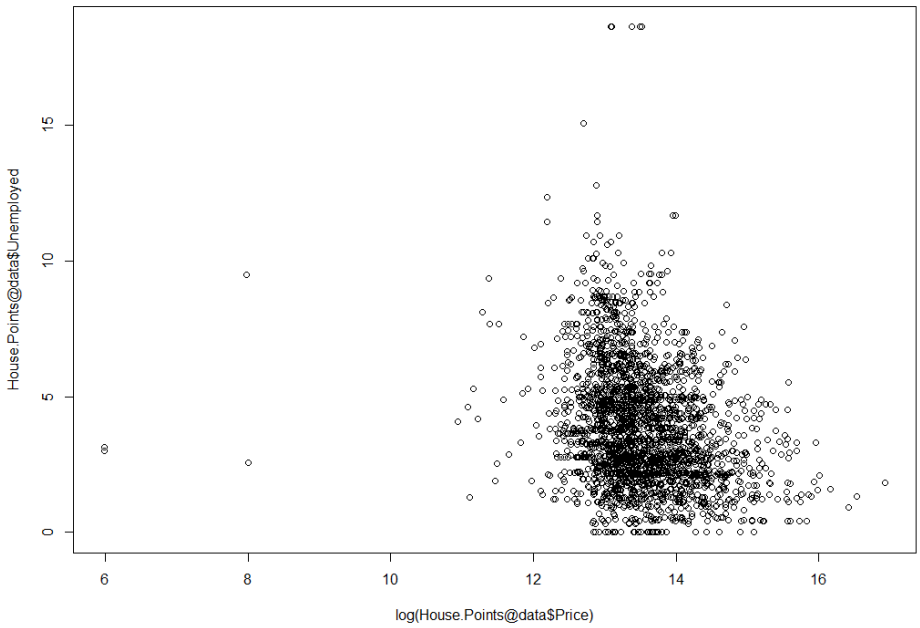

Figure 14. Scatter plot the house prices and local unemployment rates

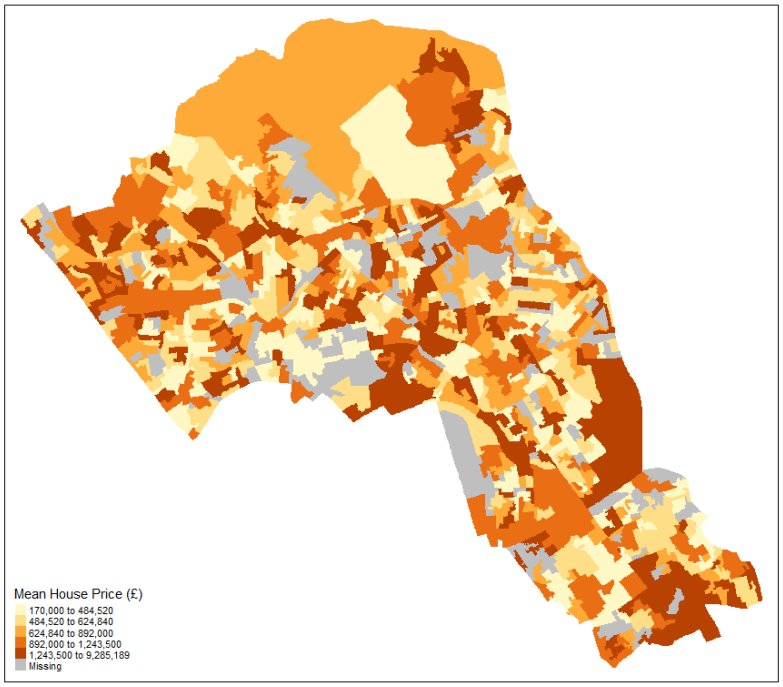

Figure 15. Choropleth map of mean house prices (£) created using tm_shape() and tm_fill() functions.

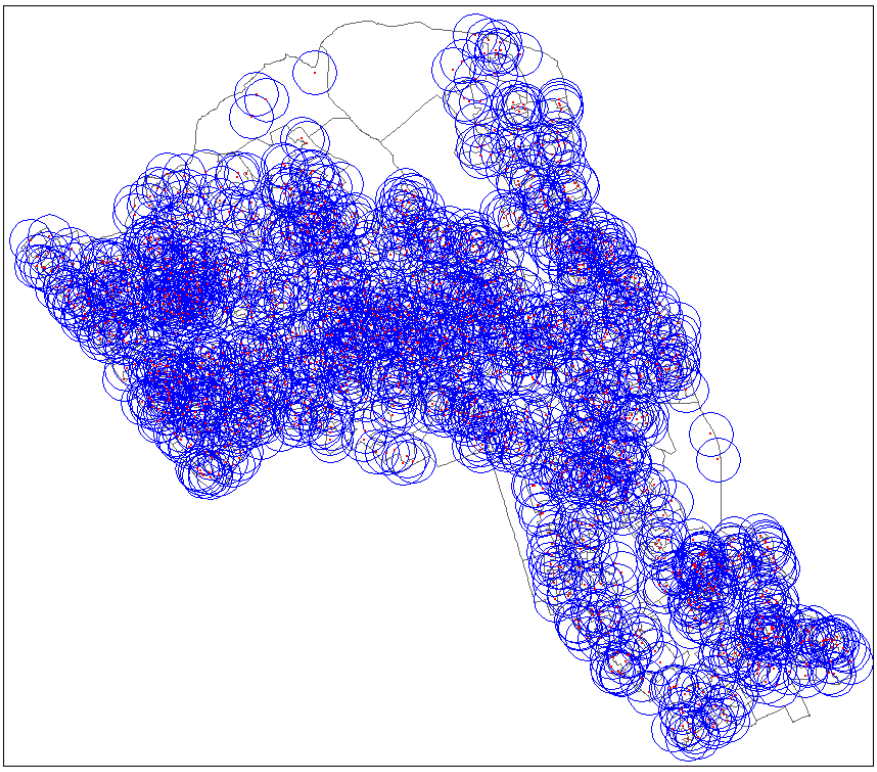

Figure 16. Map showing a 200m buffers for each house point.

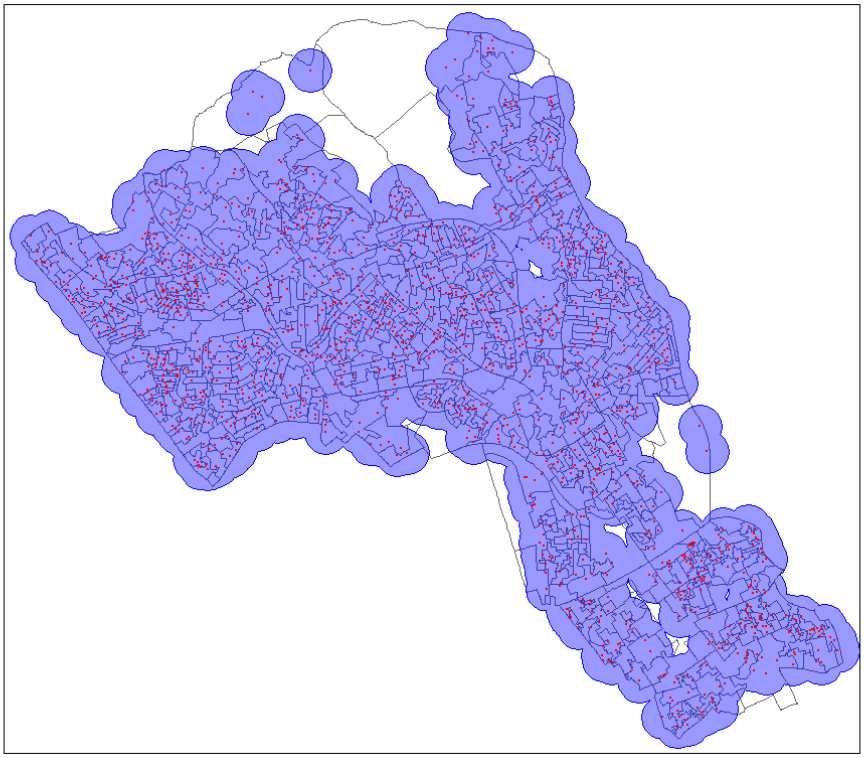

Figure 17. Map showing merged buffers along with house points.

```
> library(raster)
> library(dismo)
Error in library(dismo) : there is no package called 'dismo'
>
>
> library(dismo)
Warning message:
package 'dismo' was built under R version 4.0.3
> google.map <- gmap("Camden, London", type = "satellite")
Error in gmap("Camden, London", type = "satellite") :
  you need to supply a Google API key
> google.map <- gmap("Camden, London", type = "roadmap", filename = "Camden.gmap")
Error in gmap("Camden, London", type = "roadmap", filename = "Camden.gmap") :
  you need to supply a Google API key
>
```

Figure 18. Error in getting backing maps from Google.

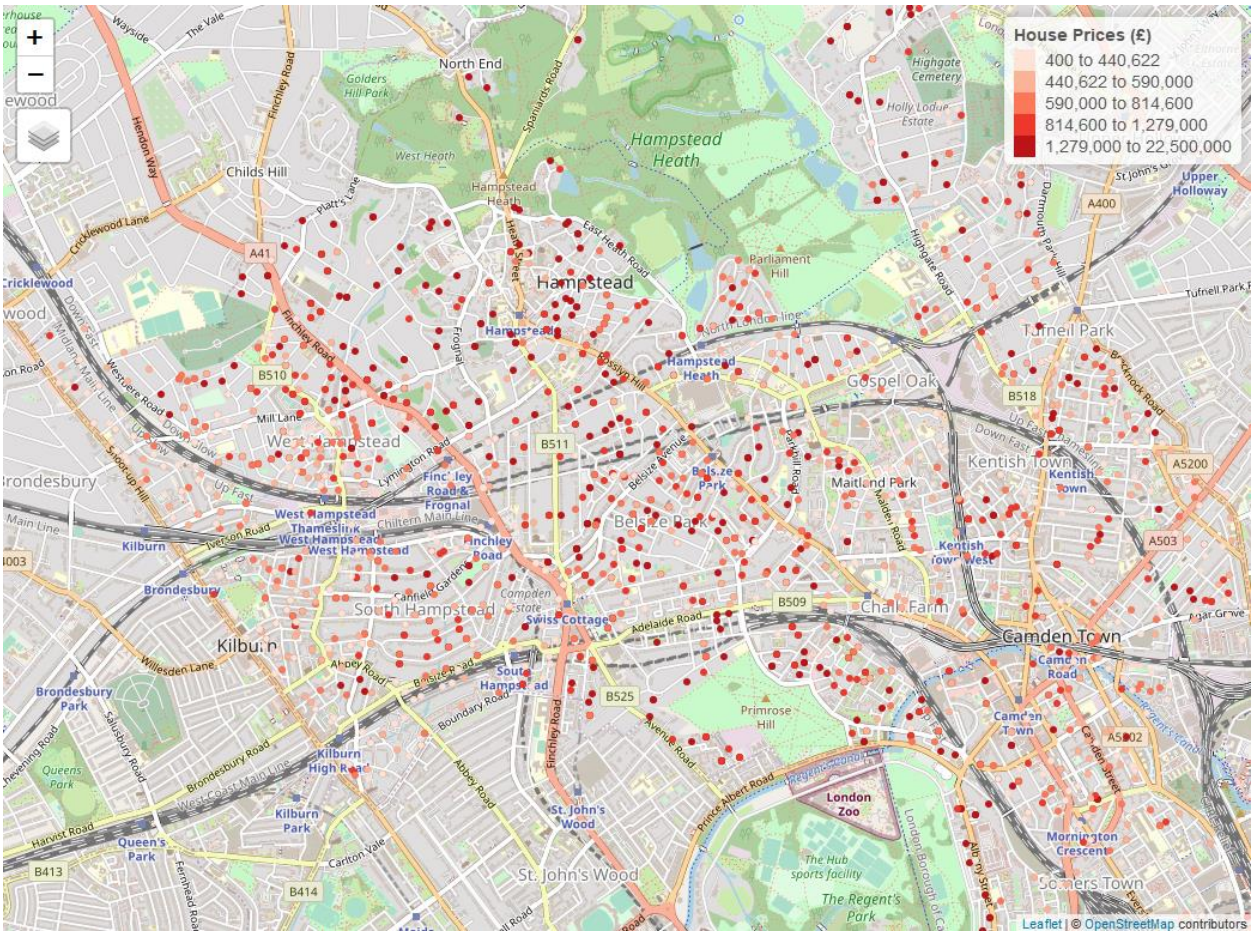

Figure 19. Screenshot of Interactive Map with OpenStreet Map as base map.

Assignment 2: Spatial Analysis and Viz with R (II)

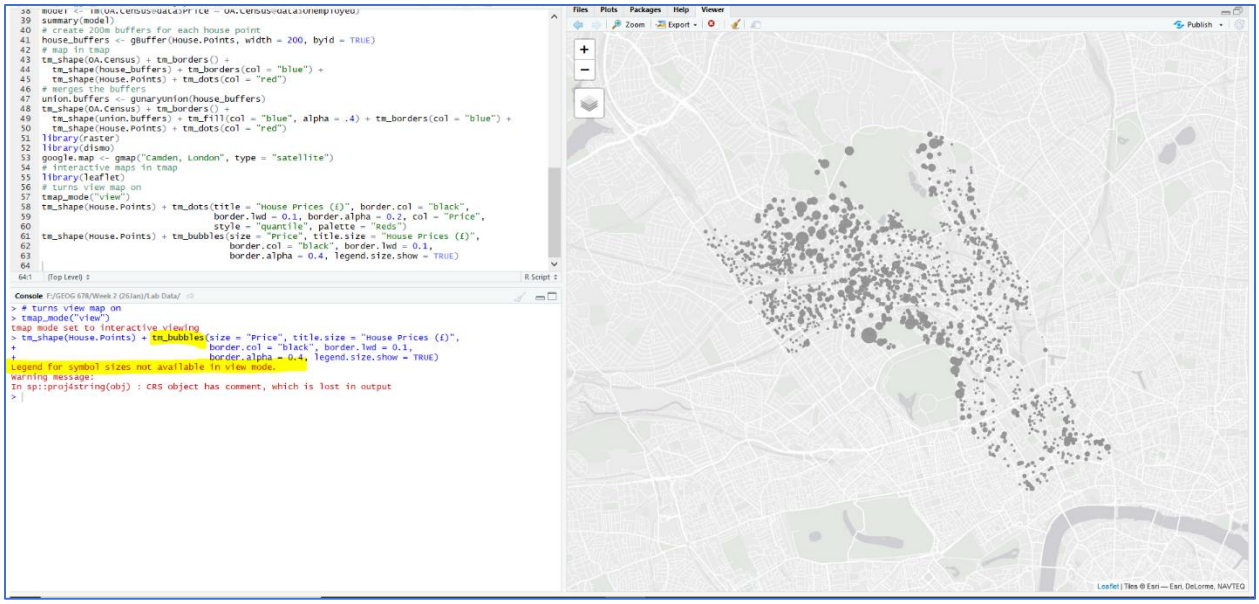

Figure 20. Screenshot of Interactive Map created using tm_shape() and tm_bubbles() functions.

