Practical 10: Geographically Weighted Regression in R

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

This tutorial will teach you how to run a Geographically Weighted Regression (GWR). GWR is a multivariate model which can indicate where non-stationarity may take place across space; it can be used to identify how locally weighted regression coefficients may vary across the study area. We will first explore the residuals of a linear model to understand its limitations. Next, we will run a GWR and observe its parameters across space. 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 tutorial we will:

- Run a linear model to predict the occurrence of a variable across small areas
- Run a geographically weighted regression to understand how models may vary across space
- Print multiple maps in one graphic using gridExtra() with tmap

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 our spatial data files.

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

# 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")
```

Run a linear model

First, we will run a linear model to understand the global relationship between our variables in our study area. In this case, the percentage of people with qualifications is our dependent variable, and the percentages of unemployed economically active adults and White British ethnicity are our two independent (or predictor) variables.

```
# runs a linear model
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 ***
## ---
## 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
```

As we saw in Practical 4, there is lots of interesting information we can derive from a linear regression model output. This model has an adjusted R-squared value of 0.463. So we can assume 46% of the variance can be explained by the model. We can also observe the influences of each of the variables. However, the overall fit of the model and each of the coefficients may vary across space if we consider different parts of our study area. It is, therefore, worth considering the standardised residuals from the model to help us understand and improve our future models.

A residual is the difference between the predicted and observed values for an observation in the model. Models with lower r-squared values would have greater residuals on average as the data would not fit the modelled regression line as well. Standardised residuals are represented as Z-scores where 0 represent the predicted values.

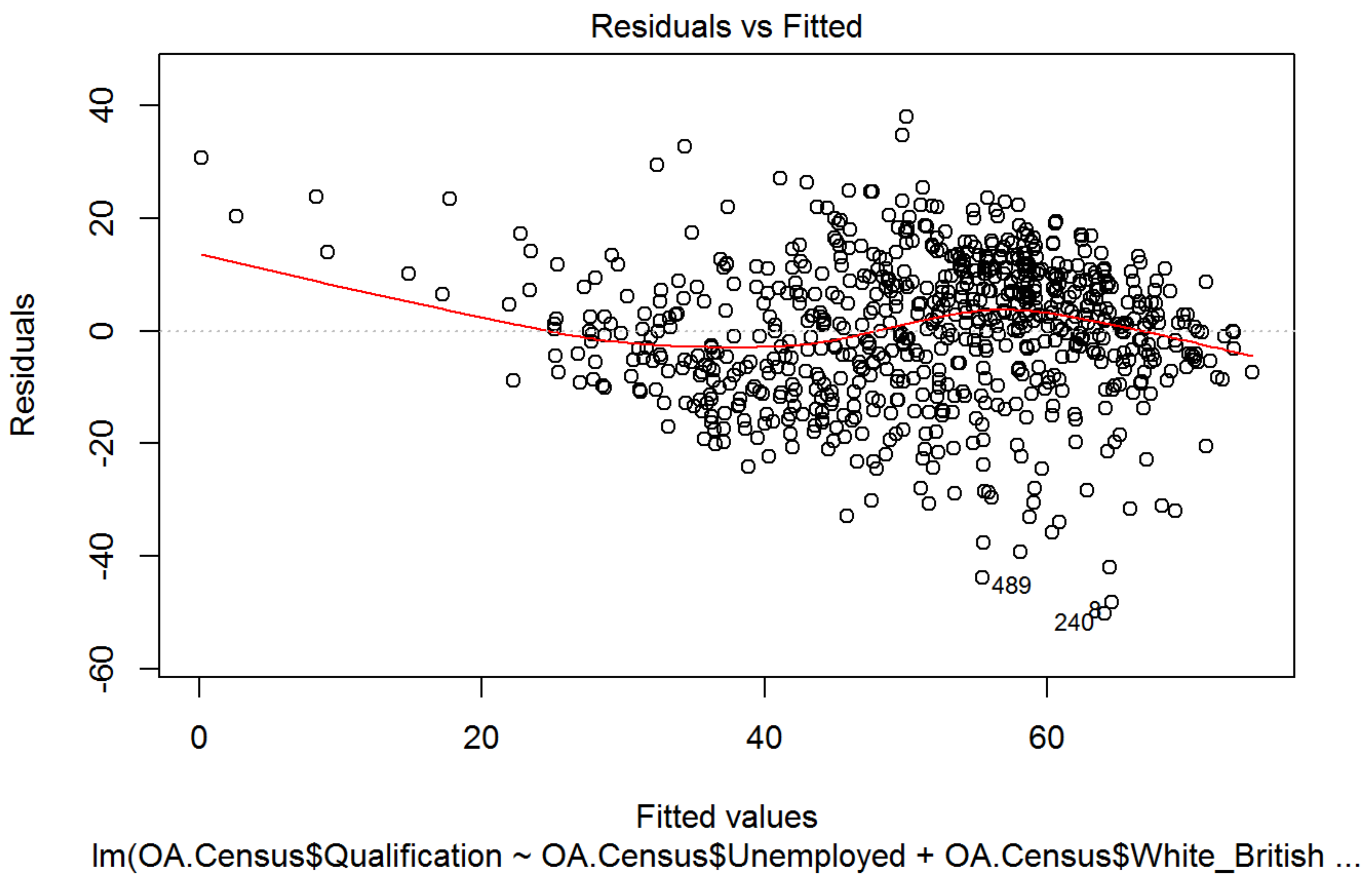
