MAGEO 06 41 - part 3 - Co-Variations in Space + Moran

Geoffrey Caruso (geoffrey.caruso@uni.lu)

Version 16/11/2020

R lab: OLS regression and residual spatial autocorrelation

Loading data from Luxembourg municipalities

We start by opening attribute data stored within a spatial data frame, a dataframe that includes also geographical information (imported earlier from a shapefile and saved as a Rdata set). The attribute data contains some variables we can use for regression in addition to geographic descriptors (just like the dbf associated with a shp can contain many fields)

```
setwd("/Users/geoffrey.caruso/Dropbox/GEOF_TEACHING/MAGEO/MAGEO_06_41/MAGEO06_41_Rlab_OLS_MORAN")
wd<-getwd()
load("Lux116sdf_unemp.RData")
ls()
```

```
## [1] "Lux116sdf_unemp" "wd"
```

The loaded object belongs to spatial class and necessitates the sp package

```
class(Lux116sdf_unemp)
```

```
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
library(sp)
library(sf)# now required for use later with spded
```

```
## Linking to GEOS 3.8.1, GDAL 3.2.1, PROJ 7.2.1
```

The structure of spatial objects will be detailed later. For now we are interested to access the attribute table. It is placed within a specific slot, data, within the spatial data frame. The geographic attributes are used for mapping purposes (this is rather the object of the second course)

We explore the data available.

```
head(Lux116sdf_unemp@data)
```

```
##   cat    COMMUNE CODES_116 IDCANT_116   CANT_116 IDDIS   DISTRICT_1 IDPAYS
## 0   1 Bascharage    110101          1   Capellen     1   Luxembourg      1
## 1   2     Tandel    120804          8    Vianden     2     Diekirch      1
## 2   3   Beaufort    131001         10 Echternach     3 Grevenmacher      1
## 3   4       Bech    131002         10 Echternach     3 Grevenmacher      1
## 4   5  Beckerich    120701          7    Redange     2     Diekirch      1
## 5   6    Berdorf    131003         10 Echternach     3 Grevenmacher      1
##         PAYS UnemRate_08 RoadLuxKm PopDensity_2011 PC_Foreign
## 0 Luxembourg        4.15      26.7          391.59    27.8719
## 1 Luxembourg        7.08      36.2          164.05    43.1677
## 2 Luxembourg        2.52      27.4           46.25    23.5622
## 3 Luxembourg        3.96      42.8           80.71    27.2133
## 4 Luxembourg        7.47      36.2           77.34    40.4481
## 5 Luxembourg        4.15       7.6          357.27    47.8513
```

```
plot(Lux116sdf_unemp)
```

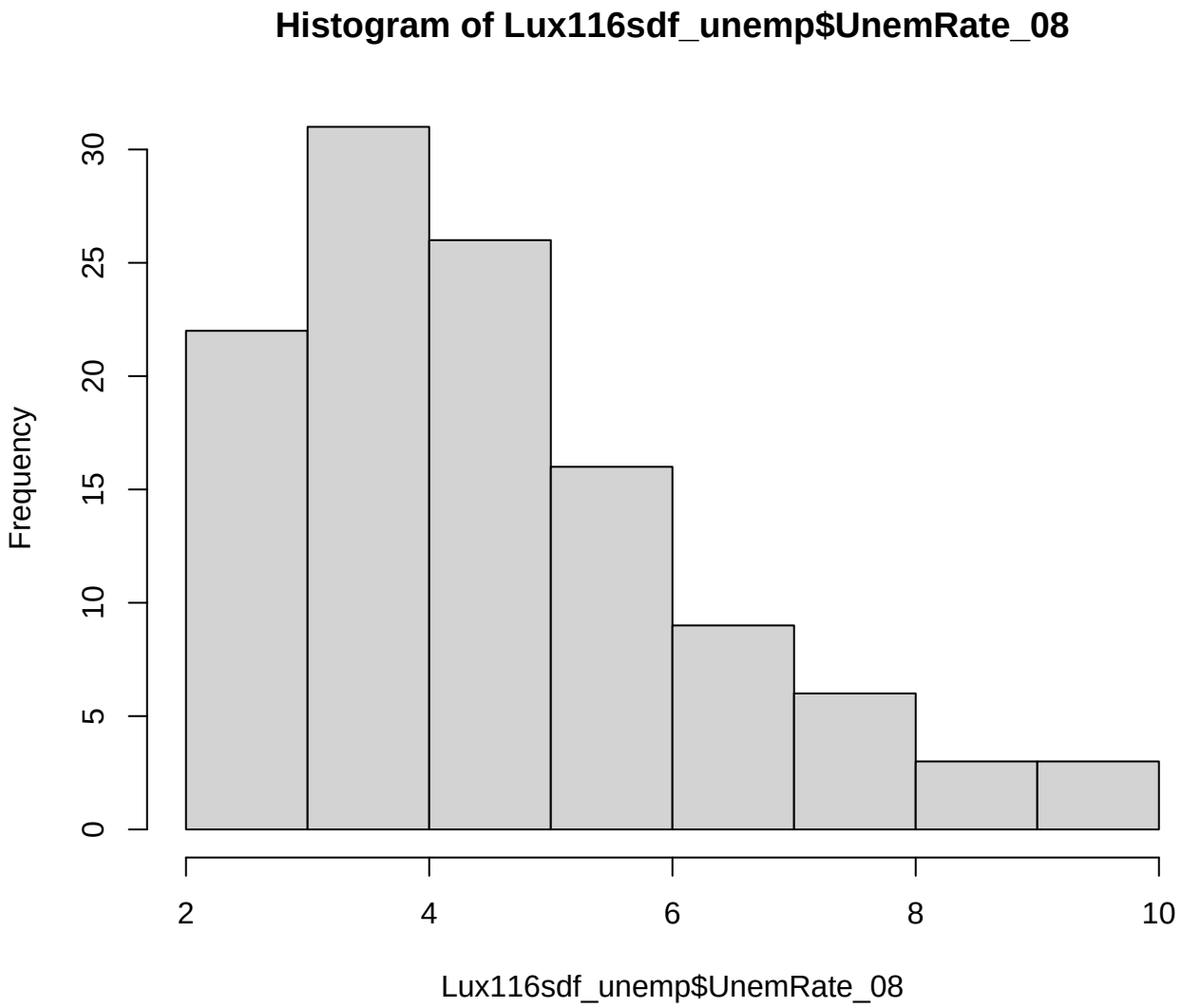
The last 4 columns contain data of interest. We will focus on Unemployment rate, which will be our dependent variable

```
summary(Lux116sdf_unemp)
```

```
## Object of class SpatialPolygonsDataFrame
## Coordinates:
##        min      max
## x 48921.43 106106.7
## y 57005.46 138764.8
## Is projected: NA
## proj4string : [NA]
## Data attributes:
##       cat               COMMUNE      CODES_116        IDCANT_116
##  Min.   :  1.00   Bascharage:  1   Min.   :110101   Min.   : 1.000
##  1st Qu.: 29.75   Beaufort  :  1   1st Qu.:110305   1st Qu.: 3.000
##  Median : 58.50   Bech      :  1   Median :120604   Median : 6.000
##  Mean   : 58.50   Beckerich :  1   Mean   :118799   Mean   : 6.034
##  3rd Qu.: 87.25   Berdorf   :  1   3rd Qu.:120908   3rd Qu.: 9.000
##  Max.   :116.00   Berg      :  1   Max.   :131210   Max.   :12.000
##                   (Other)   :110
##        CANT_116      IDDIS              DISTRICT_1     IDPAYS          PAYS
##  Esch      :14   Min.   :1.000   Diekirch    :43   Min.   :1   Luxembourg:116
##  Diekirch  :12   1st Qu.:1.000   Grevenmacher:26   1st Qu.:1
##  Capellen  :11   Median :2.000   Luxembourg  :47   Median :1
##  Luxembourg:11   Mean   :1.819                     Mean   :1
##  Mersch    :11   3rd Qu.:2.000                     3rd Qu.:1
##  Redange   :10   Max.   :3.000                     Max.   :1
##  (Other)   :47
##   UnemRate_08      RoadLuxKm     PopDensity_2011     PC_Foreign
##  Min.   :2.070   Min.   : 1.70   Min.   :  27.76   Min.   :15.03
```