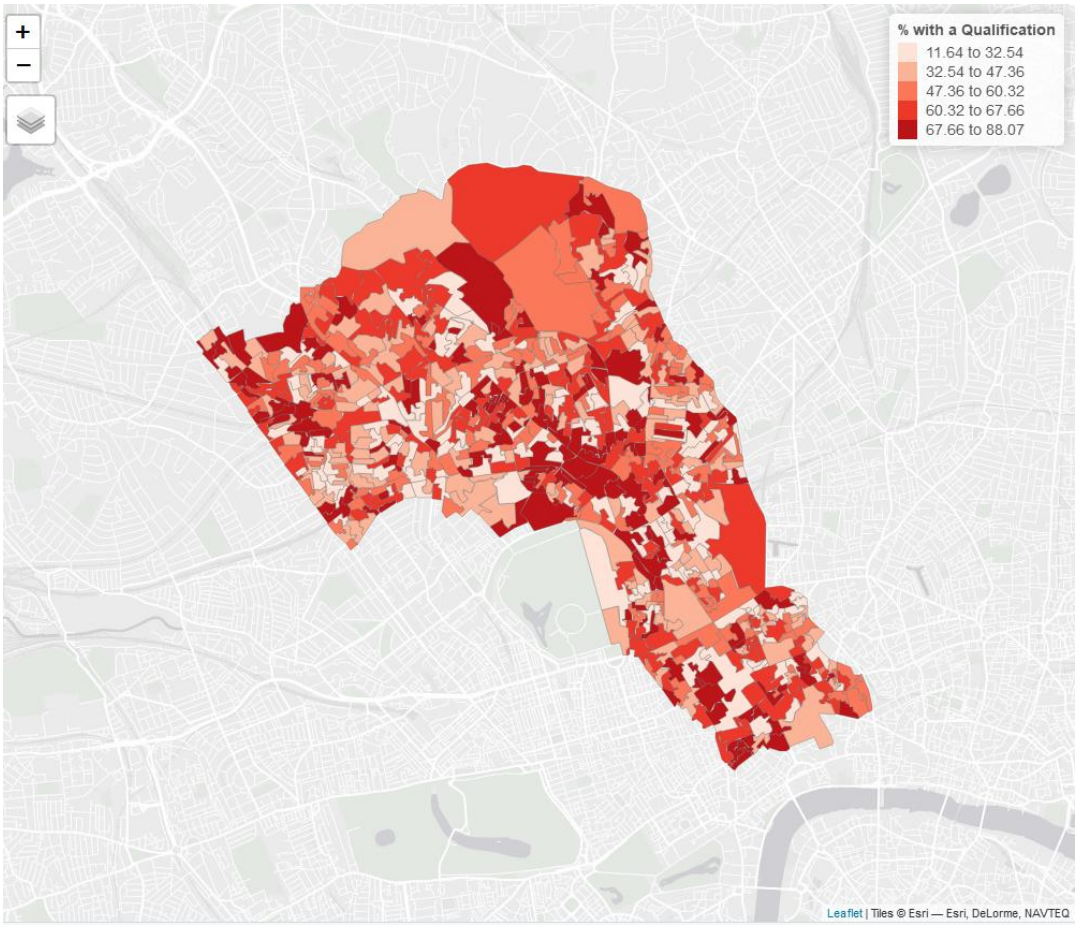

Figure 21. Interactive choropleth map of qualification variable with quantile classification.

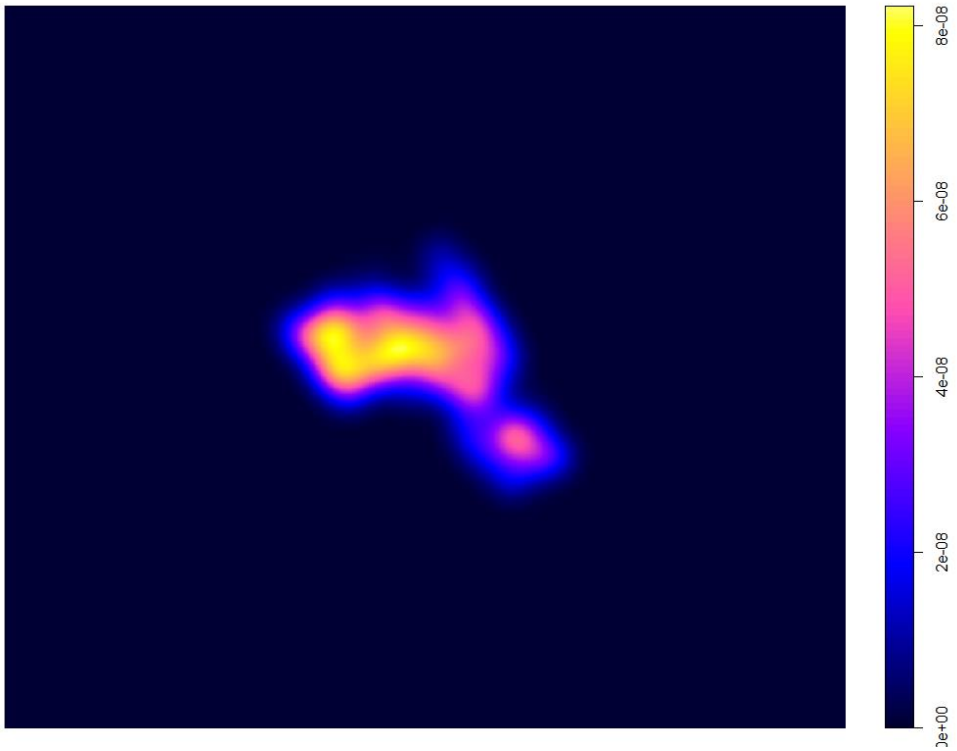

Figure 22. Kernel density estimation for Land Registry house price data using KernelUD() function of adehabitatHR package.

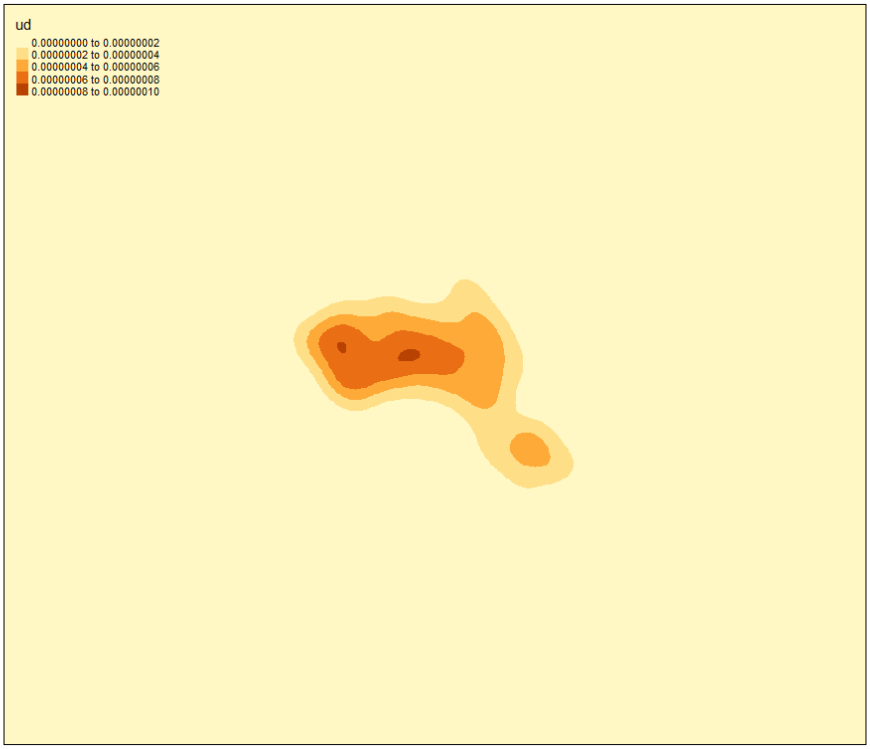

Figure 23. Raster Map of Kernel density estimation for Land Registry house price data.

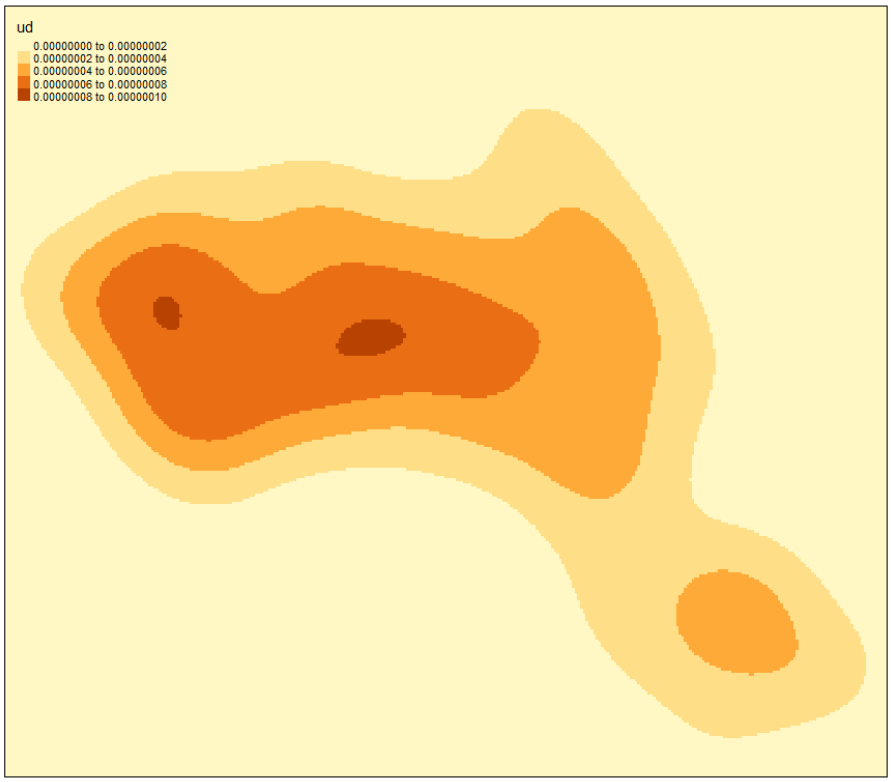

Figure 24. Raster Map of Kernel density estimation for Land Registry house price data for extents of the Camden shapfile.

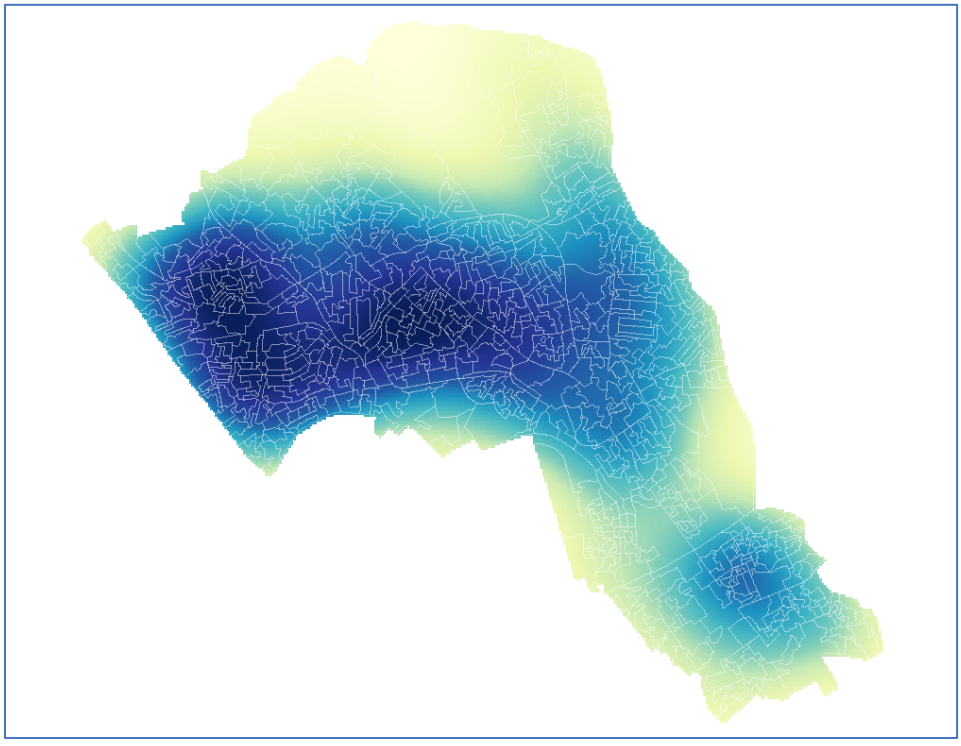

Figure 25. Masked raster map of Kernel density estimation for Land Registry house price data.