If you plot a linear model (i.e. our *model* object), R will print out four different plots of which are useful for evaluating the model fit. These are very briefly summarised as:

- **Residuals vs Fitted** - considers the relationship between the actual and the predicted data. The more dispersed the residuals are, the weaker the R2 should be. This can be useful to identify outliers too.The fit also tells us if the residuals are non-linearly distributed.
- **Normal Q-Q** - Demonstrates the extent to which the residuals are normally distributed. Normal residuals should fit the straight line well.

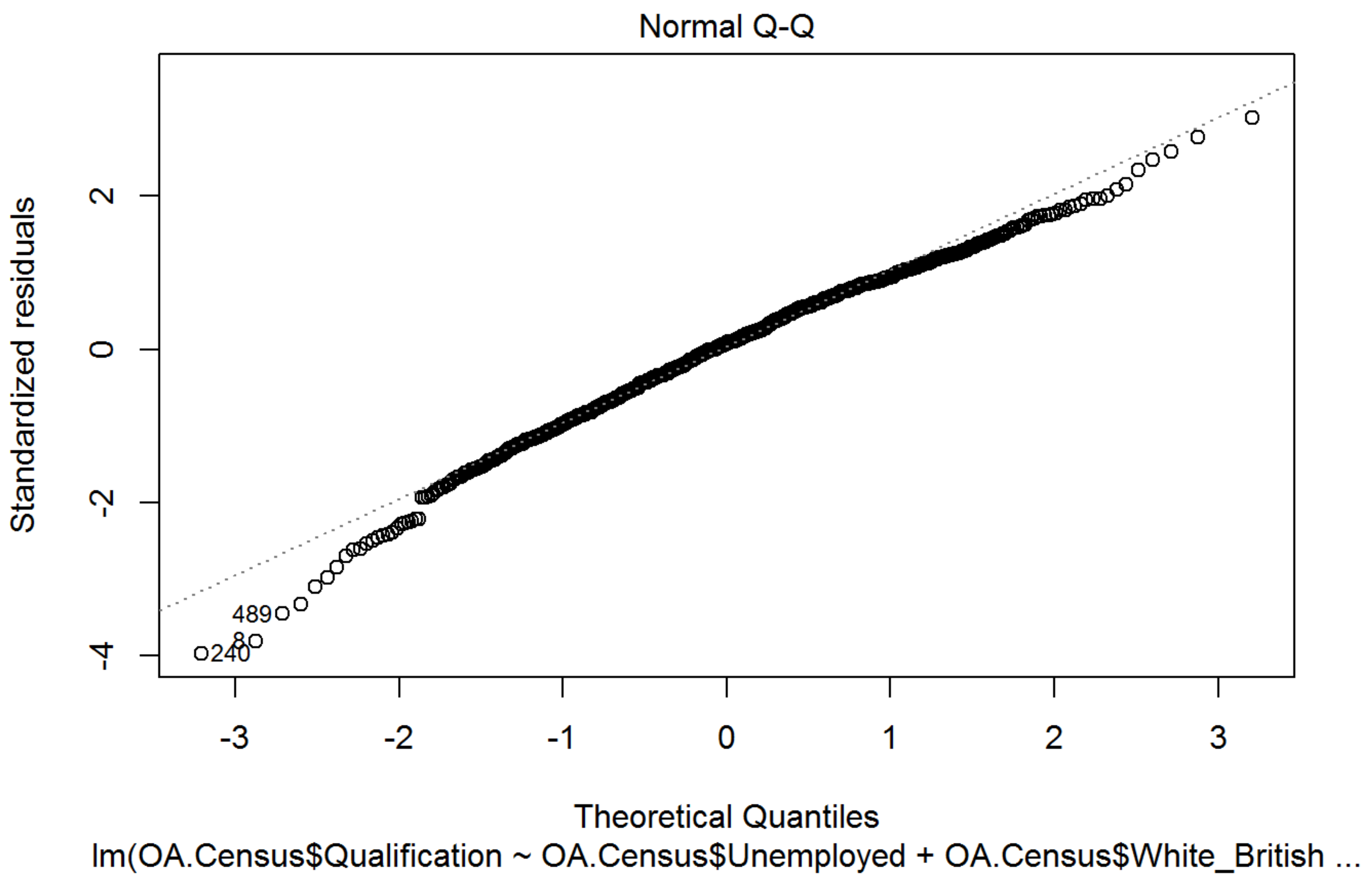
- **Scale-Location** - Shows if the residuals are spread equally across the full range of the predictors. If the values in this chart display a linear positive relationship, it suggests that the residuals spread wider and wider for greater values (this is known as heteroscedasticity).