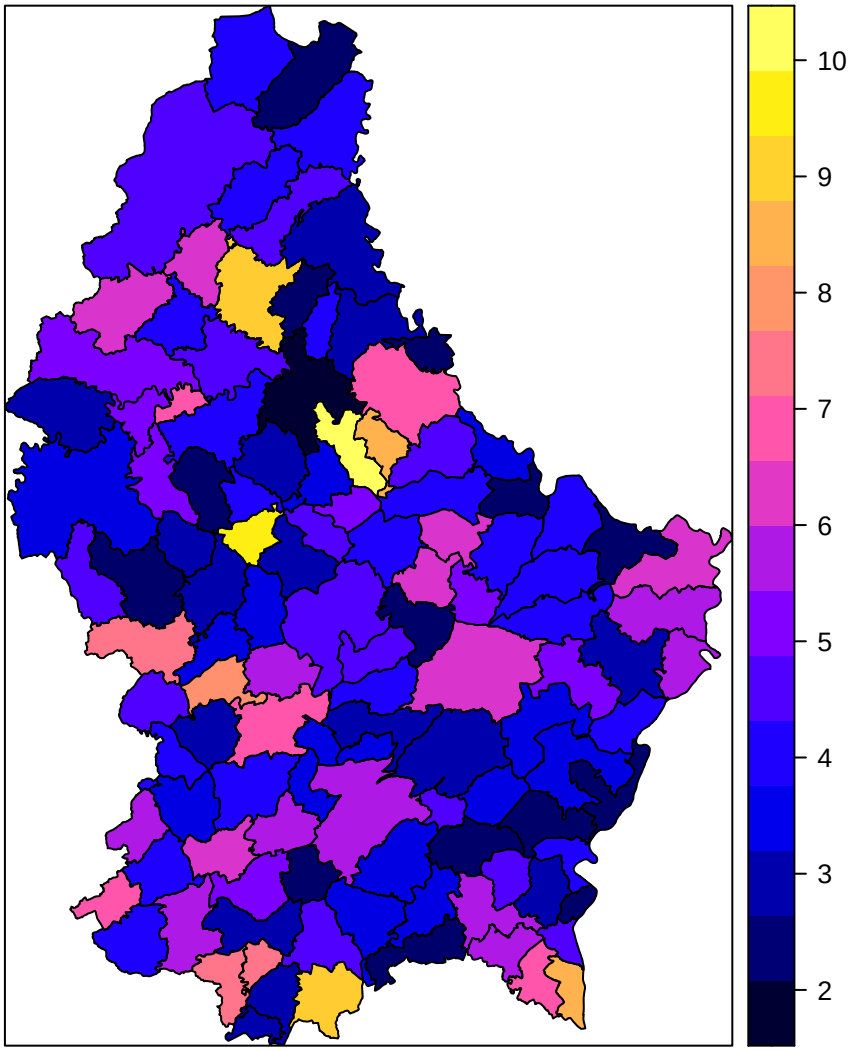
```
##  1st Qu.:3.178   1st Qu.:20.75   1st Qu.:  56.36   1st Qu.:25.21
##  Median :4.150   Median :29.95   Median : 110.48   Median :31.52
##  Mean   :4.522   Mean   :32.27   Mean   : 229.83   Mean   :32.82
##  3rd Qu.:5.567   3rd Qu.:41.58   3rd Qu.: 224.86   3rd Qu.:39.34
##  Max.   :9.920   Max.   :74.80   Max.   :2099.30   Max.   :64.88
##
```

```
hist(Lux116sdf_unemp$UnemRate_08)
```



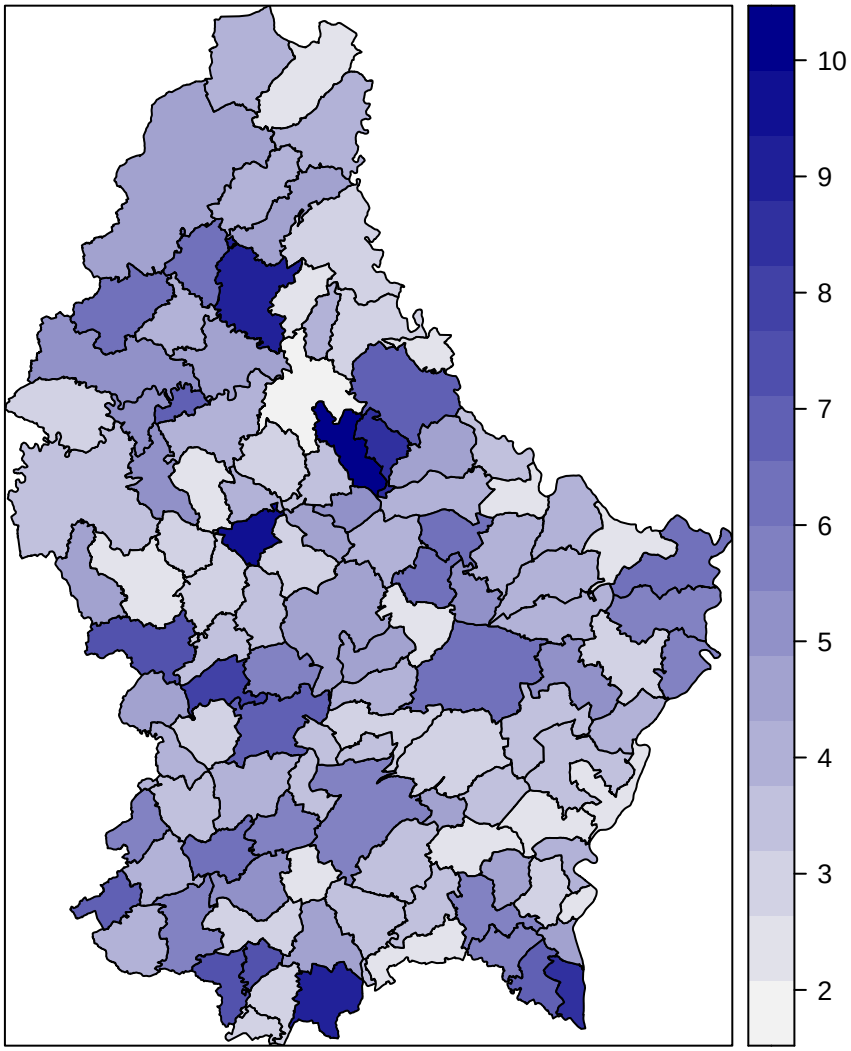
A default map is obtained with spplot, which of course needs fine tuning, adequate discretization, colours (reversed ramp,etc.) etc... but at least allows for a quick exploration of the distribution from within R

```
spplot(Lux116sdf_unemp, zcol="UnemRate_08")
```



For a quick map with a more reasonable coloring (for more fine tuning, see summer semester's course)

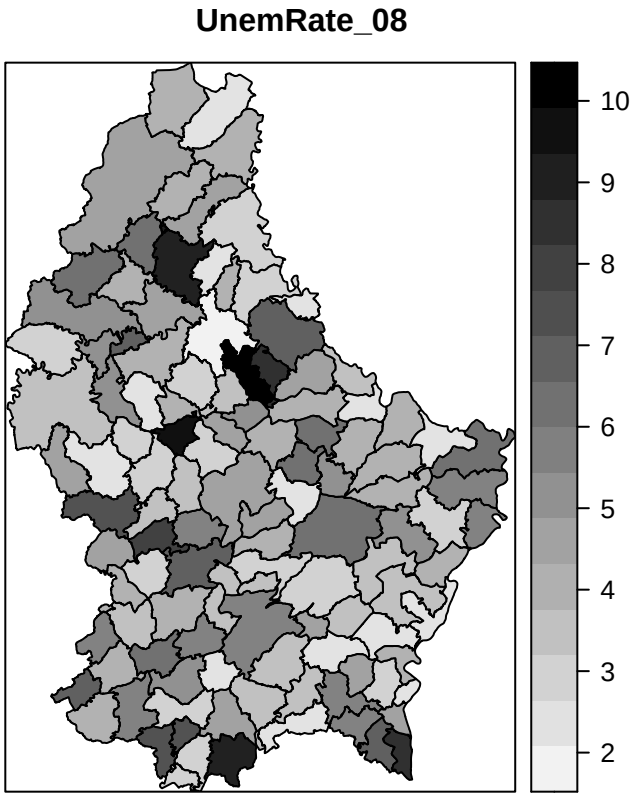
```
spplot(Lux116sdf_unemp, zcol="UnemRate_08", col.regions=colorRampPalette(c("grey95", "darkblue"))( 116 )
```



Let's implement this (plus title) as a function (quickdirtymap), so we can quickly do several maps later just by referring to variable's name and its spatial dataframe:

```
quickdirtymap<-function(varmap, sdf){spplot(sdf, zcol=varmap, col.regions=colorRampPalette(c("grey95", "

quickdirtymap("UnemRate_08", Lux116sdf_unemp)
```



Regression and diagnostic

We hypothesize that employment is related to 'urbanness' (using density), distance the main jobs (using road distance) and percentage of foreigners (although in Luxembourg foreigners amount to 40 pc of population)

```
Model1 <- lm(UnemRate_08 ~ RoadLuxKm+PopDensity_2011+PC_Foreign, data=Lux116sdf_unemp)
summary(Model1)
```

```
##
## Call:
## lm(formula = UnemRate_08 ~ RoadLuxKm + PopDensity_2011 + PC_Foreign,
##     data = Lux116sdf_unemp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7450 -0.7089 -0.0467  0.6583  3.9982
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -0.857891   0.582764  -1.472 0.143796
## RoadLuxKm        0.067080   0.007843   8.553 7.04e-14 ***
## PopDensity_2011  0.001735   0.000446   3.890 0.000171 ***
## PC_Foreign       0.085820   0.015072   5.694 1.01e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.257 on 112 degrees of freedom
## Multiple R-squared:  0.514,  Adjusted R-squared:  0.501
## F-statistic: 39.49 on 3 and 112 DF,  p-value: < 2.2e-16
```