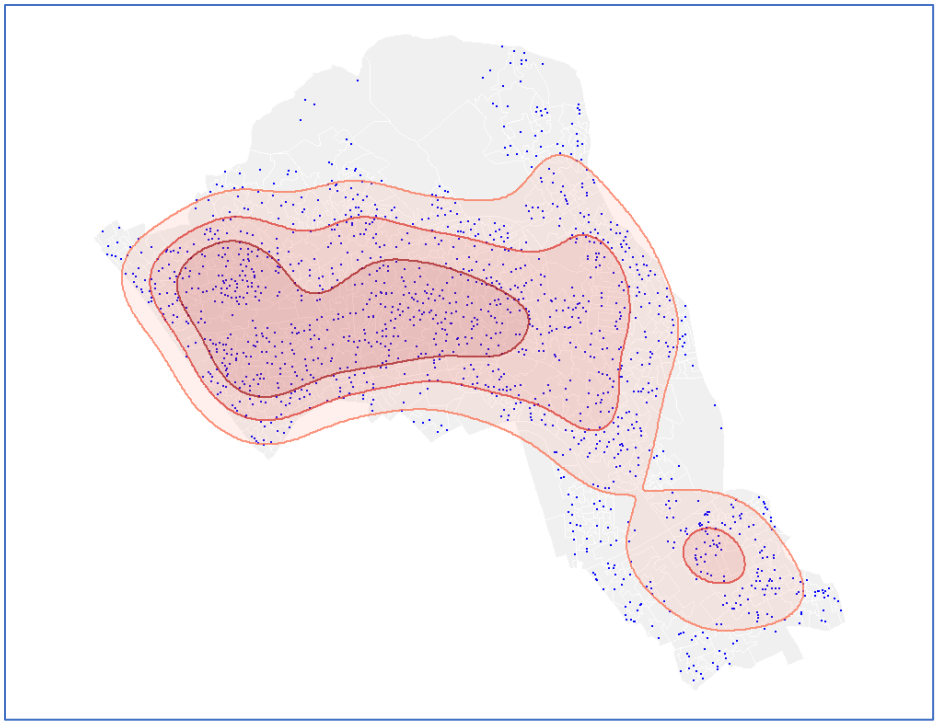

Figure 26. Catchment boundaries for 75%, 50% and 25% range from the kernel density estimations.

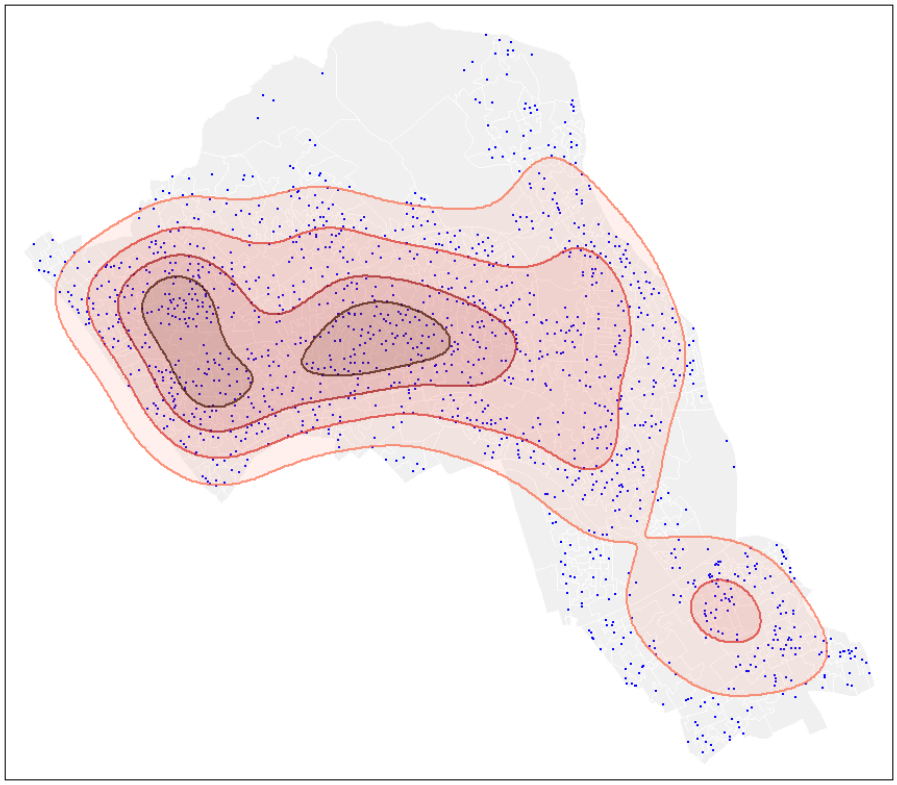

Figure 27. Catchment boundaries for 75%, 50%, 25% and 10% range from the kernel density estimations.

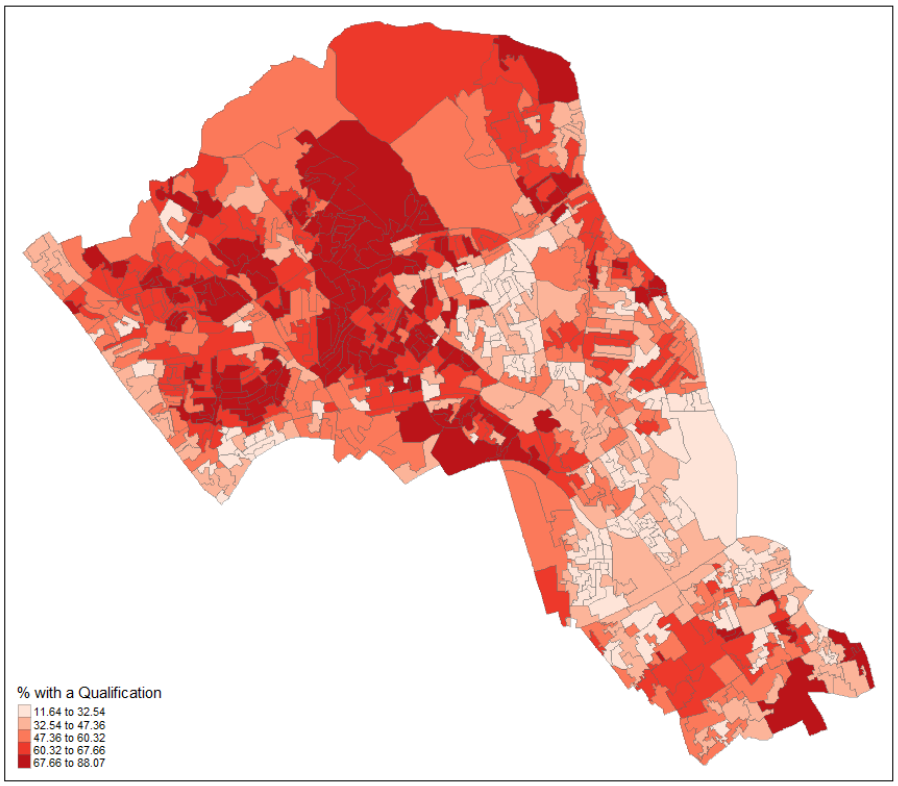

Figure 28. Choropleth map showing % of qualification distribution.

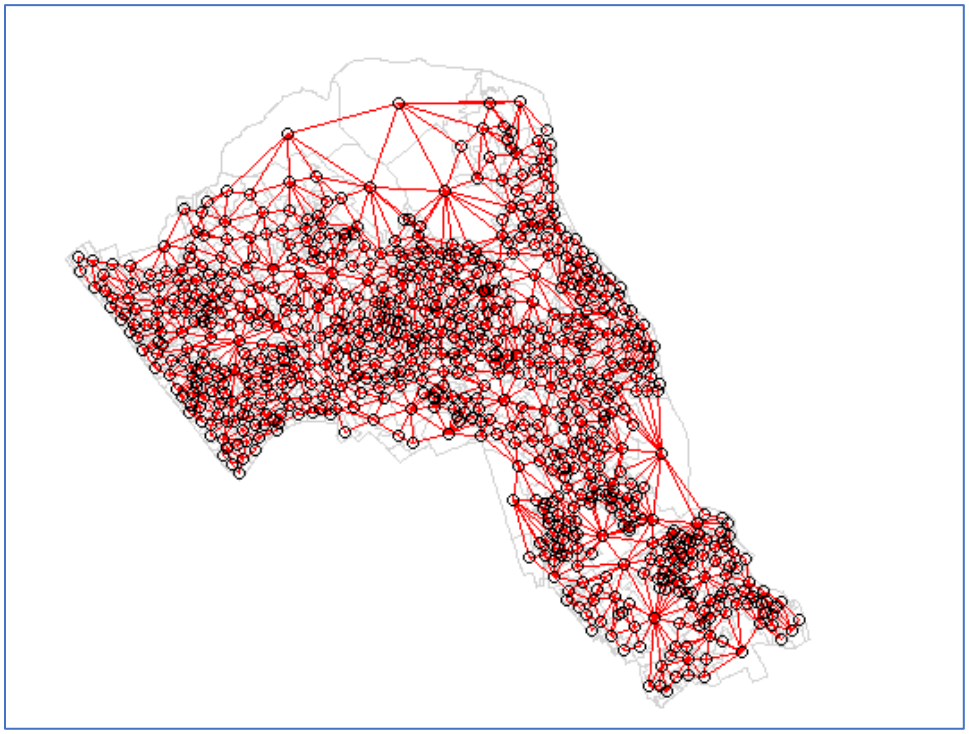

Figure 29. Map showing links between neighbors for OA.Census polygons.

Assignment 2: Spatial Analysis and Viz with R (II)

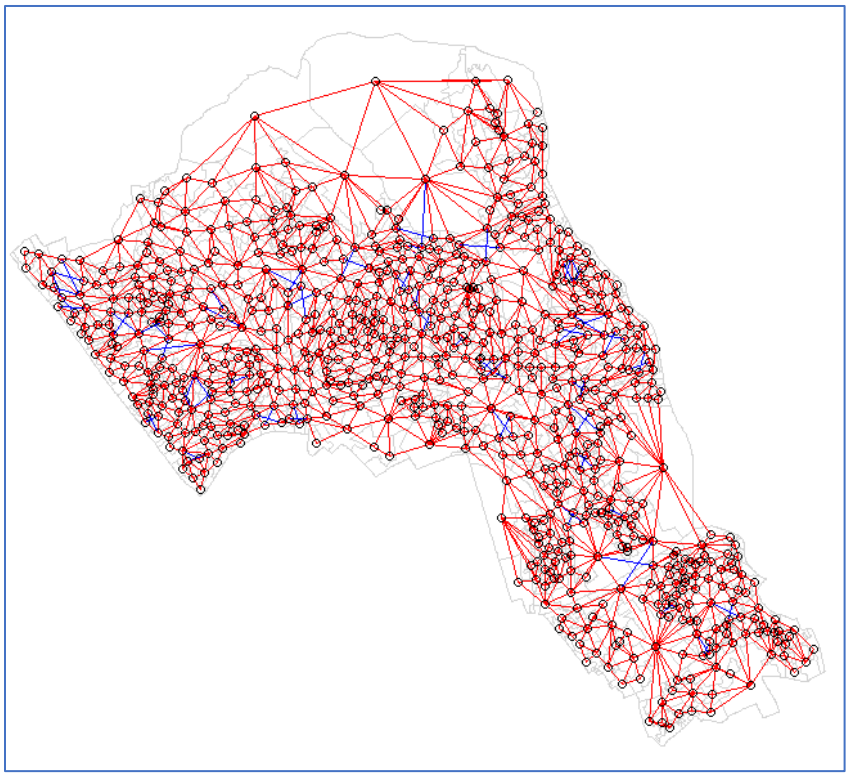

Figure 30. Map showing two different neighbors links for OA.Census polygons.

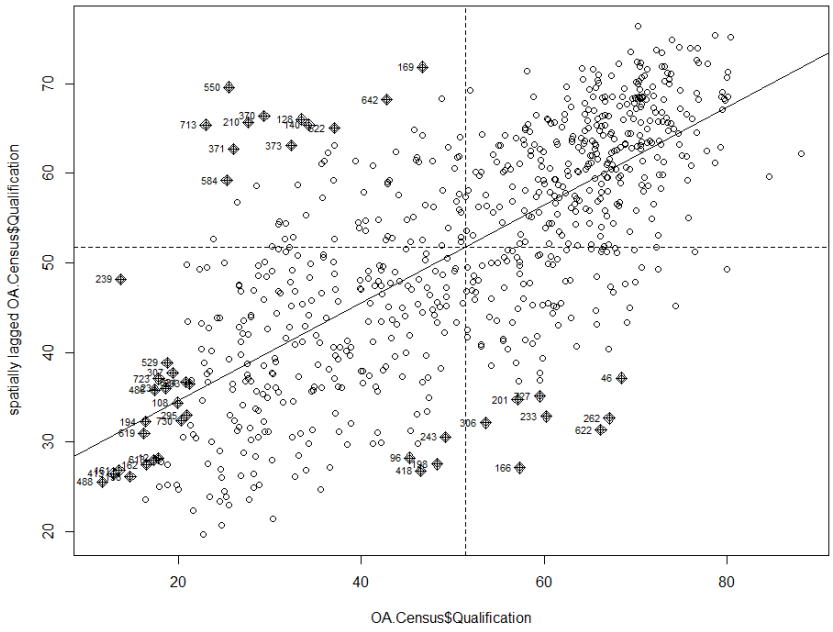

Figure 31. Moran's I plot showing relationship between variable qualification and its spatially lagged values.