- **Residuals vs Leverage** - this graph identifies outliers, high-leverage points and influential observations. This plot is pretty difficult to interpret and there are other means of identifying these values.

A good description of these plots and how to interpret them can be found here (http://data.library.virginia.edu/diagnostic-plots/).

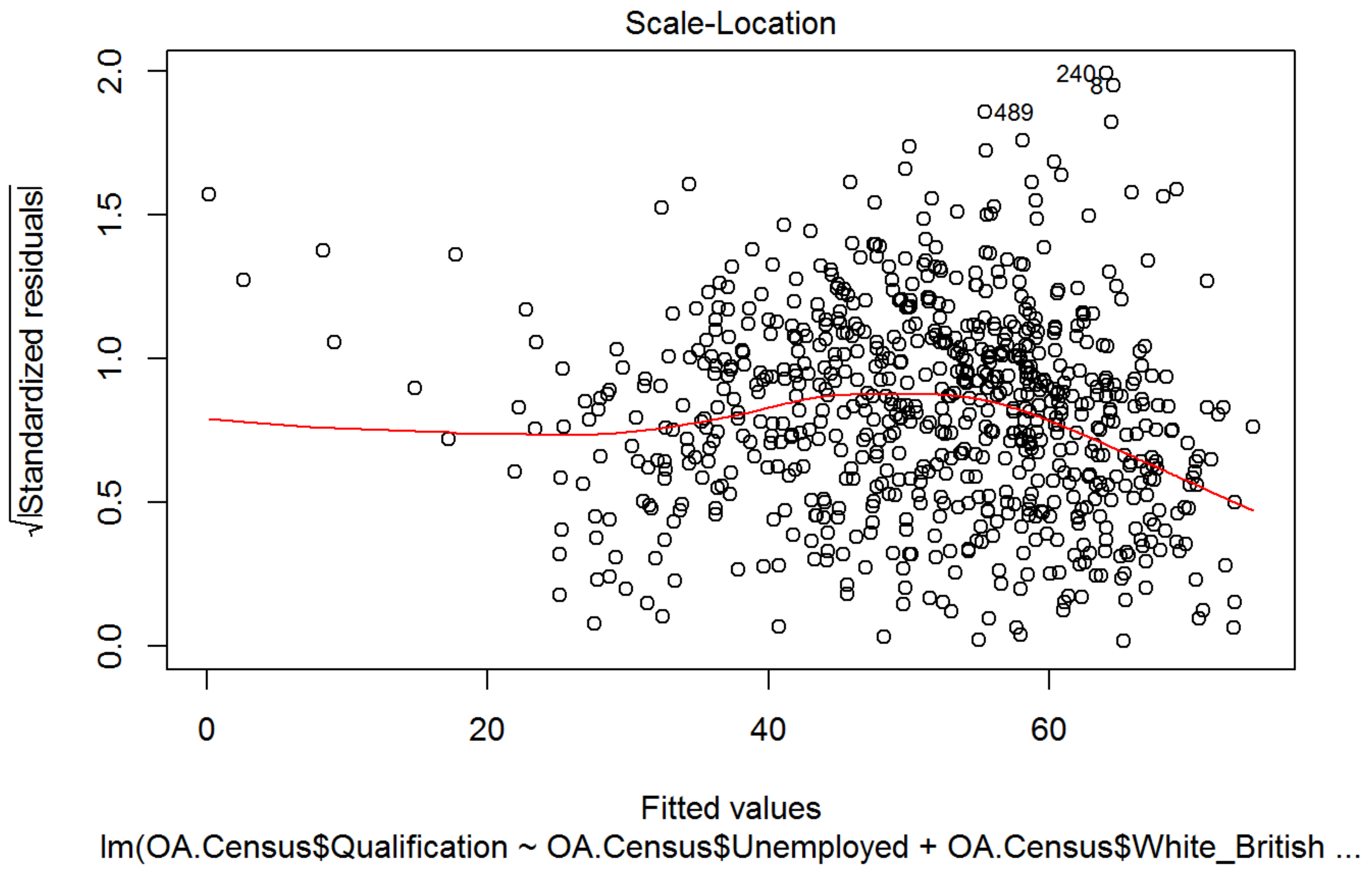
```
# this will plot 4 scatter plots from the linear model
plot(model)
```

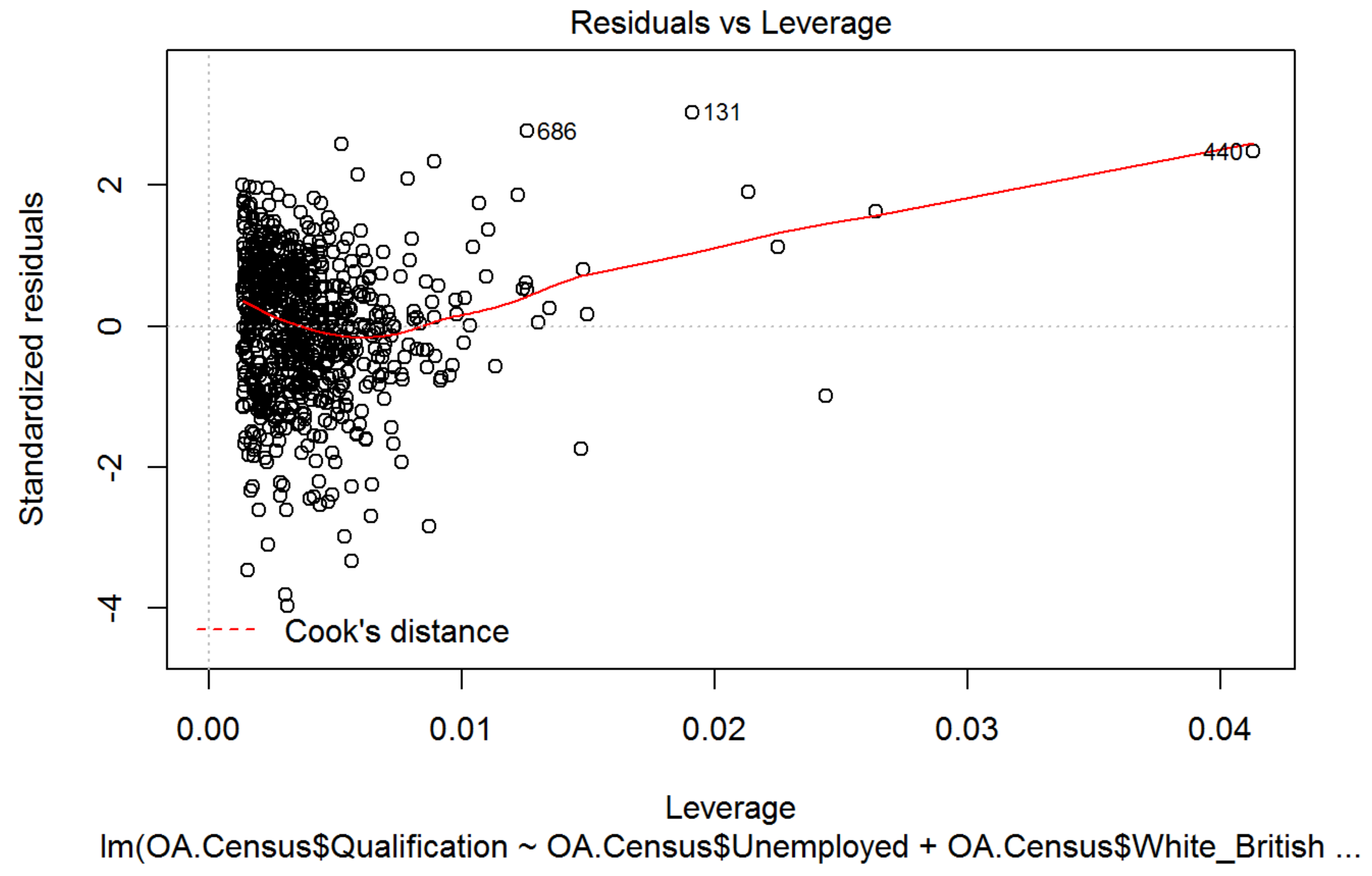


Residuals vs Fitted
Residuals
40
20
0
-20
-40
-60
0
20
40
60
489
240
8
Fitted values
lm(OA.Census$Qualification ~ OA.Census$Unemployed + OA.Census$White_British ...

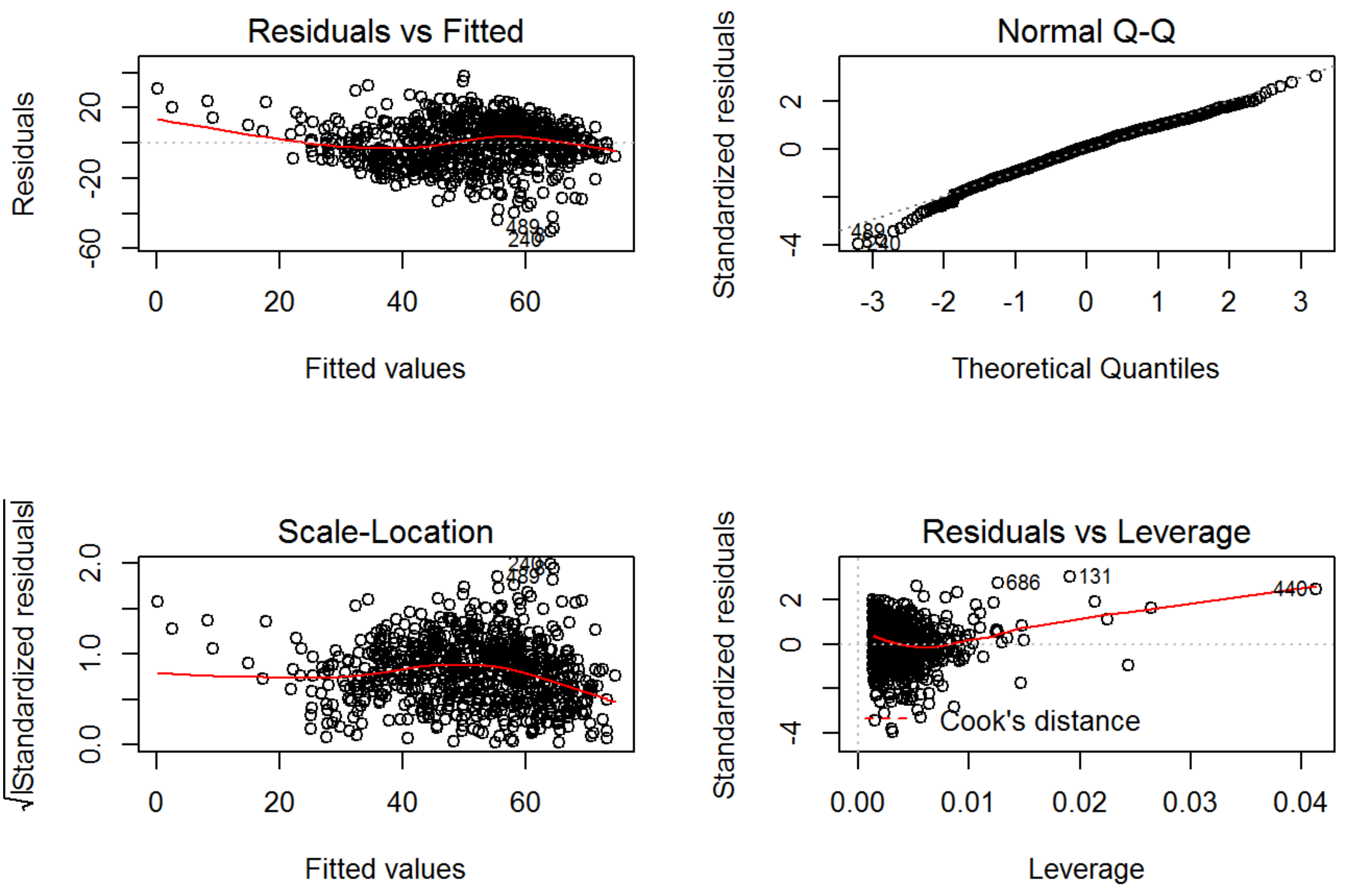


Normal Q-Q
Standardized residuals
2