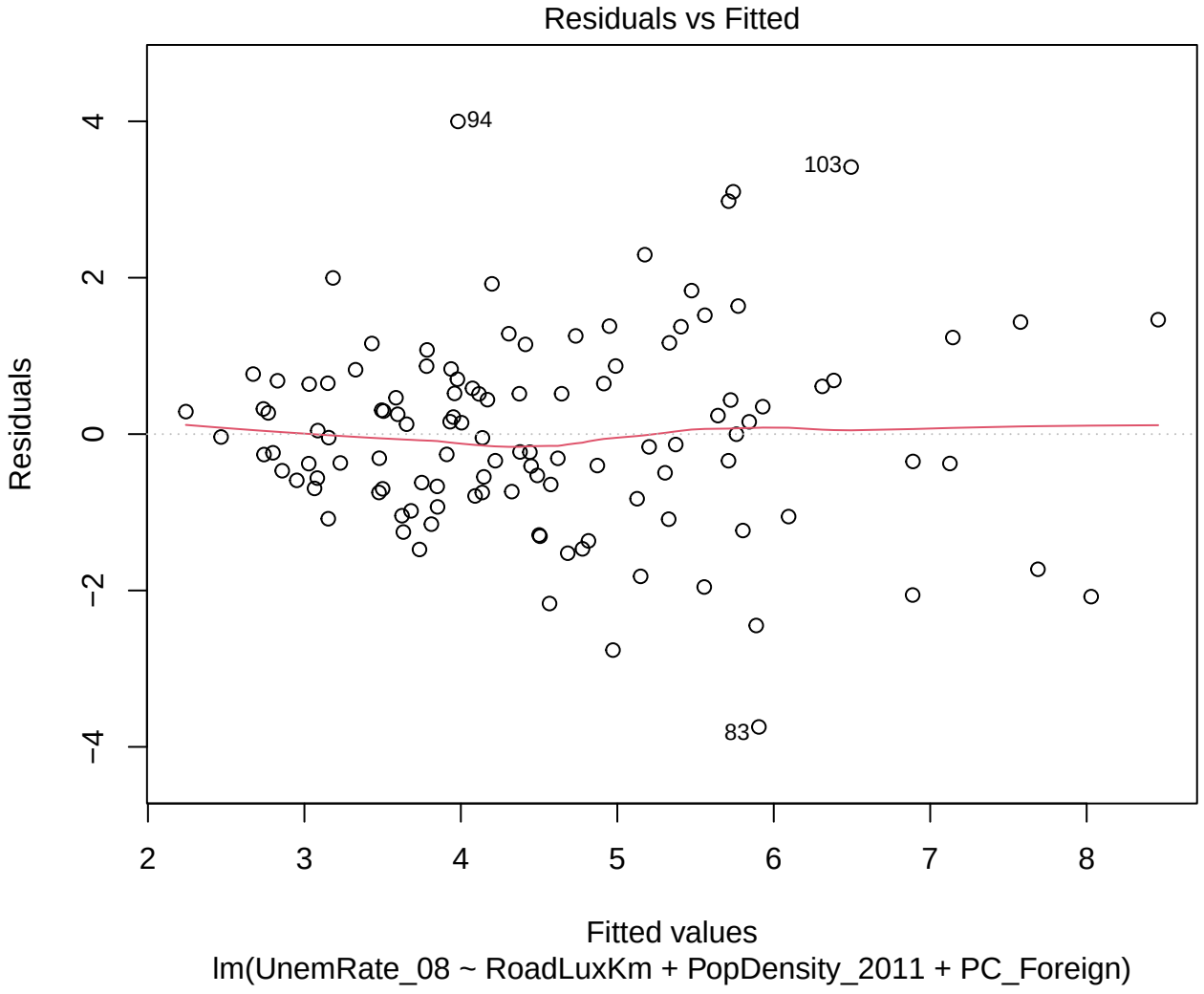
All effects hypothesized are significant and have expected sign