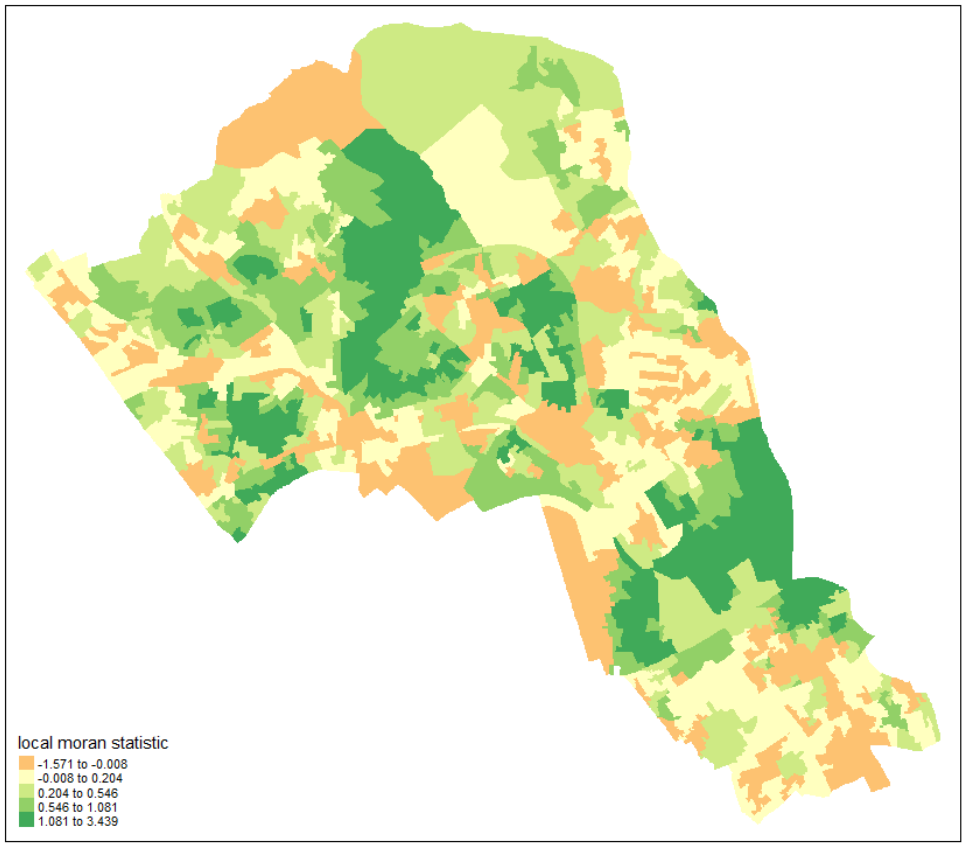

Figure 32. Map showing Local Moran's Statistic

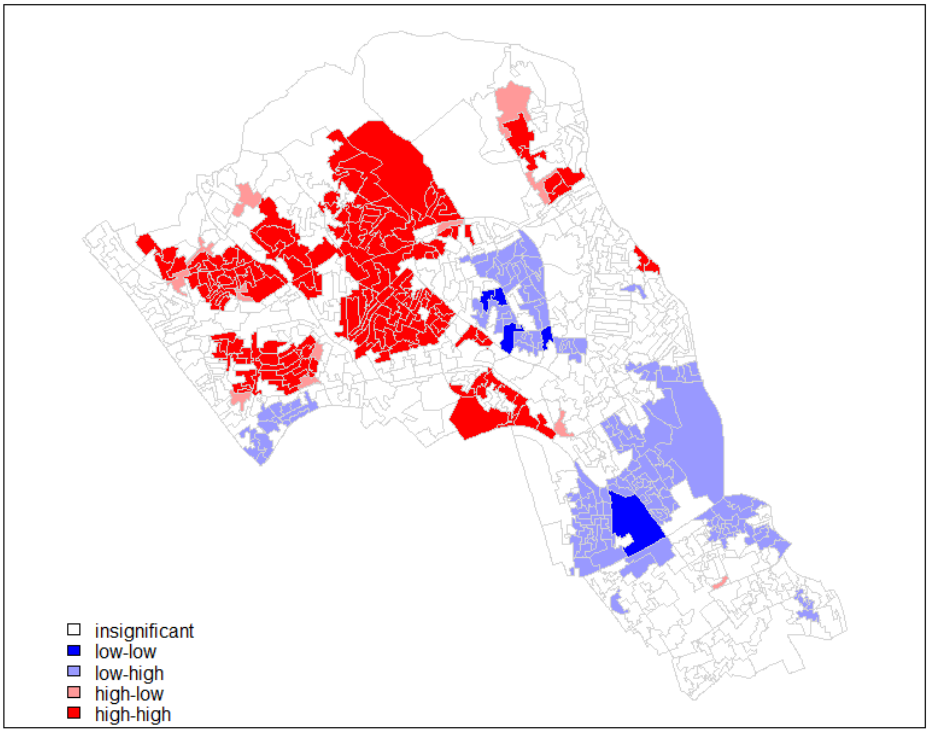

Figure 33. LISA Map showing significant hot spots and cold spots.

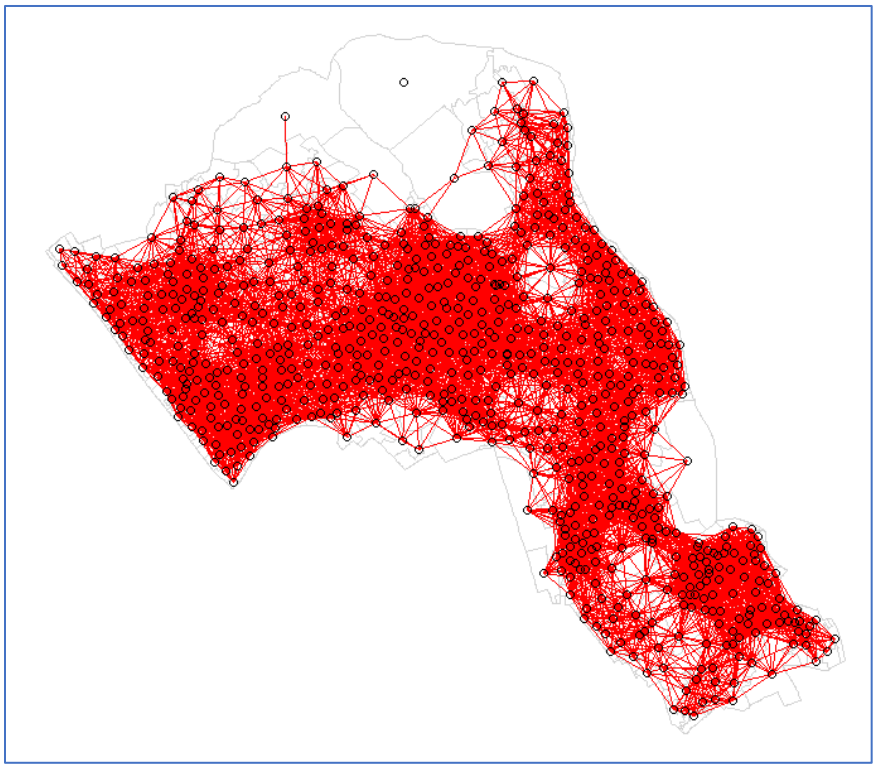

Figure 34. Map showing links between neighbors in proximity of given 500 units distance.

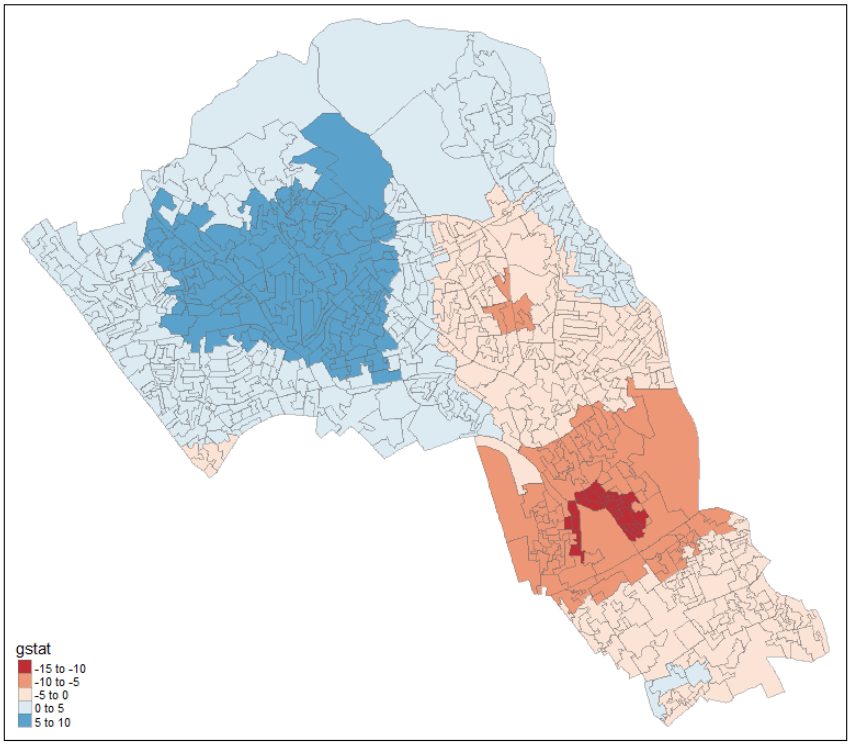

Figure 35. Getis-Ord Gi statistic Map showing the location of hot-spots across Camden.

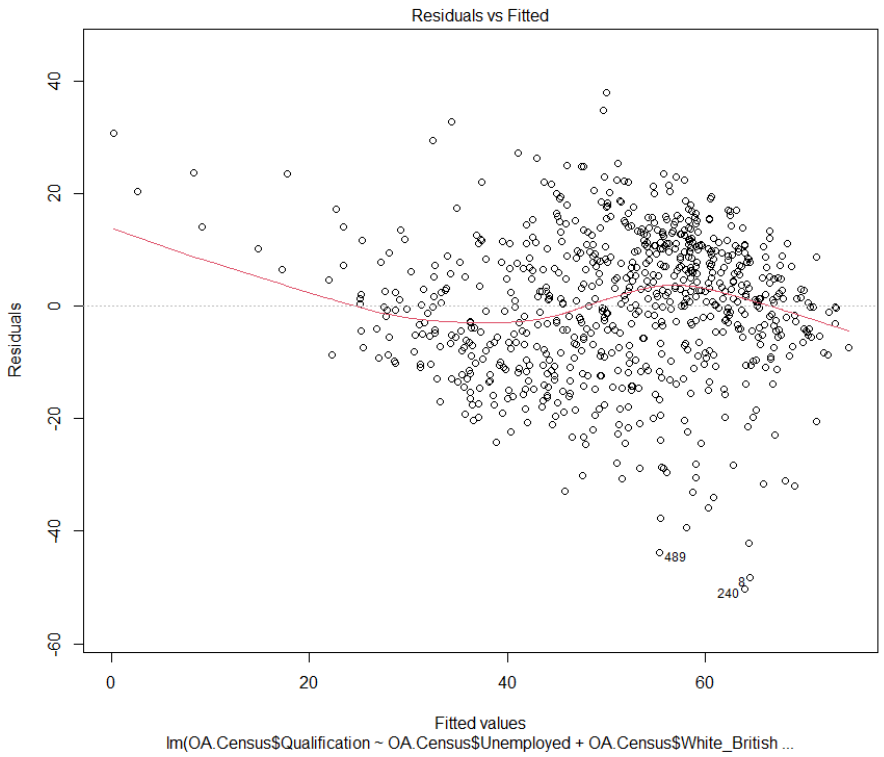

Figure 36. Residual vs Fitted Plot

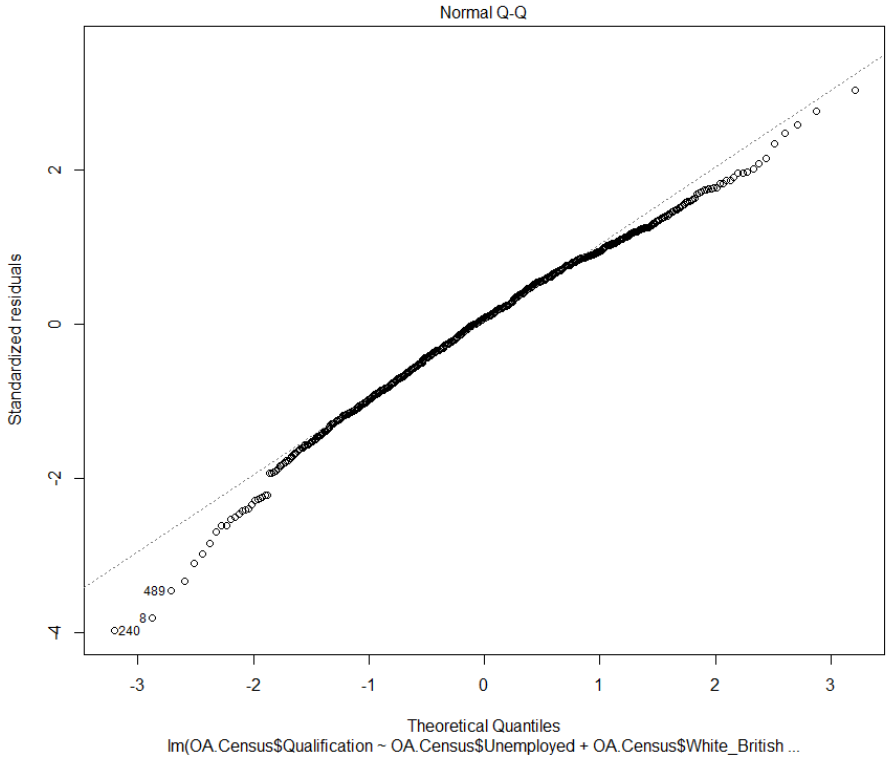

Figure 37. Normal Q-Q Plot

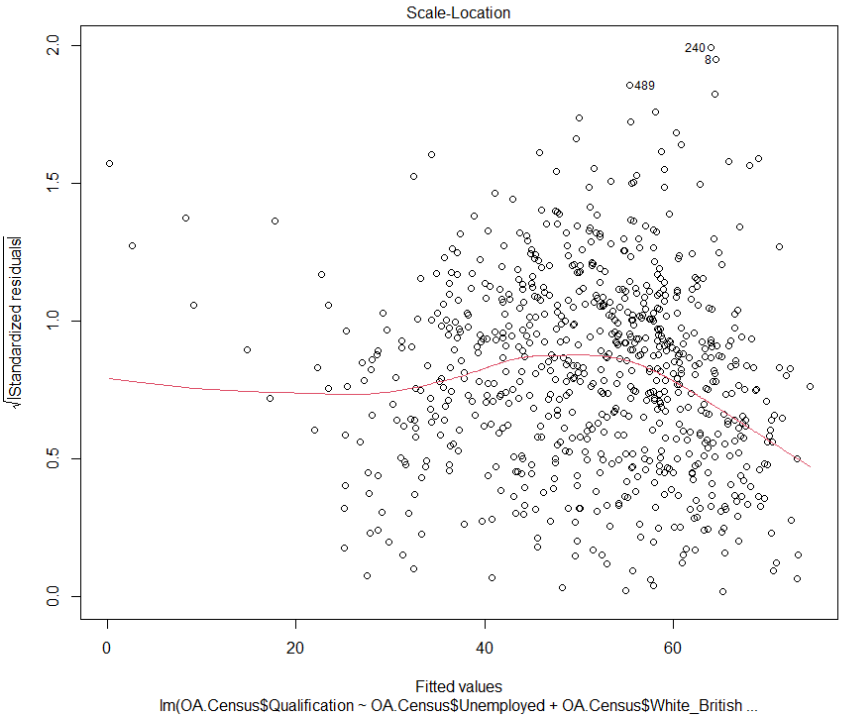

Figure 38. Scale-Location Plot

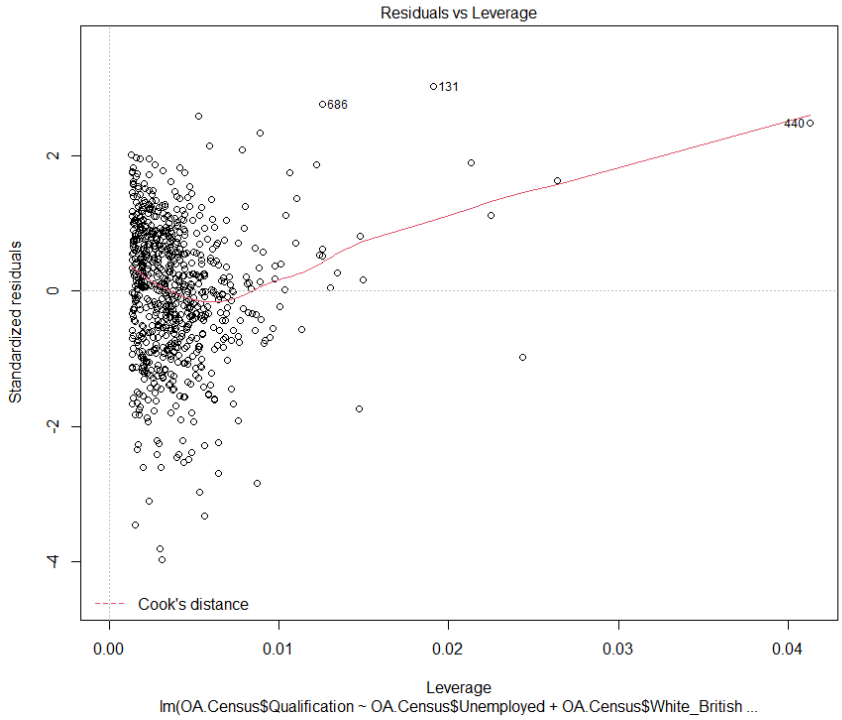

Figure 39. Residuals vs Leverage Plot

Assignment 2: Spatial Analysis and Viz with R (II)