0
-2
-4
-3
-2
-1
0
1
2
3
489
8
240
Theoretical Quantiles
lm(OA.Census$Qualification ~ OA.Census$Unemployed + OA.Census$White_British ...


Scale-Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
240
8
489
0
20
40
60
Fitted values
lm(OA.Census$Qualification ~ OA.Census$Unemployed + OA.Census$White_British ...


Residuals vs Leverage
Standardized residuals
-4
-2
0
2
686
131
440
Cook's distance
0.00
0.01
0.02
0.03
0.04
Leverage
lm(OA.Census$Qualification ~ OA.Census$Unemployed + OA.Census$White_British ...

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

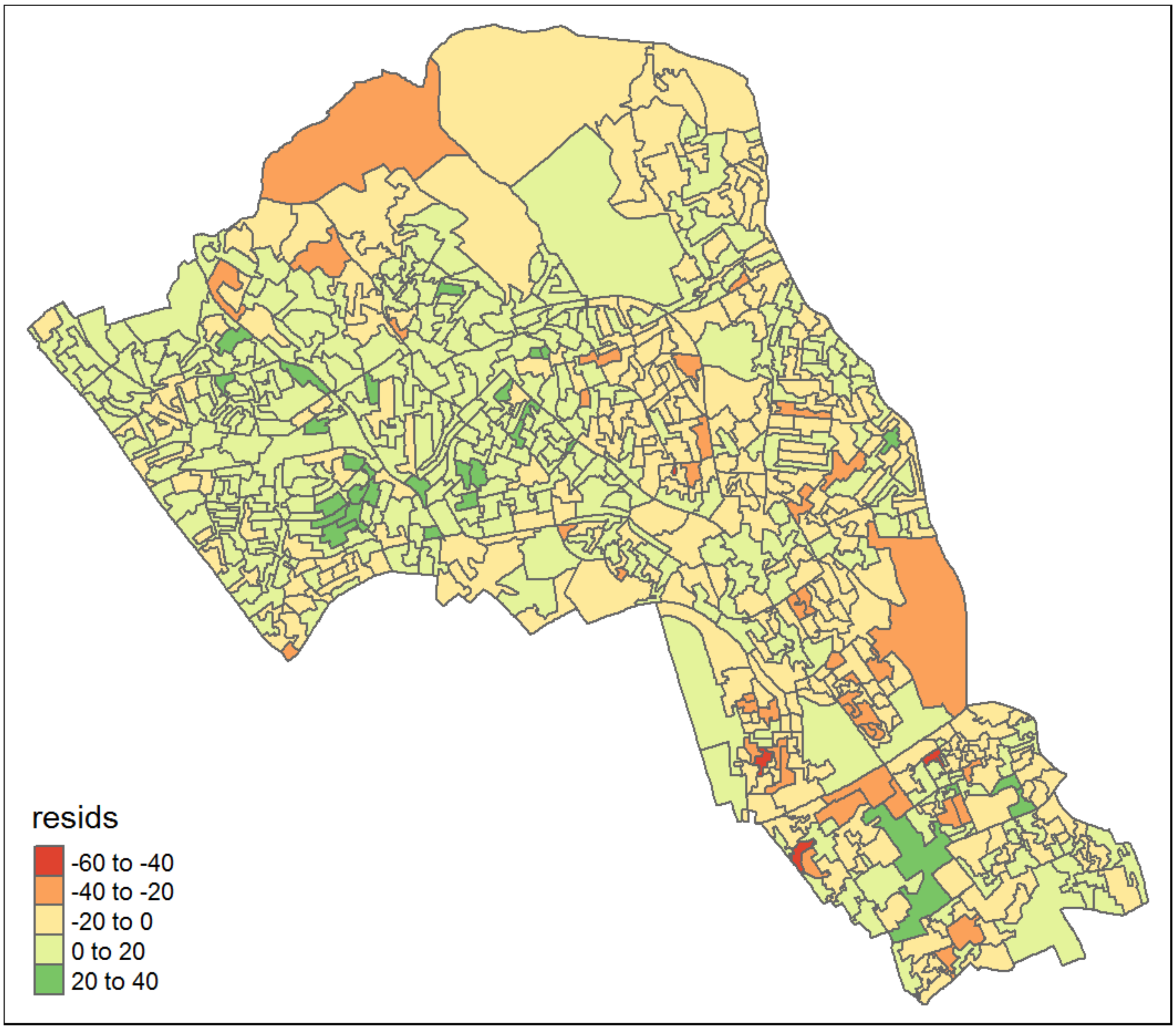


If you want to print just one of the plots you can enter *which = n* within the `plot()` function. i.e. `plot(model, which = 3)`

Mapping the residuals

We can also 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")
```