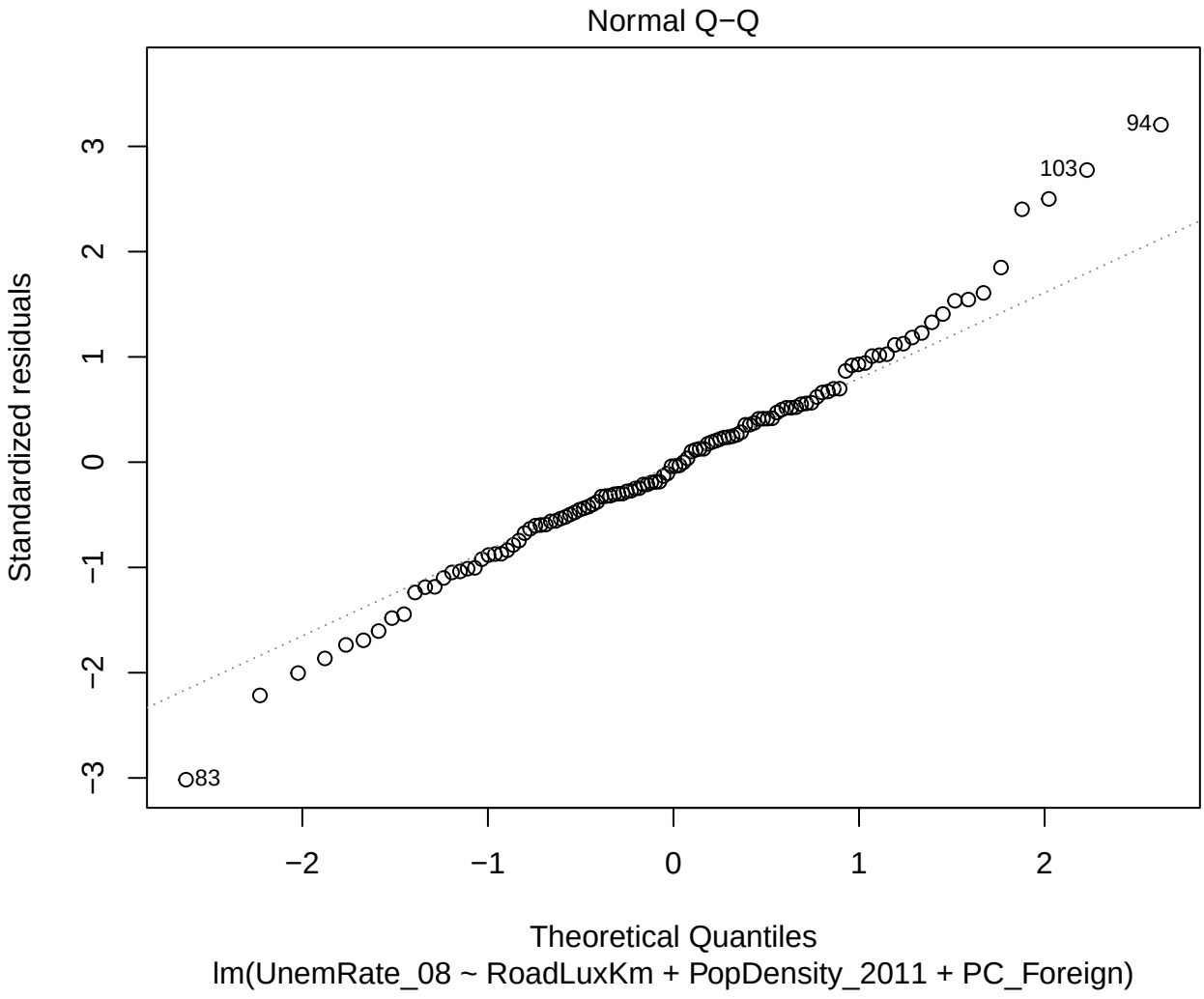
```
anova(Model1)
```

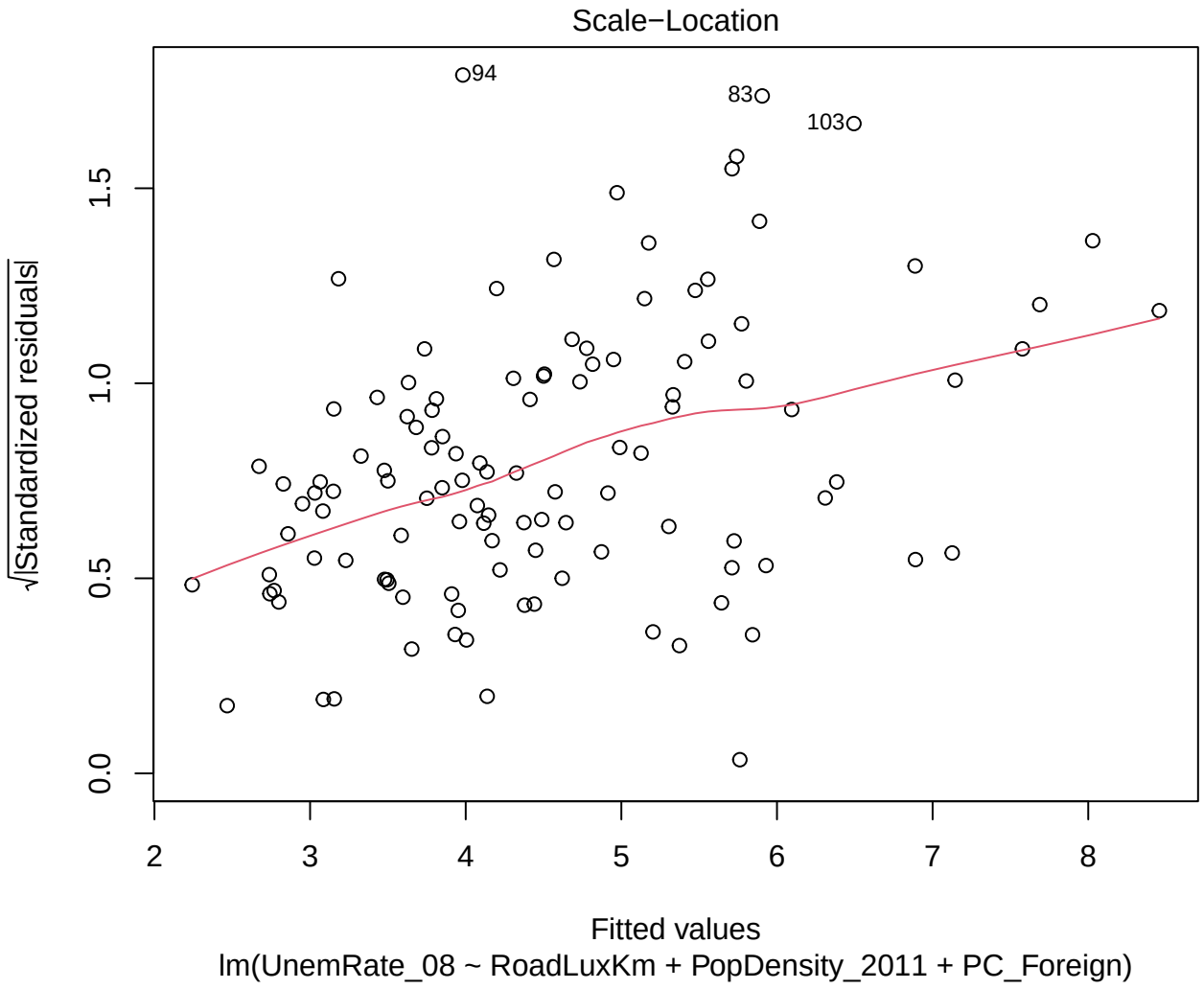
```
## Analysis of Variance Table
##
## Response: UnemRate_08
##                   Df  Sum Sq Mean Sq F value    Pr(>F)
## RoadLuxKm          1  36.400  36.400  23.040 4.940e-06 ***
## PopDensity_2011    1  99.551  99.551  63.014 1.721e-12 ***
## PC_Foreign         1  51.220  51.220  32.421 1.012e-07 ***
## Residuals        112 176.941   1.580
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

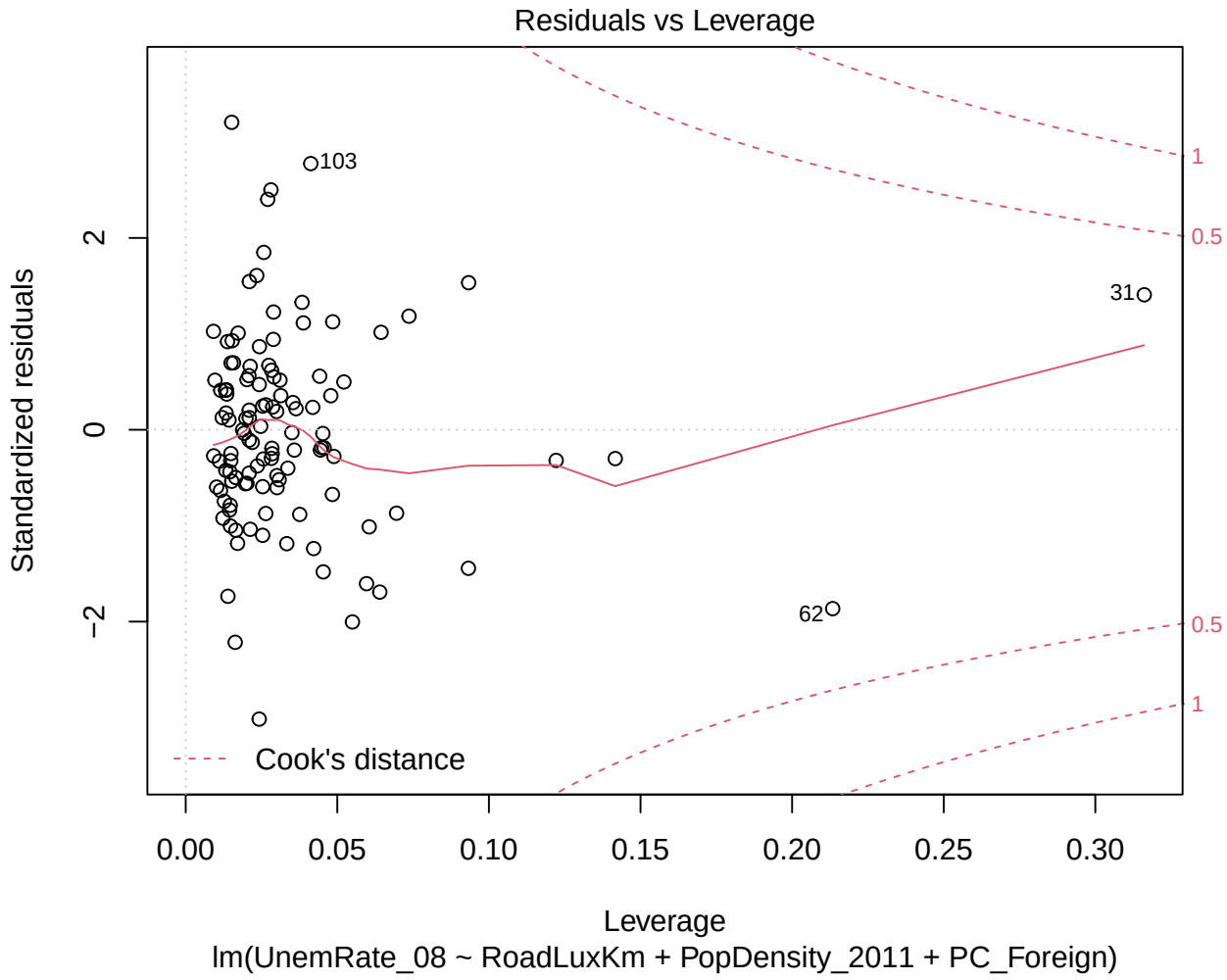
A part of the diagnostic can be obtained from

```
plot(Model1)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(UnemRate_08 ~ RoadLuxKm + PopDensity_2011 + PC_Foreign)
94
103
83
−3
−2
−1
0
1
2
3


Scale−Location
94
83
103
√|Standardized residuals|
0.0
0.5
1.0
1.5
2
3
4
5
6
7
8
Fitted values
lm(UnemRate_08 ~ RoadLuxKm + PopDensity_2011 + PC_Foreign)



Given these, we can suspect heteroscedasticity. We run a Breusch-Pagan test for measuring this:

```
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
bptest(Model1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Model1
## BP = 13.68, df = 3, p-value = 0.003375
```

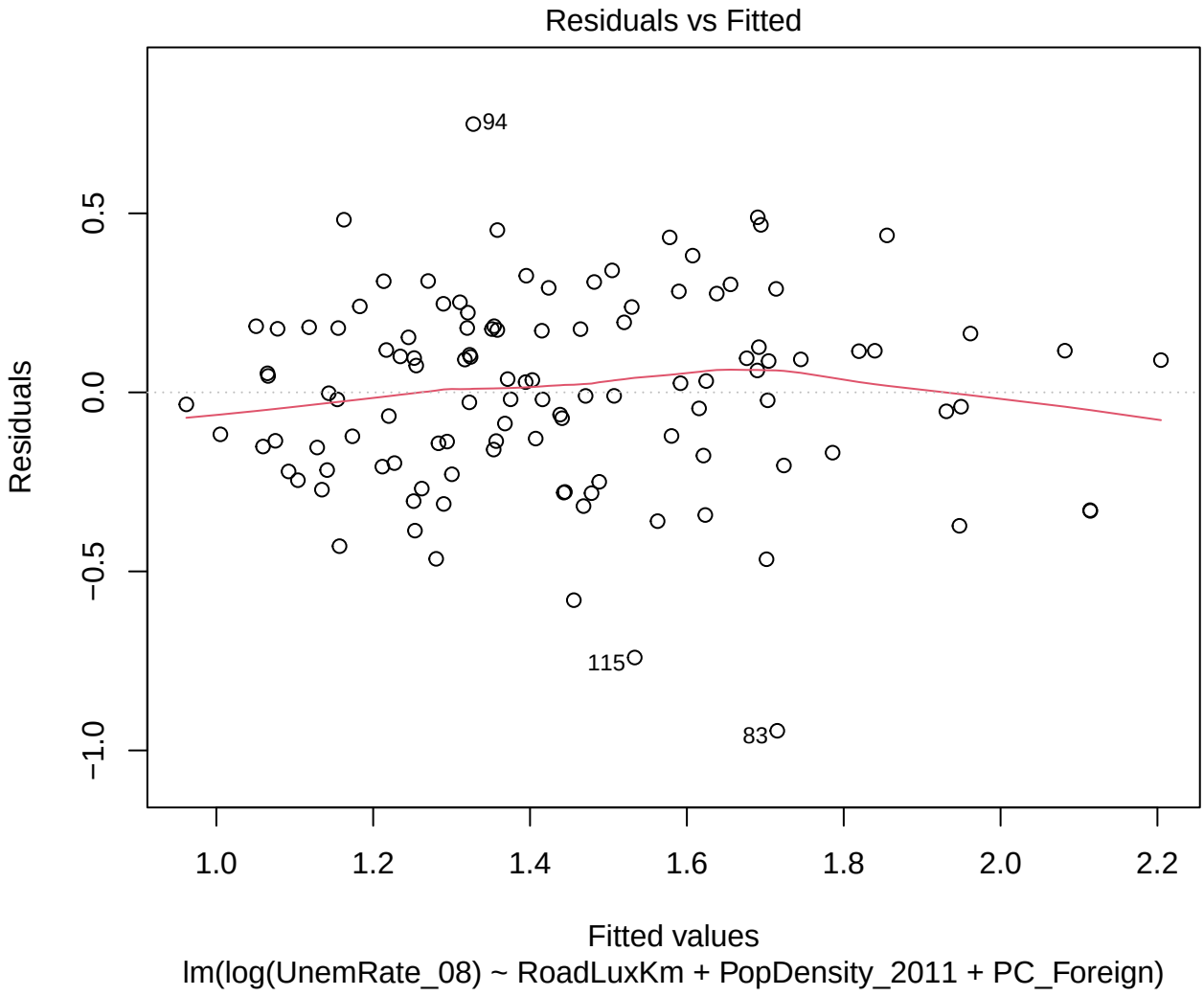
. . . and find hetereoscedasticity cannot be rejected.