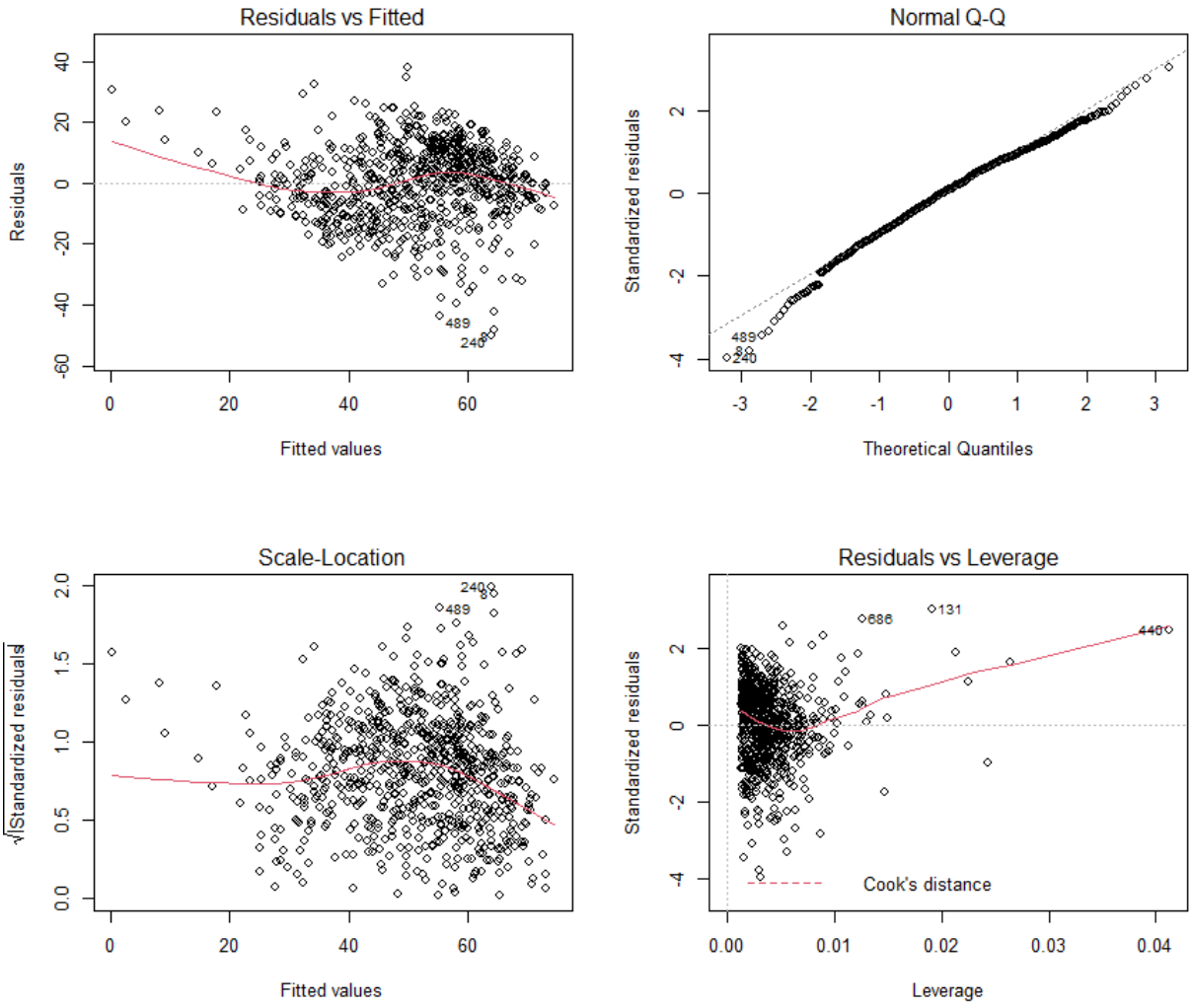

Figure 40. Four Results shown in 2 x 2 frame using par() function.

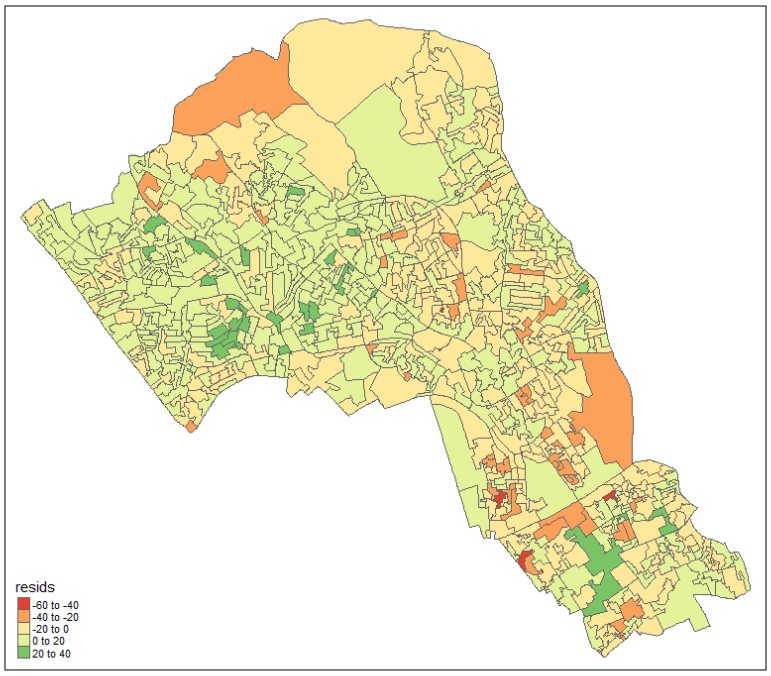

Figure 41. Residual Map using the quickmap function from tmap.

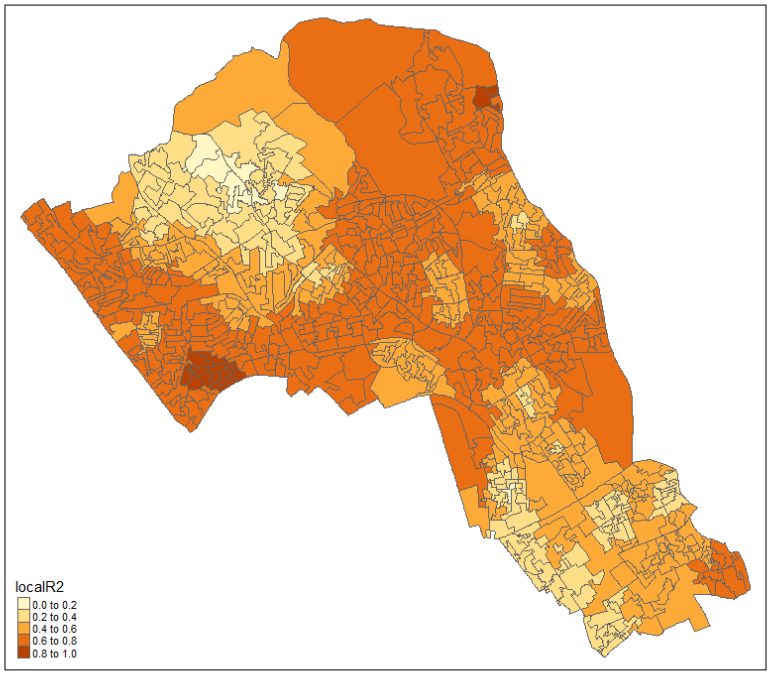

Figure 42. Map showing localR2 obtained from GWR results.

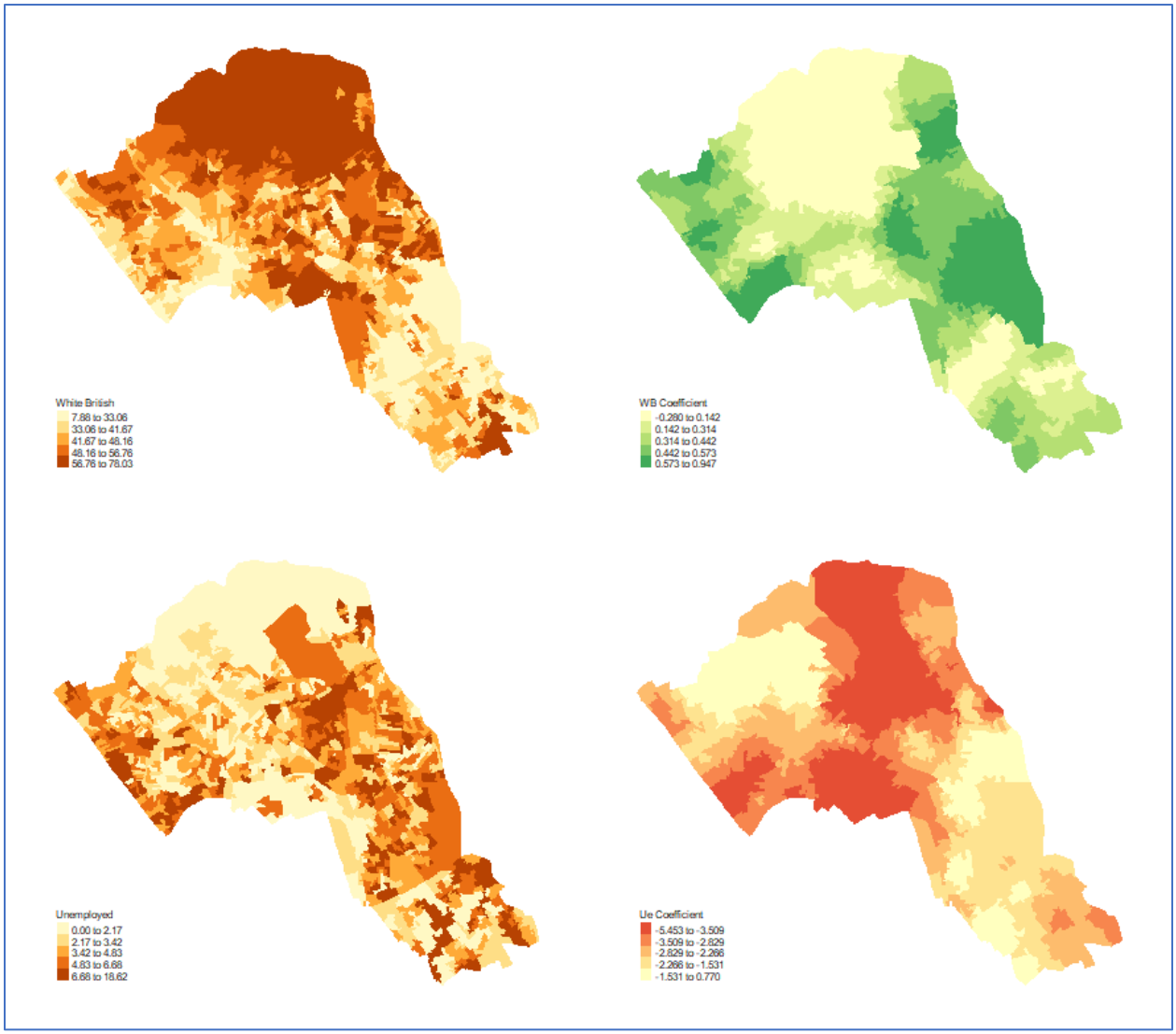

Figure 43. Maps showing original distribution of variables used in GWR Model.

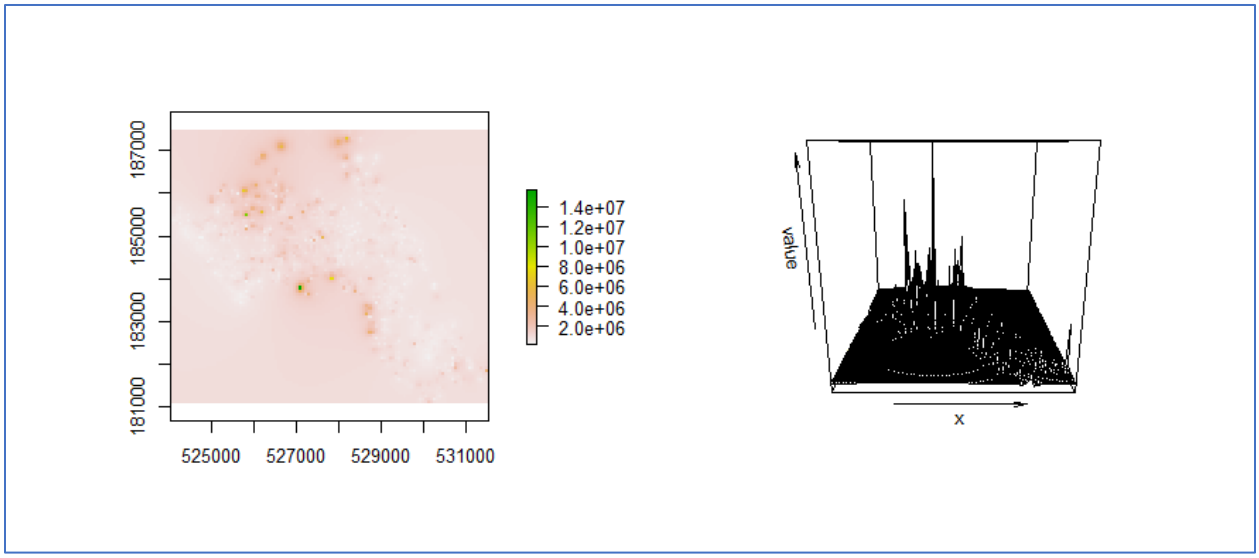

Figure 44. Point data visualization using Inverse Distance Weighting (IDW).