If you notice a geographic pattern to the residuals, it is possible that an unobserved variable may also be influencing our dependent variable in the model (% with a qualification).

Geographically Weighted Regression (GWR)

> GWR is the term introduced by Fotheringham, Charlton and Brunsdon (1997, 2002) to describe a family of regression models in which the coefficients are allowed to vary spatially. GWR uses the coordinates of each sample point or zone centroid, ti, as a target point for a form of spatially weighted least squares regression (for some models the target points can be separately defined, e.g. as grid intersection points, rather than observed data points). (de Smith *et al*, 2015) (http://www.spatialanalysisonline.com/HTML/?geographically_weighted_regres.htm)

Prior to running the GWR model we need to calculate a kernel bandwidth. This will determine now the GWR subsets the data when its test multiple models across space.

```
library("spgwr")

#calculate kernel bandwidth
GWRbandwidth <- gwr.sel(OA.Census$Qualification ~ OA.Census$Unemployed+OA.Census$W
hite_British, data=OA.Census,adapt=T)
```

```
## 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
```

Next, we can run the model and view the results.

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

#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.            11.0800 34.4300 45.7700 59.7500 85.0200 47.8670
## OA.Census.Unemployed    -5.4530 -3.2830 -2.5540 -1.7940  0.7702 -3.2946
## OA.Census.White_British -0.2805  0.1995  0.3779  0.5322  0.9468  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
```

Upon first glance, much of the outputs of this model are identical to the outputs of the linear model. However, we can explore the coefficients of this model across each area unit.

We create a results output from the model which contains a number of attributes which correspond with each unique output area from our OA.Census file. We have printed the names of each of the new variables in the example below. They include a local R2 value, the predicted values (for % qualifications) and local coefficients for each variable. We will then bind the outputs to our OA.Census polygon so we can map them.

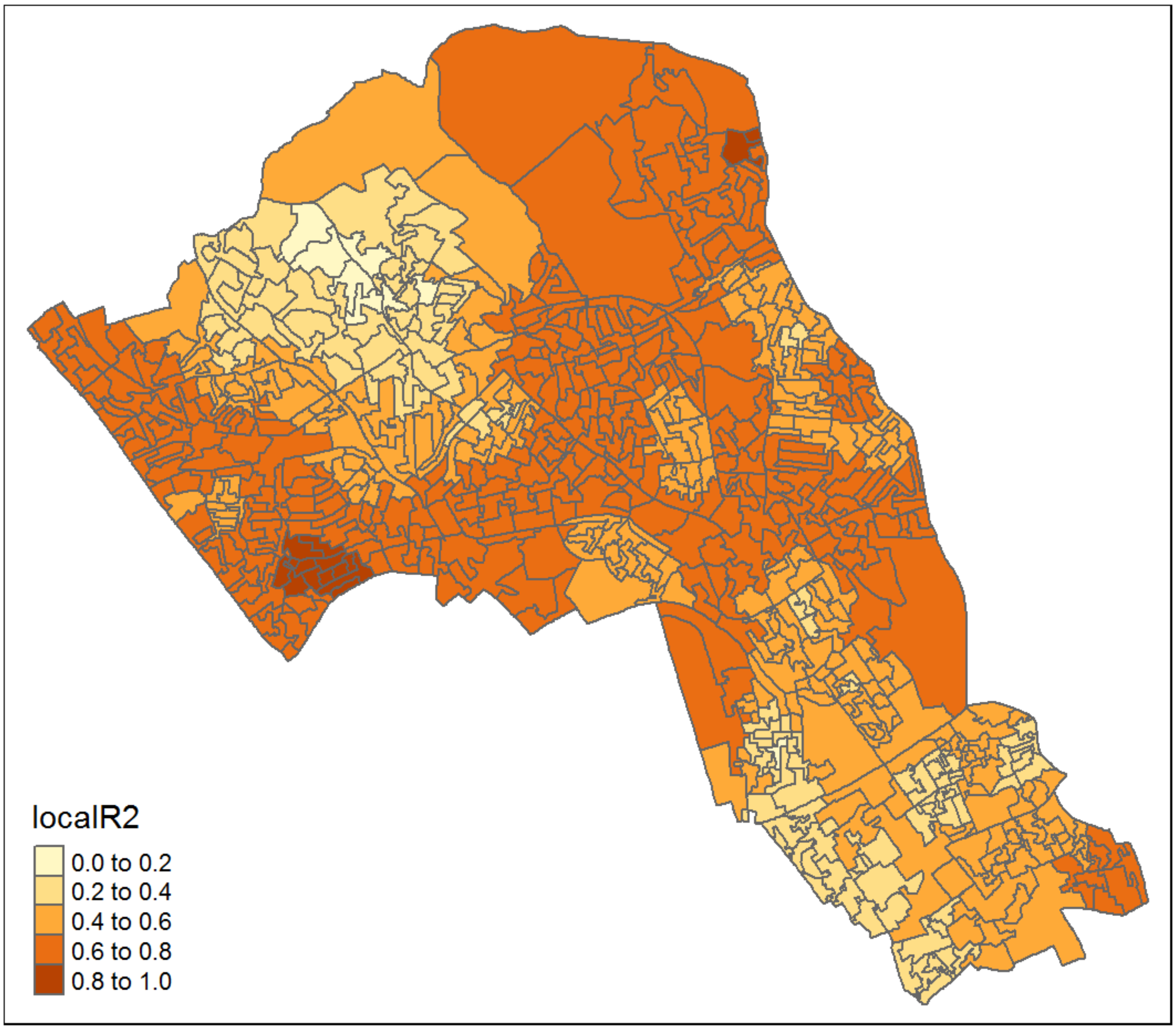
```
results <-as.data.frame(gwr.model$SDF)

names(results)
```

```
##  [1] "sum.w"                          "X.Intercept."
##  [3] "OA.Census.Unemployed"           "OA.Census.White_British"
##  [5] "X.Intercept._se"                "OA.Census.Unemployed_se"
##  [7] "OA.Census.White_British_se"     "gwr.e"
##  [9] "pred"                           "pred.se"
## [11] "localR2"                        "X.Intercept._se_EDF"
## [13] "OA.Census.Unemployed_se_EDF"    "OA.Census.White_British_se_EDF"
## [15] "pred.se_EDF"
```

```
gwr.map <- cbind(OA.Census, as.matrix(results))
```

The variable names followed by the name of our original data frame (i.e. OA.Census.Unemployed) are the coefficients of the model.