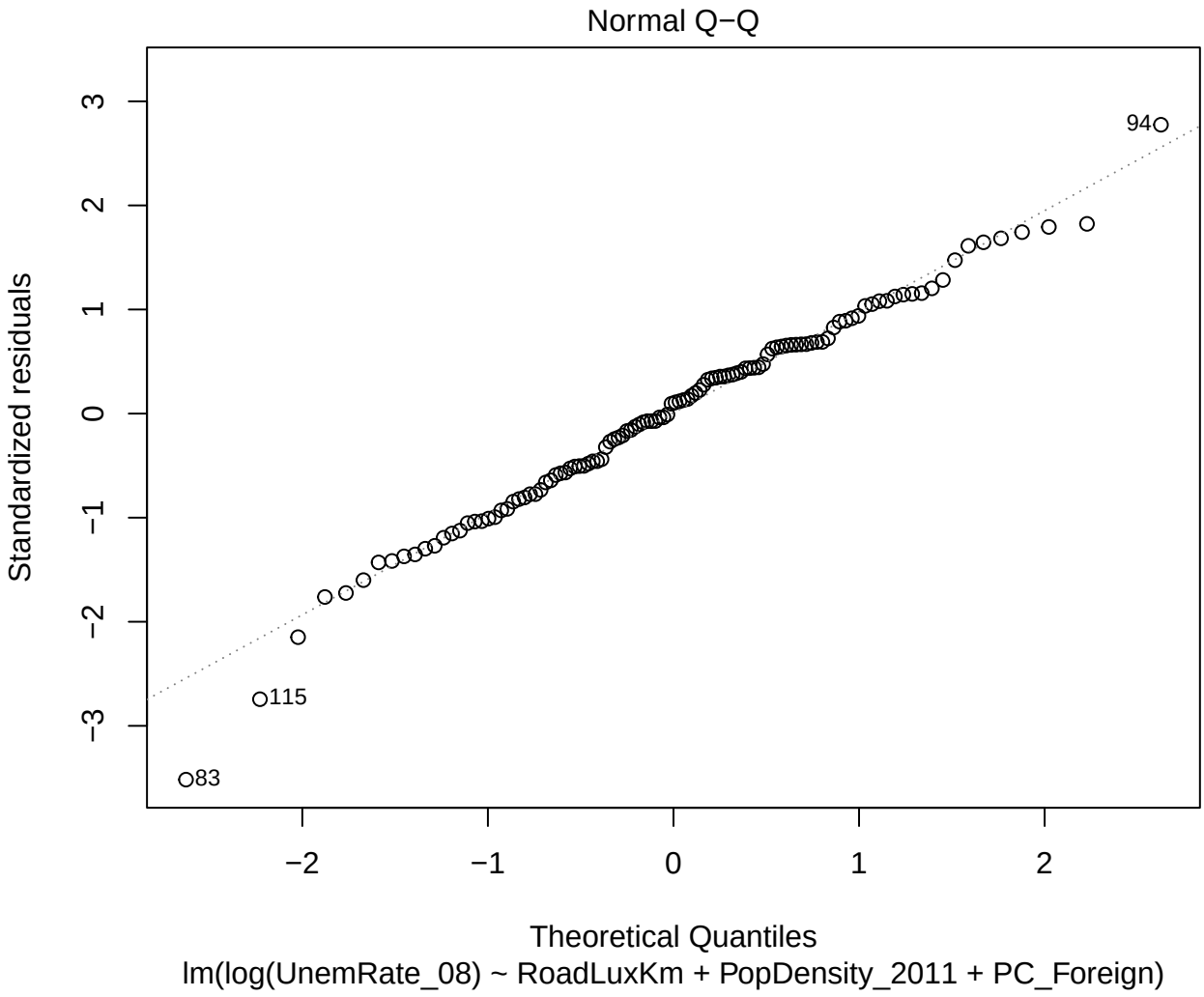
After log transform of the dependent variable, things go slightly better both graphically and according to Breusch-Pagan's test:

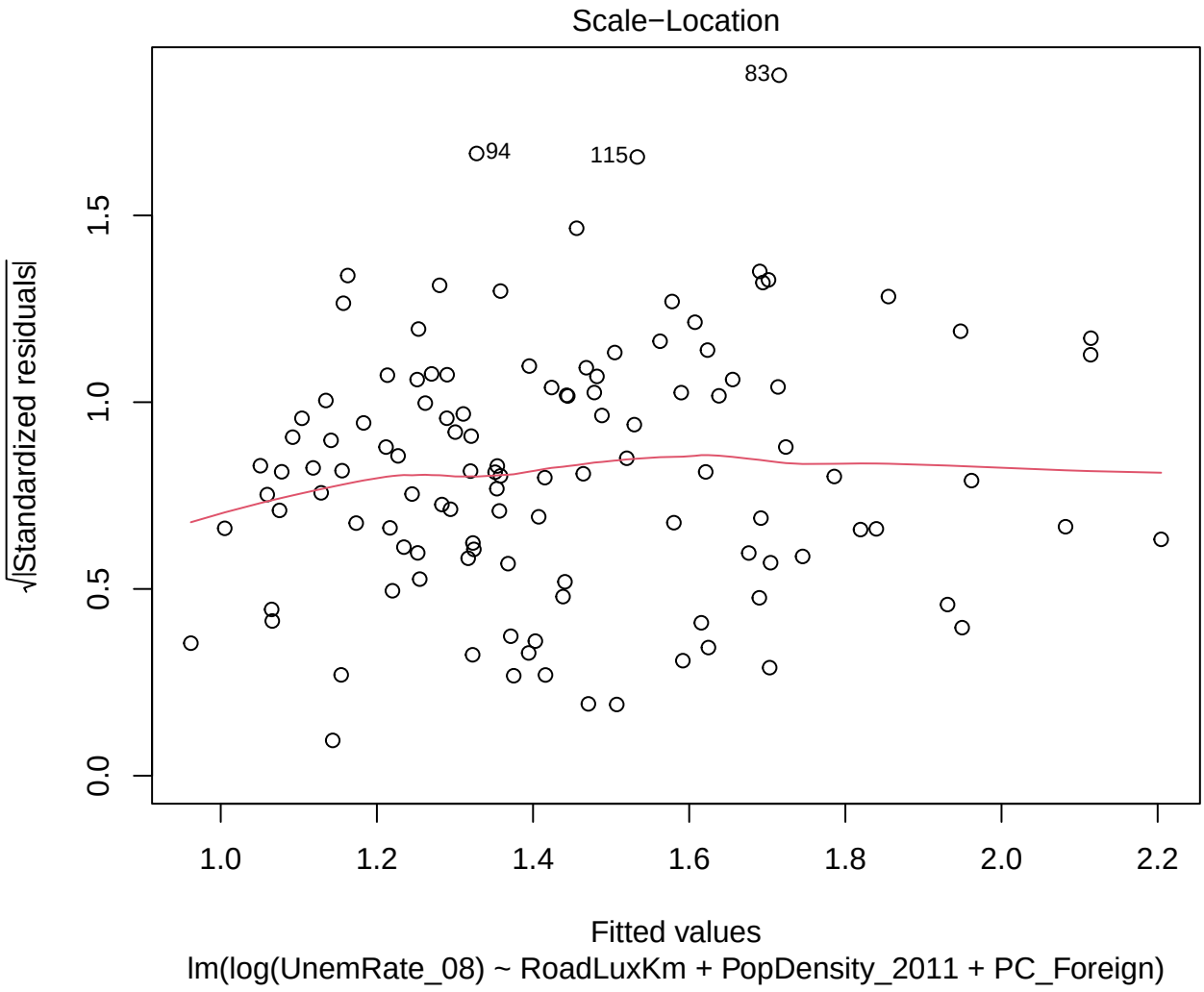
```r
Model2 <- lm(log(UnemRate_08) ~ RoadLuxKm+PopDensity_2011+PC_Foreign, data=Lux116sdf_unemp)
summary(Model2)
```