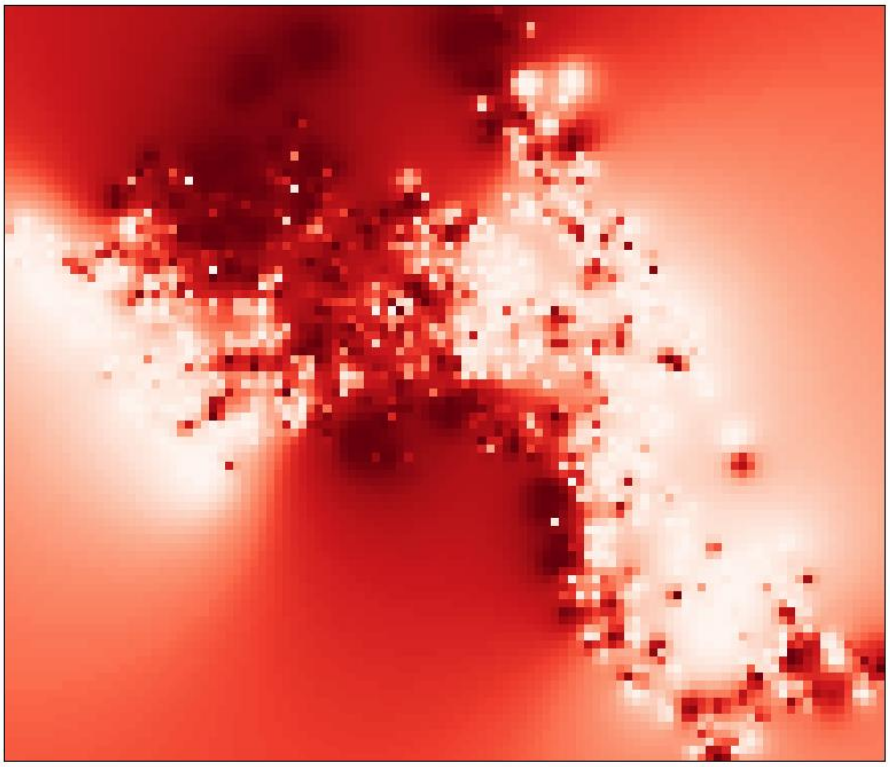

Figure 45. Data visualization by creating raster of IDW for point data.

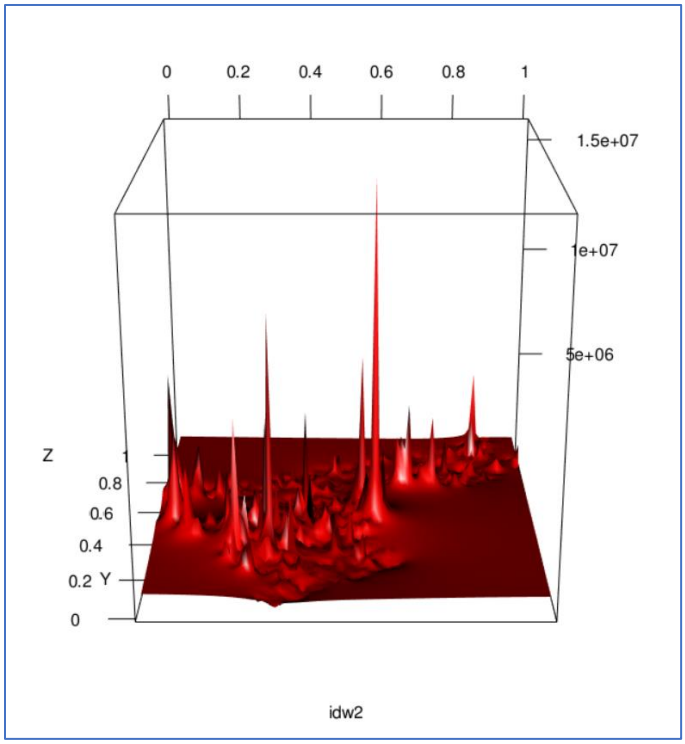

Figure 46. 3D interactive chart using the rgl package and its persp3d() function.

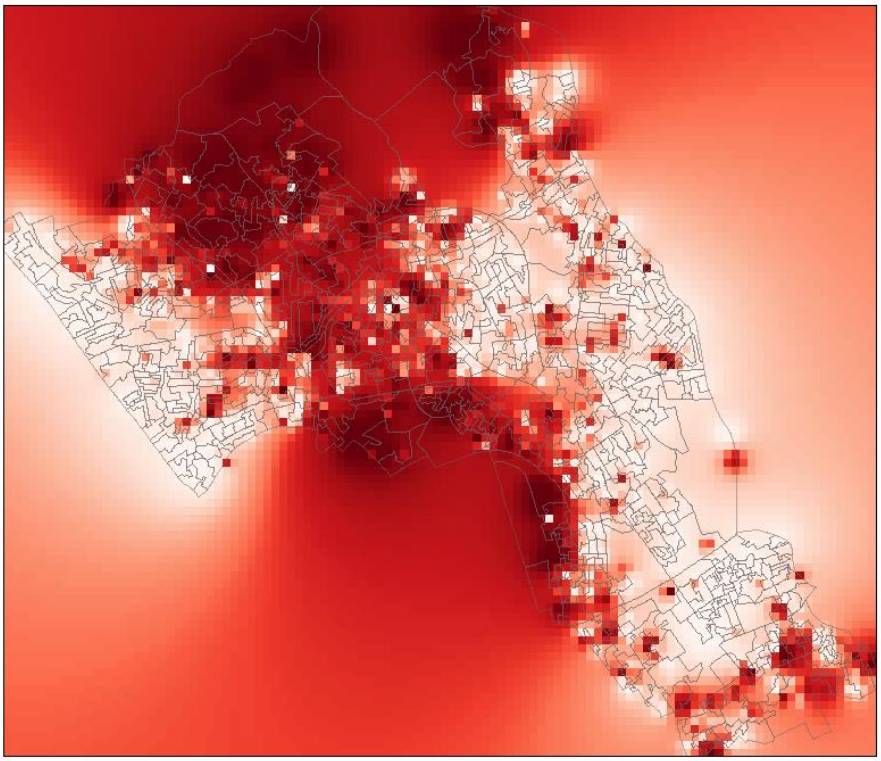

Figure 47. Map showing raster of IDW for point data with polygon shapefile of study area.

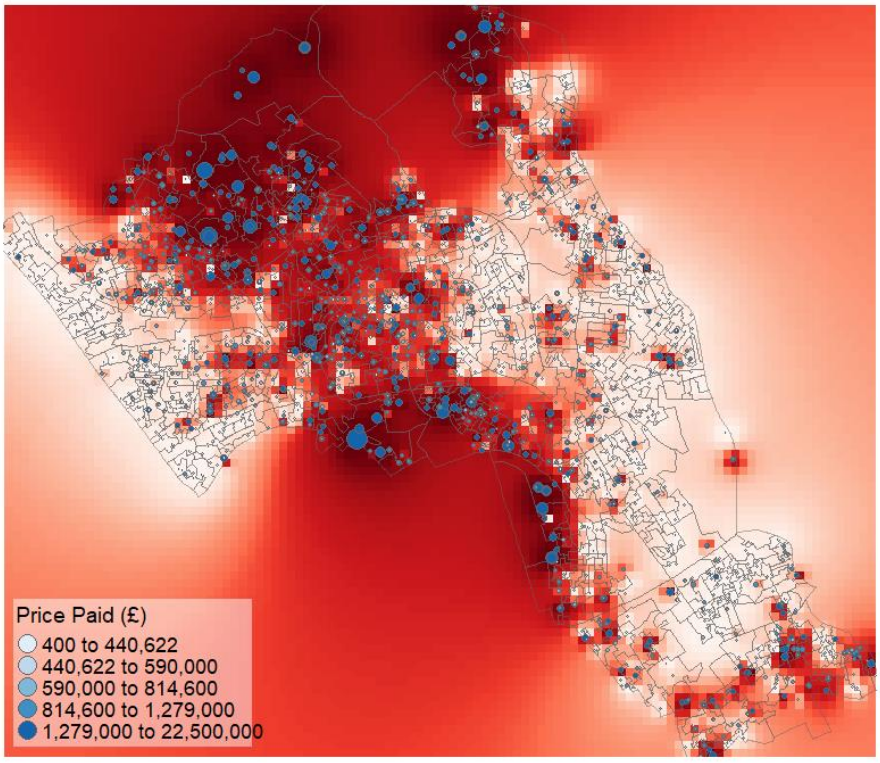

Figure 48. Map showing raster of IDW for point data with proportionate symbols for house prices.

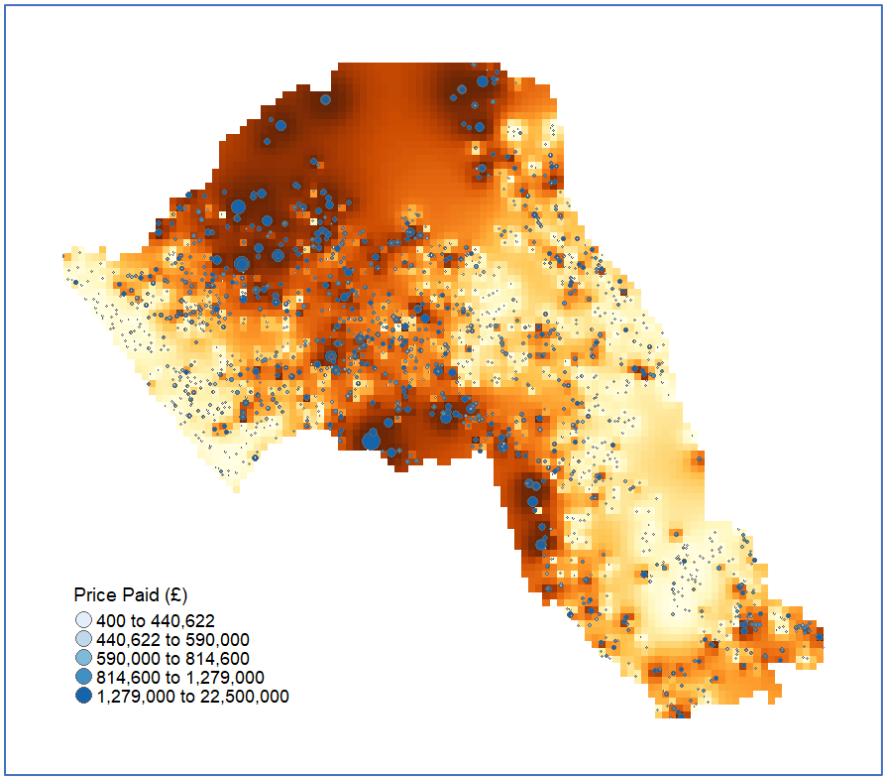

Figure 49. Raster IDW masked to polygon extent with proportionate symbols for house prices.