```
qtm(gwr.map, fill = "localR2")
```



Using gridExtra

We will now consider some of the other outputs. We will create four maps in one image to show the original distributions of our unemployed and White British variables and their coefficients in the GWR model.

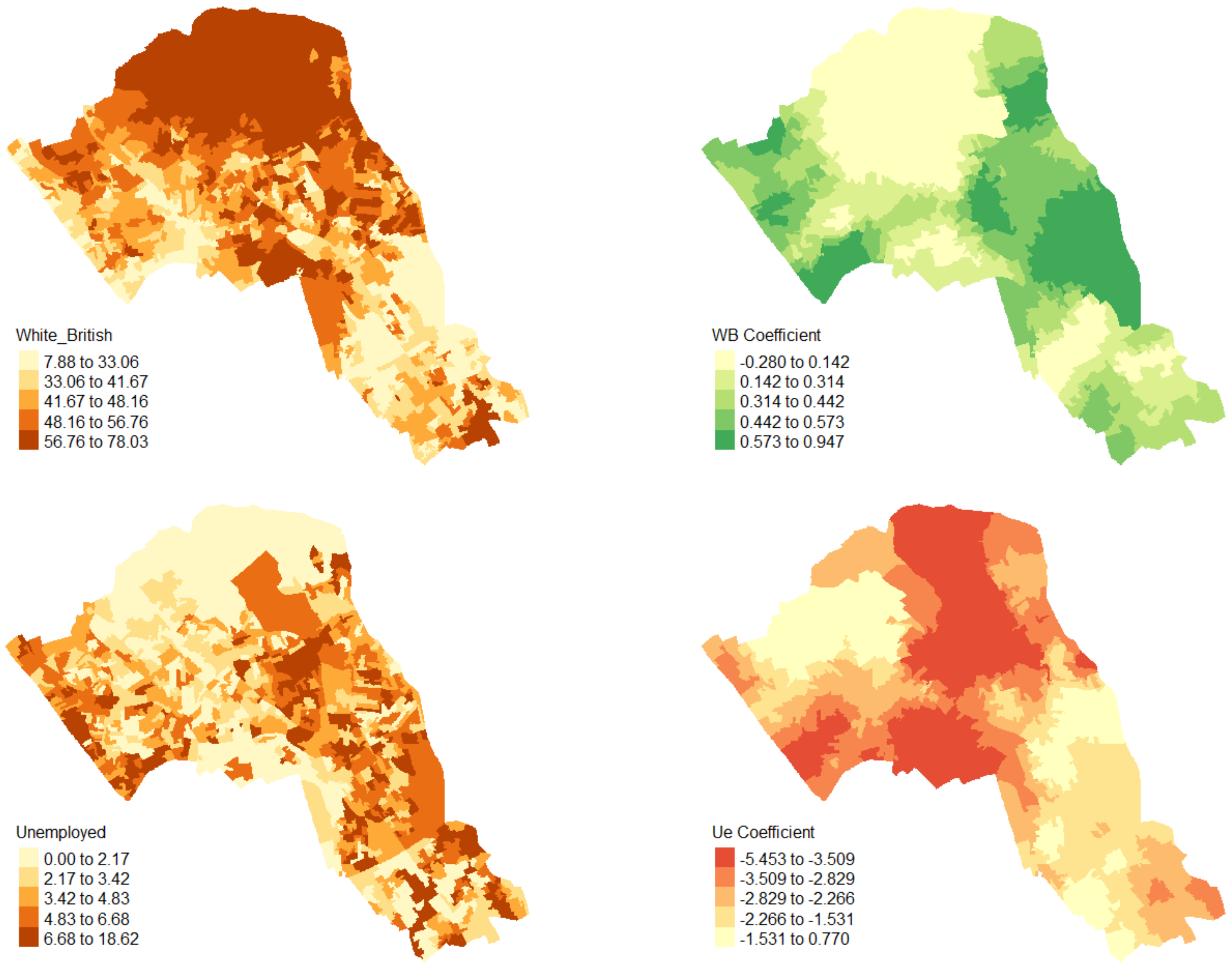
To facet four maps in tmap we can use functions from the grid and gridExtra packages which allow us to split the output window into segments. We will divide the output into four and print a map in each window.

Firstly, we will create four map objects using tmap. Instead of printing them directly as we have done usually, we all assign each map an object ID so it can be called later.

```
# create tmap objects
map1 <- tm_shape(gwr.map) + tm_fill("White_British", n = 5, style = "quantile")  +
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 = "qua
ntile", 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") + 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 = "quanti
le", title = "Ue Coefficient") + tm_layout(frame = FALSE, legend.text.size = 0.5,
legend.title.size = 0.6)
```

With the four maps ready to be printed, we will now create a grid to print them into. From now on everytime we wish to recreate the maps we will need to run the grid.newpage() function to clear the existing grid window.

```r
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))
```



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)