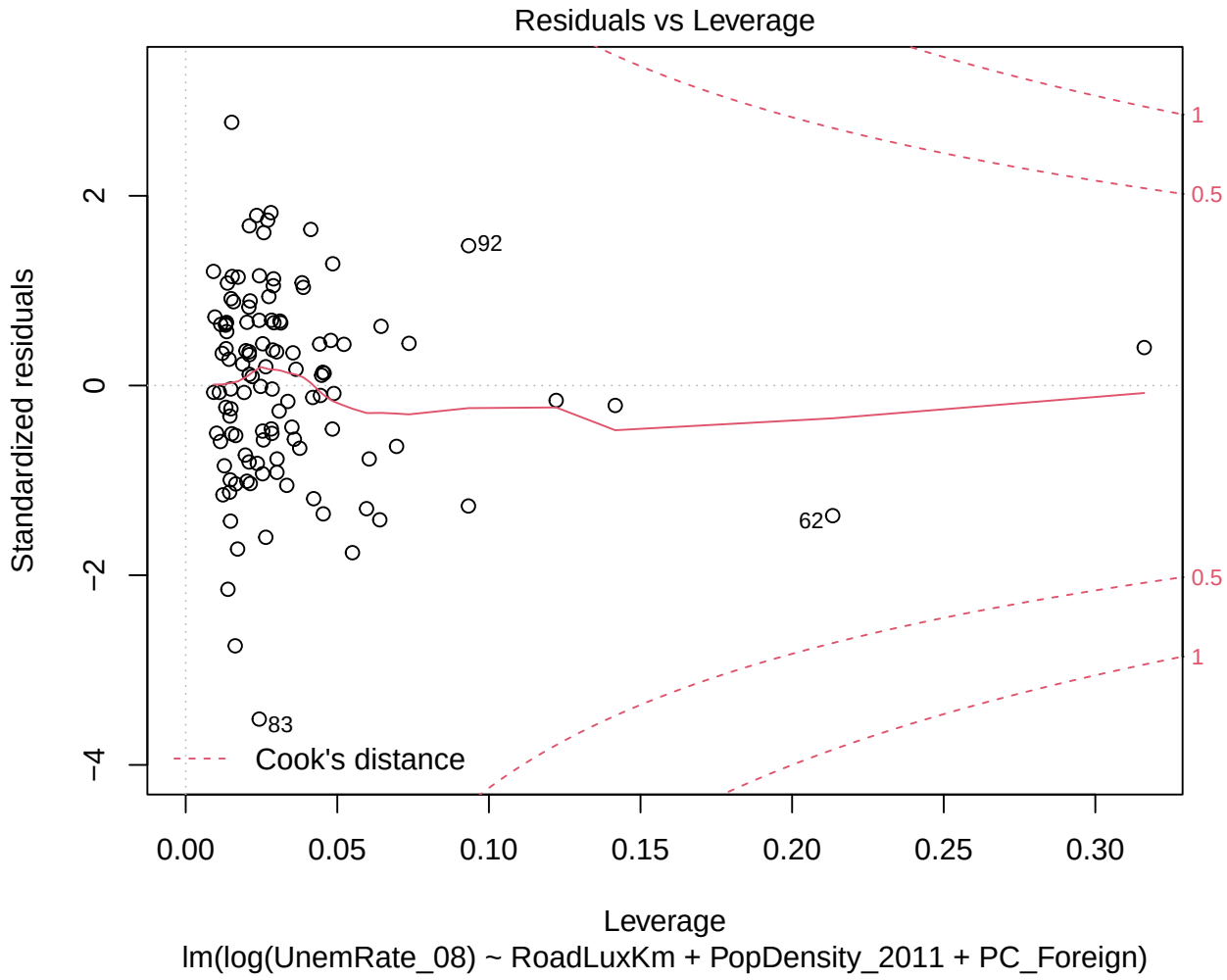
```
##
## Call:
## lm(formula = log(UnemRate_08) ~ RoadLuxKm + PopDensity_2011 +
##     PC_Foreign, data = Lux116sdf_unemp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.94515 -0.17045  0.02711  0.17803  0.74925
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.143e-01  1.261e-01   2.491 0.014198 *
## RoadLuxKm       1.432e-02  1.698e-03   8.436 1.30e-13 ***
## PopDensity_2011 3.368e-04  9.655e-05   3.488 0.000697 ***
## PC_Foreign      1.778e-02  3.263e-03   5.451 3.02e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2721 on 112 degrees of freedom
## Multiple R-squared:  0.4917, Adjusted R-squared:  0.4781
## F-statistic: 36.11 on 3 and 112 DF,  p-value: < 2.2e-16
```

```r
plot(Model2)
```


Residuals vs Fitted
94
115
83
Residuals
0.5
0.0
−0.5
−1.0
1.0
1.2
1.4
1.6
1.8
2.0
2.2
Fitted values
lm(log(UnemRate_08) ~ RoadLuxKm + PopDensity_2011 + PC_Foreign)


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(log(UnemRate_08) ~ RoadLuxKm + PopDensity_2011 + PC_Foreign)
94
115
83
3
2
1
0
−1
−2
−3
−2
−1
0
1
2


Scale−Location
83
94
115
√|Standardized residuals|
0.0
0.5
1.0
1.5
1.0
1.2
1.4
1.6
1.8
2.0
2.2
Fitted values
lm(log(UnemRate_08) ~ RoadLuxKm + PopDensity_2011 + PC_Foreign)

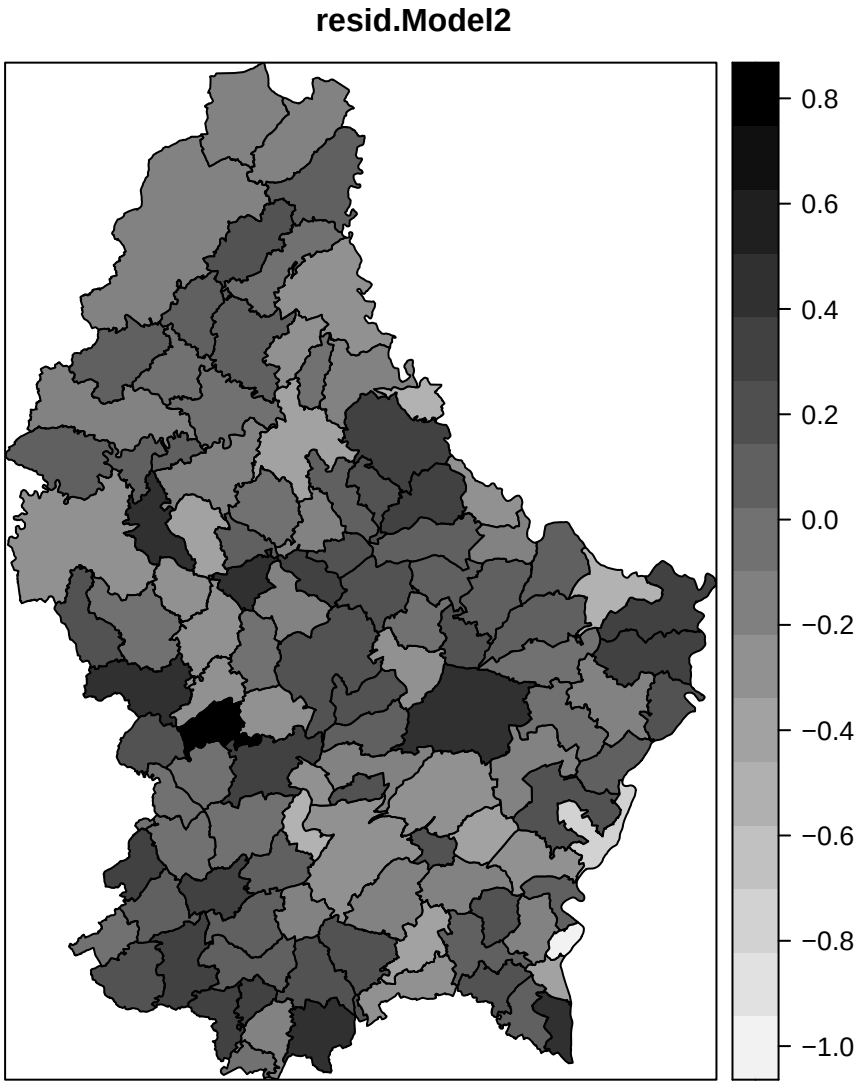



```
bptest(Model2)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Model2
## BP = 3.1527, df = 3, p-value = 0.3687
```

We can then append the residuals to the spatial data frame and map

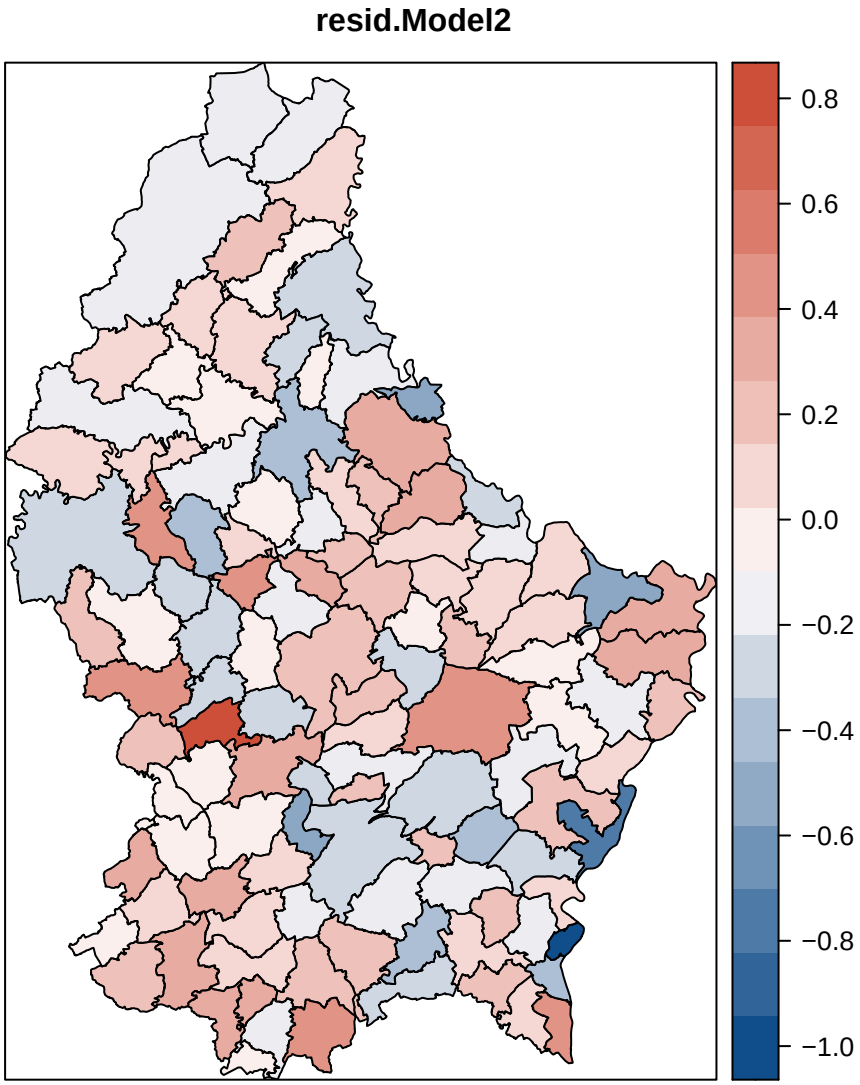
```
Lux116sdf_unemp$resid.Model2<-resid(Model2)
quickdirtymap("resid.Model2", Lux116sdf_unemp)
```