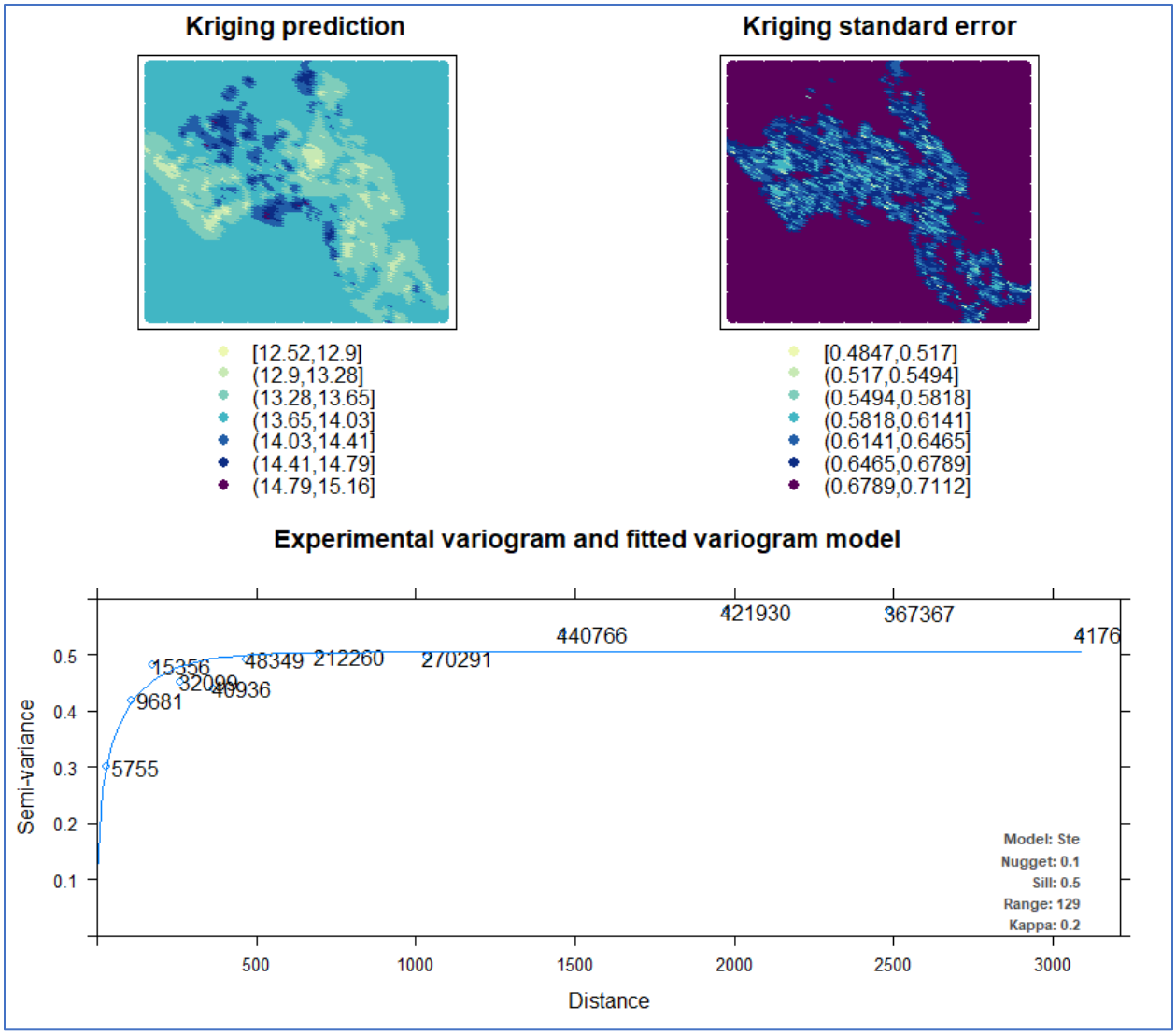

Figure 50. Geostatistical interpolation technique – Kriging Results

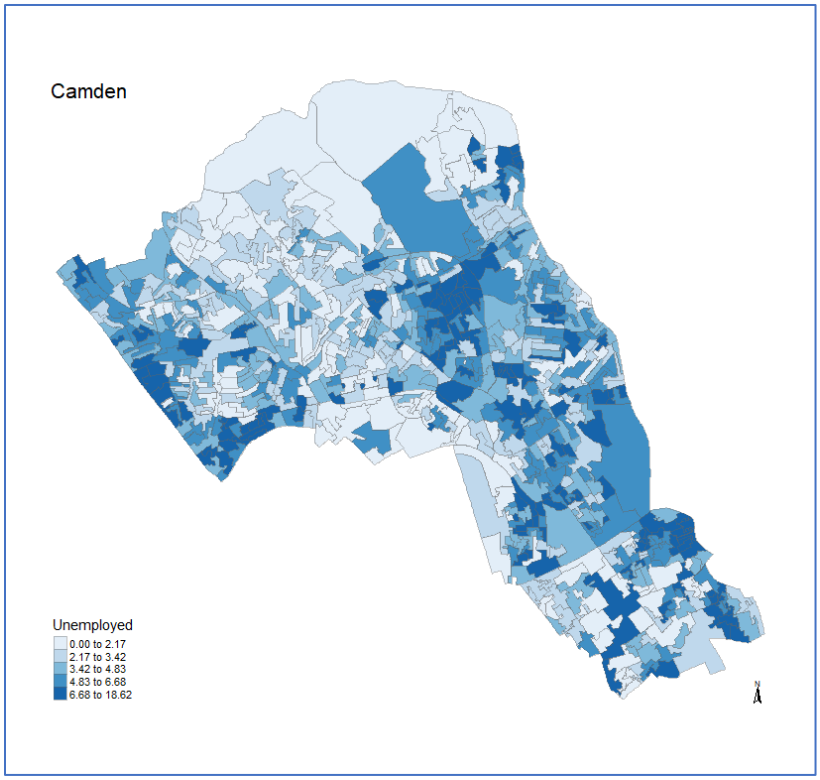

Figure 51. Choropleth map of unemployed data for Camden, created using a user-defined function.

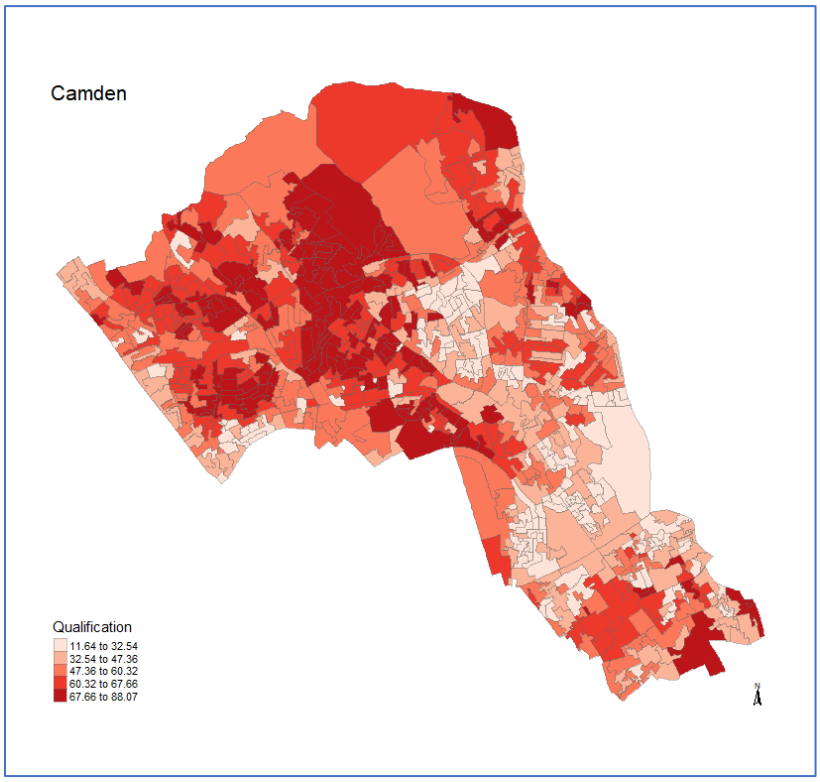

Figure 52. Choropleth map of qualification data for Camden, created using a user-defined function.

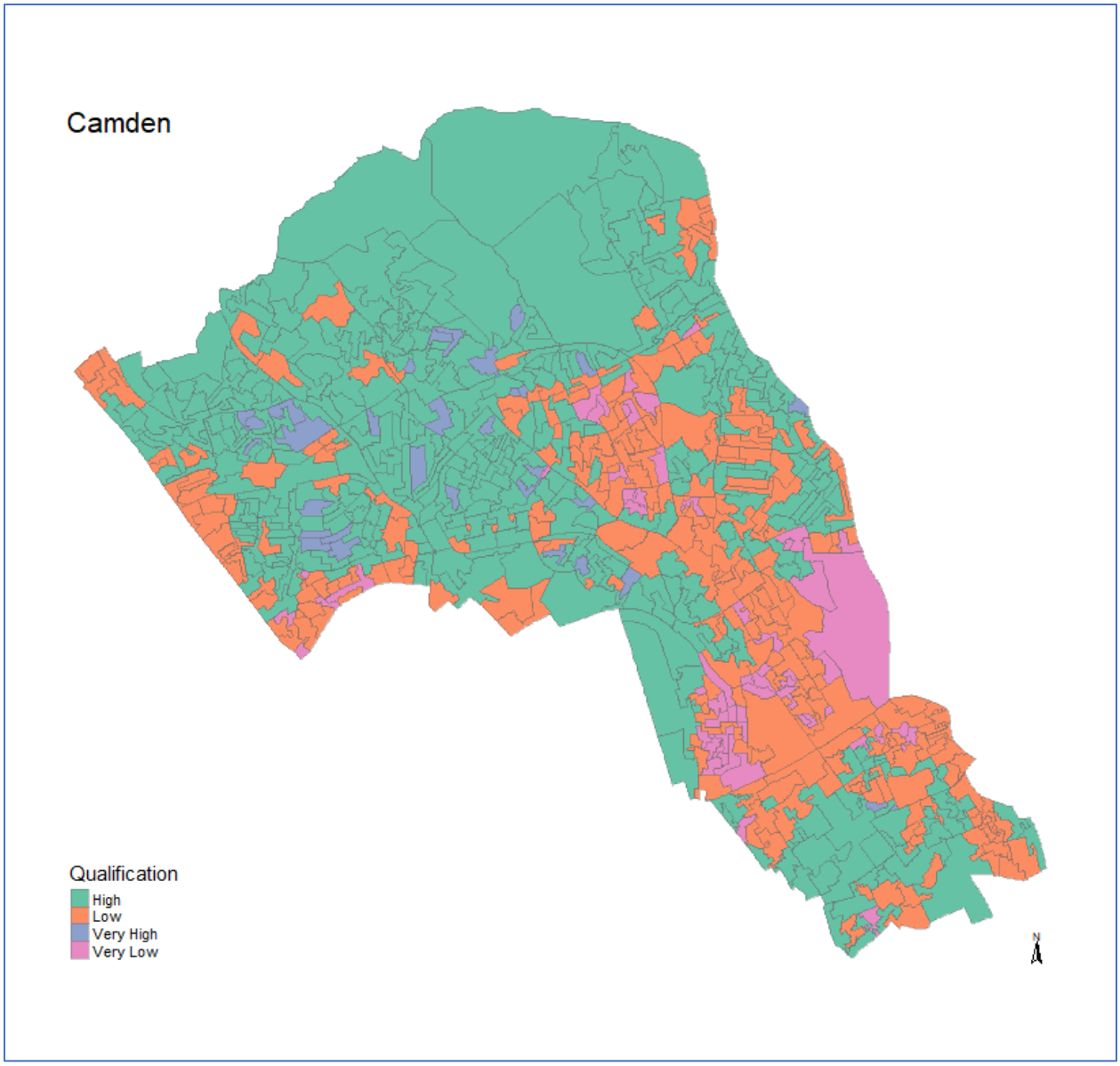

Figure 53. Map showing qualification intervals created using Loops.

3: A 300-500 words reflection summary of what you have learned in this lab.

This lab was more challenging than the previous one. I was introduced with many new libraries and functions, and was definitely amazed with all the cool results we can get with R.

Practical 7 – Using R as GIS, was slightly similar to previous practical where I learnt merging data, and creating maps using tm_shape(). It was a good exercise to learn how to perform various geoprocessing operations such as union and buffer, outside ArcGIS. I was unable to run the part where google map was used. The error was related to google API. I also learned to create interactive maps in R. However, I was not able to save the output as web page. It will be really helpful if I could figure out how to get it work since creating interactive map with R looks easier than that with python. Overall, it was an easy practical in this lab. In practical 8 – Representing Densities in R, I learned how to get kernel density map for point data.

The most interesting practical was practical 9 because I learned how to calculate neighbors for polygons and create map with links between these neighbors. My PPQ project is on spatial lag models which includes influence of neighboring parameters on the dependent parameter. This practical will help me to visualize the number of neighbors and their links for my project. I am really happy to learn this part of the assigned lab.

I learned how to do geographically weighted regression analysis using R in practical 10 and data interpolation in practical 11. Also, learned about R functionality of obtaining 3-dimensional plots to visualize results. I was unable to create the thiessen polygons which were looking pretty cool in the tutorial guide. I also learned how to create custom functions and use For loop and If…Else statements for the first-time using R, in the last practical which is on functions and loops in R.

4. Your R codes.
Practical 7 Code:

```
# Set the working directory
setwd("F:/GEOG 678/Week 2 (26Jan)/Lab Data")

# Load the data.
Census.Data <-read.csv("practicaldata.csv")

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

# Load the output area shapefiles
Output.Areas <- readOGR(".", "Camden_oa11")
# 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")
proj4string(OA.Census) <- CRS("+init=EPSG:27700")
proj4string(House.Points) <- CRS("+init=EPSG:27700")

# point in polygon. Gives the points the attributes of the polygons that they are in
pip <- over(House.Points, OA.Census)

# need to bind the census data to our original points
House.Points@data <- cbind(House.Points@data, pip)

View(House.Points@data)

# it is now possible to plot the house prices and local unemployment rates
plot(log(House.Points@data$Price), House.Points@data$Unemployed)

# first we aggregate the house prices by the OA11CD (OA names) column
# we ask for the mean for each OA
OA <- aggregate(House.Points@data$Price, by = list(House.Points@data$OA11CD), mean)
```

Assignment 2: Spatial Analysis and Viz with R (II)

```r
# change the column names of the aggregated data
names(OA) <- c("OA11CD", "Price")

# join the aggregated data back to the OA.Census polygon
OA.Census@data <- merge(OA.Census@data, OA, by = "OA11CD", all.x = TRUE)

library(tmap)

tm_shape(OA.Census) + tm_fill(col = "Price", style = "quantile", title = "Mean House Price (£)")

model <- lm(OA.Census@data$Price ~ OA.Census@data$Unemployed)

summary(model)

# create 200m buffers for each house point
house_buffers <- gBuffer(House.Points, width = 200, byid = TRUE)

# map in tmap
tm_shape(OA.Census) + tm_borders() +
tm_shape(house_buffers) + tm_borders(col = "blue") +
tm_shape(House.Points) + tm_dots(col = "red")
```

```r
# merges the buffers
union.buffers <- gUnaryUnion(house_buffers)

# map in tmap
tm_shape(OA.Census) + tm_borders() +
tm_shape(union.buffers) + tm_fill(col = "blue", alpha = .4) + tm_borders(col = "blue") +
tm_shape(House.Points) + tm_dots(col = "red")
```

```r
# interactive maps in tmap
library(leaflet)

# turns view map on
tmap_mode("view")

tm_shape(House.Points) + tm_dots(title = "House Prices (£)", border.col = "black",
border.lwd = 0.1, border.alpha = 0.2, col = "Price",
style = "quantile", palette = "Reds")

tm_shape(House.Points) + tm_bubbles(size = "Price", title.size = "House Prices (£)",
border.col = "black", border.lwd = 0.1, border.alpha = 0.4, legend.size.show = TRUE)

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

tmap_mode("plot")
```