Obviously a 2 ways colour ramp would be more appropriate given that there are negative and positive values.
We adapt our quick mapping function:

```
quickdirtymap_B2R<-function(varmap, sdf){spplot(sdf, zcol=varmap, col.regions=colorRampPalette(c("dodge

quickdirtymap_B2R("resid.Model2", Lux116sdf_unemp)
```



Spatial autocorrelation of residuals

Neighbours and spatial weights For testing spatial autocorrelation, we need first defining neighbours and spatial weights (see theory class).

Spatial weights and spatial autocorrelation analysis tools are found in the spatial dependence package: spdep

Creating a spatial weight matrix is a two stage process: defining a neighbours list object (nb) then converting it into a weights list object (using nb2listw function).

The nb ojects is a list containing for each record a vector of the index of its neighbours. It can for example be based on distance-based information or on topological information retrieved from the original shapefile

Let's use contiguity information first and then a set of distance-based neighbours using the k nearest neighbours from polygon centroid.

```
library(spdep)
```

```
## 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')`
```

```
#Neighbours list created from contiguous polygons
nb_Lux_contig<-poly2nb(Lux116sdf_unemp)
summary(nb_Lux_contig)
```

```
## Neighbour list object:
## Number of regions: 116
## Number of nonzero links: 604
## Percentage nonzero weights: 4.488704
## Average number of links: 5.206897
## Link number distribution:
##
##  2  3  4  5  6  7  8  9 10
##  5 12 20 30 28 14  3  2  2
## 5 least connected regions:
## 76 82 87 99 102 with 2 links
## 2 most connected regions:
## 15 62 with 10 links
```

```
#Neighbours list created k nearest neighbours
coords_Lux <- coordinates(Lux116sdf_unemp)
knn_Lux4<- knearneigh(coords_Lux, k=4)
nb_Lux_k4<-knn2nb(knn_Lux4)
knn_Lux10<- knearneigh(coords_Lux, k=10)
nb_Lux_k10<-knn2nb(knn_Lux10)
```

The different methods holds different set of neihbours as one can see from mapping those lists in turn. Impact on the sptaial utocorrelation measured must be handled with care

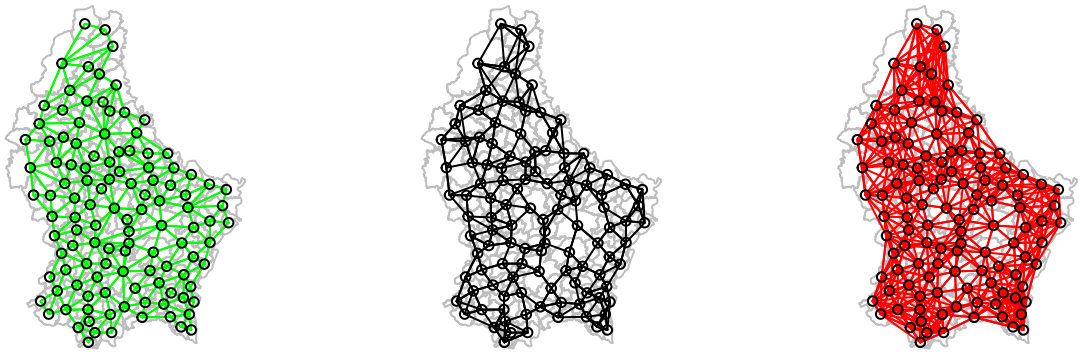
```
par(mfrow=c(1,3)) #places next 3 plots in a row

plot(Lux116sdf_unemp, border="grey")
title(main = "Neighbours list 1st order contiguity")
plot(nb_Lux_contig,coords_Lux, col="green",add=TRUE)

plot(Lux116sdf_unemp, border="grey")
title(main = "Neighbours list knn=4")
plot(nb_Lux_k4, coords_Lux, add=TRUE)

plot(Lux116sdf_unemp, border="grey")
title(main = "Neighbours list knn=10")
plot(nb_Lux_k10, coords_Lux, col="red", add=TRUE)
```

Neighbours list 1st order contig **Neighbours list knn=4** **Neighbours list knn=10**



The nb2listw function converts neighbours into a weights object using row standardisation (style W) as a default.

```
nb2listw_contig<-nb2listw(nb_Lux_contig)
nb2listw_k4<-nb2listw(nb_Lux_k4)
nb2listw_k10<-nb2listw(nb_Lux_k10)
```

Global test: Moran's I Then only we can compute a global test, using Moran's I

```
moran.test(Lux116sdf_unemp$resid.Model2, listw =nb2listw_contig )
```

```
##
##  Moran I test under randomisation
##
## data:  Lux116sdf_unemp$resid.Model2
## weights: nb2listw_contig
##
## Moran I statistic standard deviate = 0.083915, p-value = 0.4666
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      -0.003843939      -0.008695652       0.003342797
```

```
moran.test(Lux116sdf_unemp$resid.Model2, listw =nb2listw_k4 )
```

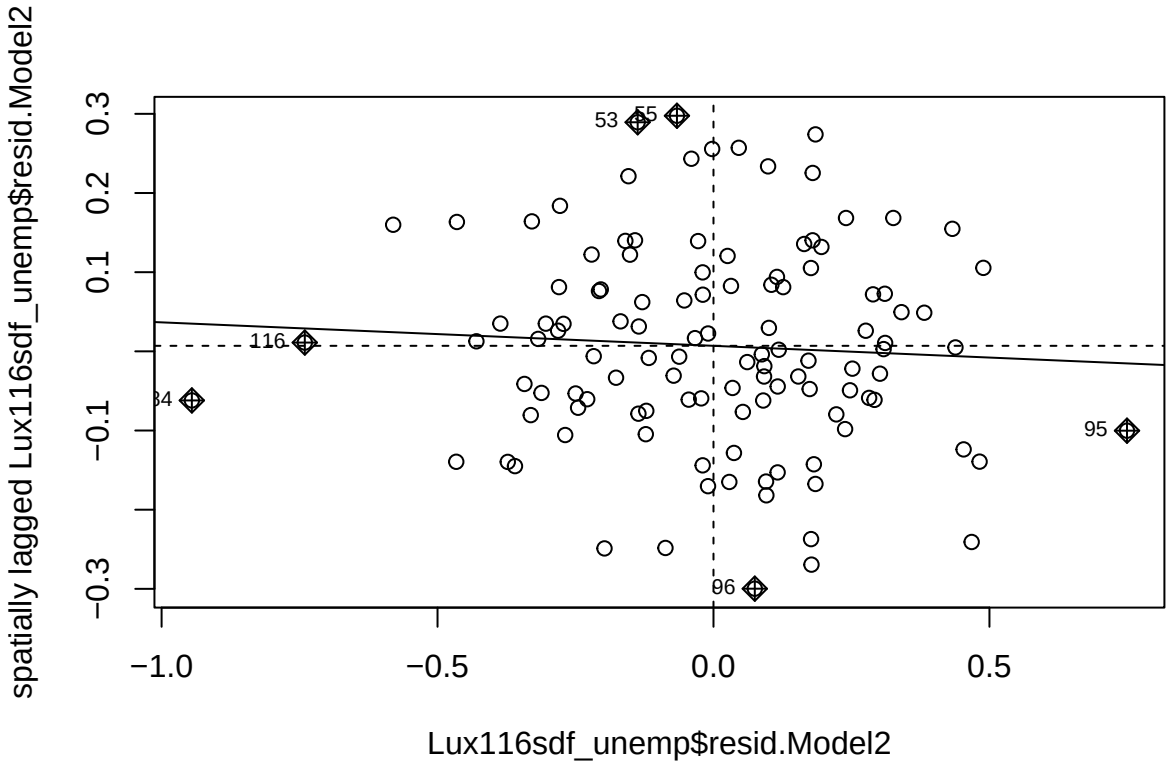
```
##
##  Moran I test under randomisation
##
## data:  Lux116sdf_unemp$resid.Model2
## weights: nb2listw_k4
##
## Moran I statistic standard deviate = -0.33993, p-value = 0.633
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      -0.029624338      -0.008695652       0.003790575
```

```
moran.test(Lux116sdf_unemp$resid.Model2, listw =nb2listw_k10)
```

```
##
##  Moran I test under randomisation
##
## data:  Lux116sdf_unemp$resid.Model2
## weights: nb2listw_k10
##
## Moran I statistic standard deviate = 0.75787, p-value = 0.2243
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.020056486      -0.008695652       0.001439294
```

from which we conclude there is no residual spatial autocorrelation across the 3 spatial weights used.

```
par(mfrow=c(1,1)) #Back to one single plot display
moran.plot(Lux116sdf_unemp$resid.Model2, listw =nb2listw_k4)
```