Practical 8 Code:

```
library("sp")
library("rgdal")
library("rgeos")

# Load the output area shapefiles, we won't join it to any data this time
Output.Areas <- readOGR(".", "Camden_oa11")

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

# load the spatial libraries
library(raster)
library(adehabitatHR)

# runs the kernel density estimation,look up the function parameters for more options
kde.output <- kernelUD(House.Points, h="href", grid = 1000)

plot(kde.output)

# converts to raster
kde <- raster(kde.output)

# sets projection to British National Grid
projection(kde) <- CRS("+init=EPSG:27700")


library(tmap)

# maps the raster in tmap, "ud" is the density variable
tm_shape(kde) + tm_raster("ud")

library(tmaptools)

# creates a bounding box based on the extents of the Output.Areas polygon
bounding_box <- bb(Output.Areas)

# maps the raster within the bounding box
tm_shape(kde, bbox = bounding_box) + tm_raster("ud")
```

```
# mask the raster by the output area polygon
masked_kde <- mask(kde, Output.Areas)

# maps the masked raster, also maps white output area boundaries
tm_shape(masked_kde, bbox = bounding_box) + tm_raster("ud", style = "quantile",
                                                      n = 100,
                                                      legend.show = FALSE,
                                                      palette = "YlGnBu") +
  tm_shape(Output.Areas) + tm_borders(alpha=.3, col = "white") +
  tm_layout(frame = FALSE)


# create catchment boundaries from the kernel density estimations.
# compute homeranges for 75%, 50%, 25% of points,
# objects are returned as spatial polygon data frames
range75 <- getverticeshr(kde.output, percent = 75)
range50 <- getverticeshr(kde.output, percent = 50)
range25 <- getverticeshr(kde.output, percent = 25)

# Create a grey background using the Output.Areas polygon with white borders
# Plot the locations of houses using House.Points
# Plot the 75% range, set attributes for border and fill (i.e. colour, transparency, line width)
# Plot the 50% range, set attributes for border and fill (i.e. colour, transparency, line width)
# Plot the 25% range, set attributes for border and fill (i.e. colour, transparency, line width)
# The outside frame is removed

# the code below creates a map of several layers using tmap
tm_shape(Output.Areas) + tm_fill(col = "#f0f0f0") + tm_borders(alpha=.8, col = "white") +
  tm_shape(House.Points) + tm_dots(col = "blue") +
  tm_shape(range75) + tm_borders(alpha=.7, col = "#fb6a4a", lwd = 2) +
  tm_fill(alpha=.1, col = "#fb6a4a") +
  tm_shape(range50) + tm_borders(alpha=.7, col = "#de2d26", lwd = 2) +
  tm_fill(alpha=.1, col = "#de2d26") +
  tm_shape(range25) + tm_borders(alpha=.7, col = "#a50f15", lwd = 2) +
  tm_fill(alpha=.1, col = "#a50f15") +
  tm_layout(frame = FALSE)
```

Practical 9 Code:

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

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

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

# 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")

library("tmap")

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

library(spdep)

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

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

# 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')

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

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

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

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

# 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")

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

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

Assignment 2: Spatial Analysis and Viz with R (II)

```
# 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")
```

```
# creates centroid and joins neighbours within 0 and x units
# neighbours based on proximity
nb <- dnearneigh(coordinates(OA.Census),0,500)

# 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')

# 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_borders(alpha=.4)
```

Practical 10 Code:

```
# Set the working directory
setwd("F:/GEOG 678/Week 2 (26Jan)/Lab Data")

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

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

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

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

# runs a linear model
model <- lm(OA.Census$Qualification ~ OA.Census$Unemployed+OA.Census$White_British)
summary(model)

# this will plot 4 scatter plots from the linear model
plot(model)

# we can use the par function if we want to plot them in a 2x2 frame
par(mfrow=c(2,2))
plot(model)

#map the residuals to see if there is a spatial distribution of them across Camden.
resids<-residuals(model)
map.resids <- cbind(OA.Census, resids)

# we need to rename the column header from the resids file
# in this case its the 6th column of map.resids
names(map.resids)[6] <- "resids"

# maps the residuals using the quickmap function from tmap
qtm(map.resids, fill = "resids")
```

Assignment 2: Spatial Analysis and Viz with R (II)

```
#GWR Model
library("spgwr")

#calculate kernel bandwidth
GWRbandwidth <- gwr.sel(OA.Census$Qualification ~ OA.Census$Unemployed +
                        OA.Census$White_British, data=OA.Census, adapt =TRUE)

#run the gwr model
gwr.model = gwr(OA.Census$Qualification ~ OA.Census$Unemployed+OA.Census$White_British,
                data = OA.Census, adapt=GWRbandwidth, hatmatrix=TRUE, se.fit=TRUE)

#print the results of the model
gwr.model

results <-as.data.frame(gwr.model$SDF)
names(results)
gwr.map <- cbind(OA.Census, as.matrix(results))
qtm(gwr.map, fill = "localR2")

# create tmap objects
map1 <- tm_shape(gwr.map) + tm_fill("White_British", n = 5, style = "quantile",
                                    title = "White British") +
  tm_layout(frame = FALSE, legend.text.size = 0.5, legend.title.size = 0.6)

map2 <- tm_shape(gwr.map) + tm_fill("OA.Census.White_British", n = 5, style = "quantile",
                                    title = "WB Coefficient") +
  tm_layout(frame = FALSE, legend.text.size = 0.5, legend.title.size = 0.6)
map3 <- tm_shape(gwr.map) + tm_fill("Unemployed", n = 5, style = "quantile",
                                    title = "Unemployed") +
  tm_layout(frame = FALSE, legend.text.size = 0.5, legend.title.size = 0.6)
map4 <- tm_shape(gwr.map) + tm_fill("OA.Census.Unemployed", n = 5, style = "quantile",
                                    title = "Ue Coefficient") +
  tm_layout(frame = FALSE, legend.text.size = 0.5, legend.title.size = 0.6)
```

```
library(grid)
library(gridExtra)

# creates a clear grid
grid.newpage()

# assigns the cell size of the grid, in this case 2 by 2
pushViewport(viewport(layout=grid.layout(2,2)))

# prints a map object into a defined cell
print(map1, vp=viewport(layout.pos.col = 1, layout.pos.row =1))
print(map2, vp=viewport(layout.pos.col = 2, layout.pos.row =1))
print(map3, vp=viewport(layout.pos.col = 1, layout.pos.row =2))
print(map4, vp=viewport(layout.pos.col = 2, layout.pos.row =2))
```

Practical 11 Code:

```
> plot(kriging_result)
> library("sp")
> library("rgdal")
> library("rgeos")
> library("tmap")
> Output.Areas <- readOGR(".", "Camden_oa11")
OGR data source with driver: ESRI Shapefile
Source: "F:\GEOG 678\Week 2 (26Jan)\Lab Data", layer: "Camden_oa11"
with 749 features
It has 1 fields
> House.Points <- readOGR(".", "Camden_house_sales")
OGR data source with driver: ESRI Shapefile
Source: "F:\GEOG 678\Week 2 (26Jan)\Lab Data", layer: "Camden_house_sales"
with 2547 features
It has 4 fields
> library(spatstat)
> library(maptools)
> dat.pp <- as(dirichlet(as.ppp(House.Points)), "SpatialPolygons")
> dat.pp <- as(dat.pp,"SpatialPolygons")
> proj4string(dat.pp) <- CRS("+init=EPSG:27700")
> proj4string(House.Points) <- CRS("+init=EPSG:27700")
> int.Z <- over(dat.pp,House.Points, fn=mean)
```

```
# Create a SpatialPolygonsDataFrame
thiessen <- SpatialPolygonsDataFrame(dat.pp, int.Z)

# maps the thiessen polygons and House.Points
tm_shape(Output.Areas) + tm_fill(alpha=.3, col = "grey") + tm_shape(thiessen) + tm_borders(alpha=.5, col = "black") +
  tm_shape(House.Points) + tm_dots(col = "blue", scale = 0.5)

library(raster)
# crops the polygon by our output area shapefile
thiessen.crop <-crop(thiessen, Output.Areas)
# maps cropped thiessen polygons and House.Points
tm_shape(Output.Areas) + tm_fill(alpha=.3, col = "grey") + tm_shape(thiessen.crop) + tm_borders(alpha=.5, col = "black") +
  tm_shape(House.Points) + tm_dots(col = "blue", scale = 0.5)

# maps house prices across thiessen polygons
tm_shape(thiessen.crop) + tm_fill(col = "Price", style = "quantile", palette = "Reds", title = "Price Paid (£)") +
  tm_borders(alpha=.3, col = "black") +
  tm_shape(House.Points) + tm_dots(col = "black", scale = 0.5) +
  tm_layout(legend.position = c("left", "bottom"), legend.text.size = 1.05,legend.title.size = 1.2, frame = FALSE)
```

Figure 54. Unable to execute this part of code since int.z variable was not created.

Assignment 2: Spatial Analysis and Viz with R (II)

```r
#To run an IDW
library(gstat)
library(xts)

# define sample grid based on the extent of the House.Points file
grid <-spsample(House.Points, type = 'regular', n = 10000)

# runs the idw for the Price variable of House.Points
idw <- idw(House.Points$Price ~ 1, House.Points, newdata= grid)

idw.output = as.data.frame(idw)
names(idw.output)[1:3] <- c("long", "lat", "prediction")

library(raster)

# create spatial points data frame
spg <- idw.output
coordinates(spg) <- ~ long + lat

# coerce to SpatialPixelsDataFrame
gridded(spg) <- TRUE

# coerce to raster
raster_idw <- raster(spg)

# sets projection to British National Grid
projection(raster_idw) <- CRS("+init=EPSG:27700")

# we can quickly plot the raster to check its okay
plot(raster_idw)

persp(raster_idw)

# this package lets us create interactive 3d visualisations
library(rgl)

# we need to first convert the raster to a matrix object type
idw2 <- as.matrix(raster_idw)

persp3d(idw2, col = "red")
```

Assignment 2: Spatial Analysis and Viz with R (II)

```
library(tmap)
tm_shape(raster_idw) + tm_raster("prediction", style = "quantile", n = 100,
                                 palette = "Reds", legend.show = FALSE)
tm_shape(raster_idw) + tm_raster("prediction", style = "quantile", n = 100,
                                 palette = "Reds", legend.show = FALSE) +
  tm_shape(Output.Areas) + tm_borders(alpha=.5)

tm_shape(raster_idw) + tm_raster("prediction", style = "quantile", n = 100,
                                 palette = "Reds",legend.show = FALSE) +
  tm_shape(Output.Areas) + tm_borders(alpha=.5,) +
  tm_shape(House.Points) + tm_bubbles(size = "Price", col = "Price",
                                      palette = "Blues", style = "quantile",
                                      legend.size.show = FALSE,
                                      title.col = "Price Paid (£)") +
  tm_layout(legend.position = c("left", "bottom"), legend.text.size = 1.1,
            legend.title.size = 1.4,frame = FALSE, legend.bg.color = "white",
            legend.bg.alpha = 0.5)

# masks our raster by our output areas polygon file
masked_idw <- mask(raster_idw, Output.Areas)

# plots the masked raster
tm_shape(masked_idw) + tm_raster("prediction", style = "quantile", n = 100,
                                 legend.show = FALSE) +
  tm_shape(House.Points) + tm_bubbles(size = "Price", col = "Price",
                                      palette = "Blues", style = "quantile",
                                      legend.size.show = FALSE,
                                      title.col = "Price Paid (£)") +
  tm_layout(legend.position = c("left", "bottom"), legend.text.size = 1.1,
            legend.title.size = 1.4, frame = FALSE)

library(automap)
# this is the same grid as before
grid <-spsample(House.Points, type = 'regular', n = 10000)

# runs the kriging
kriging_result = autoKrige(log(Price)~1, House.Points, grid)

plot(kriging_result)
```

Practical 12 Code:

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

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

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

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

# rounds data to one decimal place
round(Census.Data$Qualification, 1)

# function is called "myfunction", accepts one object (to be called x)
myfunction <- function(x){
  # x is rounded to one decimal place, creates object z
  z <- round(x,1)
  # object z is returned
  return(z)
  # close function
}

# lets create a new data frame so we don't overwrite our original data object
newdata <- Census.Data

# runs the function, returns the output to new Qualification_2 variable in newdata
newdata$Qualificaton_2 <- myfunction(Census.Data$Qualification)

# creates new function which logs and then rounds data
myfunction <- function(x){
  z <- log(x)
  y <- round(z,4)
  return(y)
}

# resets the newdata object so we can run it again
newdata <- Census.Data

# runs the function
newdata$Qualification_2 <- myfunction(Census.Data$Qualification)
newdata$Unemployed_2 <- myfunction(Census.Data$Unemployed)

myfunction <- function(x){
  z <- log(x)
  z <- round(z,4)
  return(z)
}
```

Assignment 2: Spatial Analysis and Viz with R (II)

```
# map function with 3 arguments
map <- function(x,y,z){
  tm_shape(x) + tm_fill(y, palette = z, style = "quantile") + tm_borders(alpha=.4) +
    tm_compass(size = 1.8, fontsize = 0.5) +
    tm_layout(title = "Camden", legend.title.size = 1.1, frame = FALSE)
}

# runs map function, remember we need to include all 3 arguments of the function
map(OA.Census, "Unemployed", "Blues")

map(OA.Census, "Qualification", "Reds")

#create a new data frame of the same properties as our data file
newdata <- Census.Data

# a for loop where i iterates from 2 to 5
for(i in 2:5){
  # i is used to identify the column number
  newdata[, i] <- round(Census.Data[,i], 1)
}

# open the new data
View(newdata)

# use ncol to determine the number of columns
for(i in 2: ncol (Census.Data)){
  newdata[, i] <- Census.Data[,i]/100
}

#creates a new newdata object so we have it saved somewhere
newdata1 <- newdata
for(i in 1:nrow(newdata)){
  if (newdata$White_British[i] < 0.5) {
    newdata$White_British[i] <- "Low";
  } else {
    newdata$White_British[i] <- "High";
  }
}

# copies the numbers back to newdata so we can start again
newdata <- newdata1

for(i in 1:nrow(newdata)){
  if (newdata$White_British[i] < 0.25) {
    newdata$White_British[i] <- "Very Low";
  } else if (newdata$White_British[i] < 0.50){
    newdata$White_British[i] <- "Low";
  } else if (newdata$White_British[i] < 0.75){
    newdata$White_British[i] <- "High";
  } else {
    newdata$White_British[i] <- "Very High";
  }
}
```

```
# copies the numbers back to newdata so we can start again
newdata <- newdata1
for(j in 2: ncol (newdata)){
  for(i in 1:nrow(newdata)){
    if (newdata[i,j] < 0.25) {
      newdata[i,j] <- "Very Low";
    } else if (newdata[i,j] < 0.50){
      newdata[i,j] <- "Low";
    } else if (newdata[i,j] < 0.75){
      newdata[i,j] <- "High";
    } else {
      newdata[i,j] <- "Very High";
    }
  }
}

# merge our new formatted data with the output areas shapefile
shapefile <- merge(Output.Areas, newdata, by.x = "OA11CD", by.y = "OA")

# runs our predefined map function
map(shapefile, "Qualification", "Set2")
```