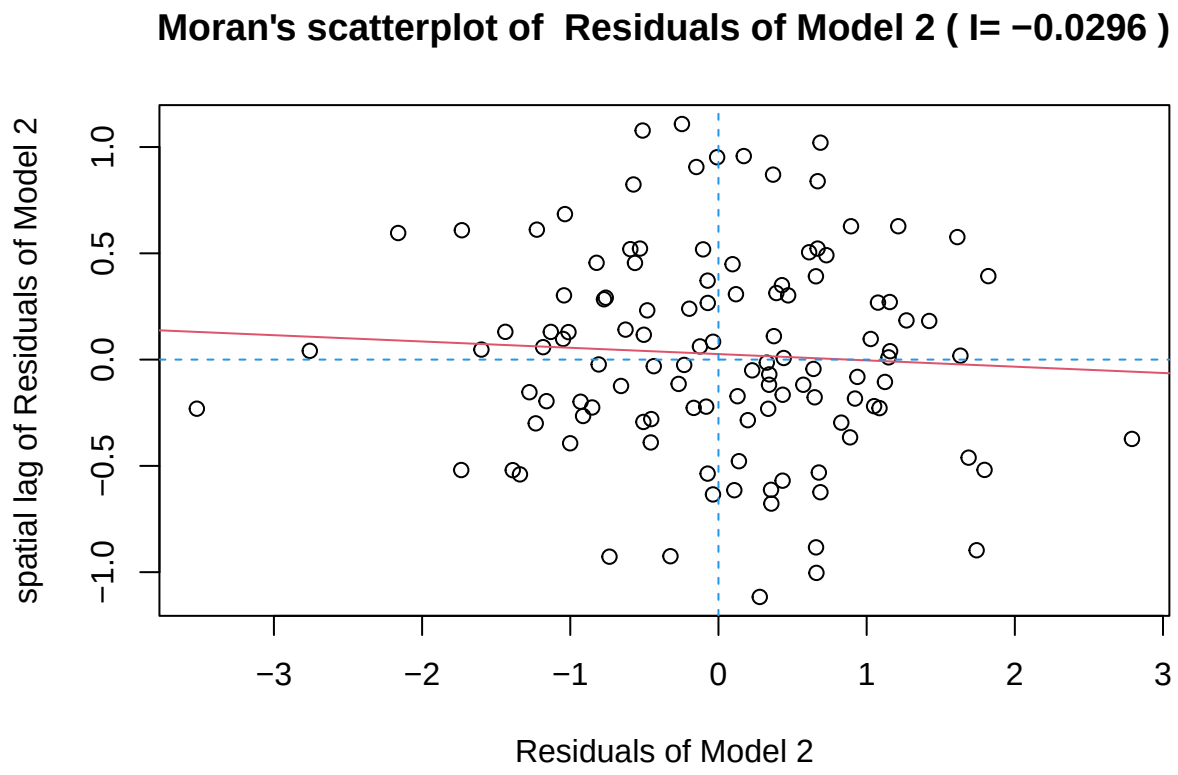
Note that Moran's I results from regressing the variable of interest and its spatially lagged counterpart after normalization. Hence the following produces same result, but the significance test.

```
library(spdep)
library(ggplot2)

moranscatterplot<-function(Attr,listw,AttrName){
  zAttr<-(Attr - mean(Attr))/sd(Attr)     #Standardise
  wzAttr <- lag.listw(listw,zAttr)        #Lag the variable
  morlm <- lm(wzAttr ~ zAttr)             #run OLS
  aa <- morlm$coefficients[1]             # to obtain intercept
  mori <- morlm$coefficients[2]           # and slope, i.e. Moran???s I
```

```
  plot(zAttr,wzAttr, xlab=AttrName, ylab=paste("spatial lag of", AttrName))
  abline(aa,mori,col=2)
  abline(h=0,lty=2,col=4)
  abline(v=0,lty=2,col=4)
  title(main = paste("Moran's scatterplot of ",AttrName, "( I=",round(mori,4),")"))
}

moranscatterplot(Lux116sdf_unemp$resid.Model2,nb2listw_k4, "Residuals of Model 2")
```




```
lisa<-localmoran(Lux116sdf_unemp$resid.Model2,nb2listw_k4)

head(lisa)
```

Local Index of Spatial Association

```
##             Ii          E.Ii     Var.Ii        Z.Ii Pr(z != E(Ii))
## 1  0.323374116 -0.0011904199 0.03357368  1.77133853     0.07650442
## 2 -0.119137309 -0.0110692725 0.30910147 -0.19437783     0.84588005
## 3  0.018633478 -0.0057301670 0.16087467  0.06074331     0.95156364
## 4  0.005855536 -0.0004696280 0.01325458  0.05494001     0.95618626
## 5  0.937163717 -0.0227918407 0.62890147  1.21048614     0.22609241
## 6 -0.066274232 -0.0001002939 0.00283170 -1.24355028     0.21366513
```

```
summary(lisa)
```

```
##        Ii                E.Ii                Var.Ii              Z.Ii
##  Min.   :-1.57619   Min.   :-0.1086869   Min.   :0.0000198   Min.   :-2.25882
##  1st Qu.:-0.21967   1st Qu.:-0.0105380   1st Qu.:0.0306232   1st Qu.:-0.72948
##  Median :-0.02088   Median :-0.0038183   Median :0.1074045   Median :-0.11105
```

```
##  Mean   :-0.02962   Mean   :-0.0086956   Mean   :0.2374397   Mean   :-0.09333
##  3rd Qu.: 0.16854   3rd Qu.:-0.0010857   3rd Qu.:0.2944207   3rd Qu.: 0.57349
##  Max.   : 0.93716   Max.   :-0.0000007   Max.   :2.7354164   Max.   : 2.08520
##  Pr(z != E(Ii))
##  Min.   :0.02389
##  1st Qu.:0.26909
##  Median :0.54819
##  Mean   :0.51114
##  3rd Qu.:0.78542
##  Max.   :0.98174
```

REMINDER: the LISA statistics Ii is the product of a variable x and the spatial lag of x. Ii is not to be confused with the spatial lag of x. Ii is positive for HH and LL quadrants and negative otherwise, i.e. for HL and LH.

Moran scatterplot quadrants (HH, LL, HL, LH) are based on x and the spatial lag of x. The signifcance of Ii is then used to stress those points that are located in significant clusters. Looking at the resulting maps while changing the significance level and the weight matrix (definition of neighbours) is good practice to explore the robustness of clusters.

Mapping local association involves identifying each record i within one of the four quadrants and applying colours only to those that have a significant Ii. For example:

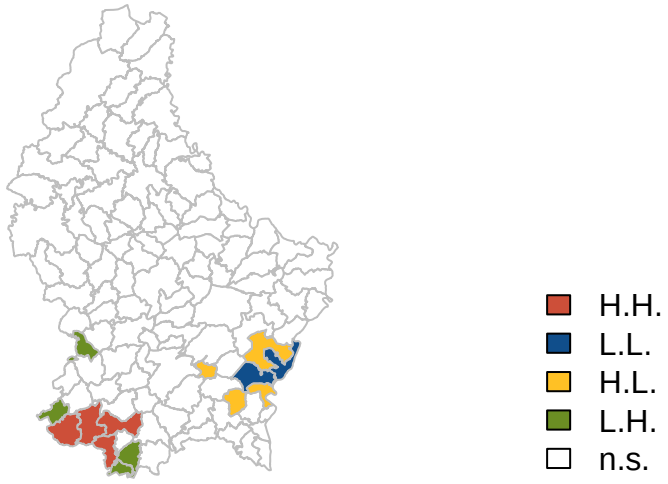
```
plotmaplisa<-function(sdf,Attr,listw,signiflevel=0.05)
  {
  AttrName <- deparse(substitute(Attr))
  lisa<-localmoran(Attr,listw)
    # Identify quadrant and significance
    zAttr<-(Attr - mean(Attr))/sd(Attr) #Standardise
    lagzAttr <- lag.listw(listw,zAttr) #Standardize lagged variable
    LISAType_qdrt<-vector(mode="numeric",length=nrow(lisa))
    LISAType_qdrt[zAttr>0 & lagzAttr>0] <- 1 #HH
    LISAType_qdrt[zAttr<0 & lagzAttr<0] <- 2 #LL
    LISAType_qdrt[zAttr>0 & lagzAttr<0] <- 3 #HL
    LISAType_qdrt[zAttr<0 & lagzAttr>0] <- 4 #LH
    LISAType_signif<-LISAType_qdrt
    LISAType_signif[lisa[,5]>signiflevel]<-5 #n.s.

    LISAcolors <- c("tomato3", "dodgerblue4", "goldenrod1", "olivedrab", "white")
    #Conventional LISA mapping uses light red for HL instead of goldenrod1 and light blue for LH instea
    plot(sdf, border="grey", col=LISAcolors[LISAType_signif], main = AttrName)
    legend("bottomright",legend=c("H.H.","L.L.","H.L.","L.H.","n.s."), fill=LISAcolors,cex=1, bty="n")
}
```

The LISA map for residuals is then obtained by

```
plotmaplisa(Lux116sdf_unemp,Lux116sdf_unemp$resid.Model2,listw=nb2listw_k10)
```

Lux116sdf_unemp$resid.Model2



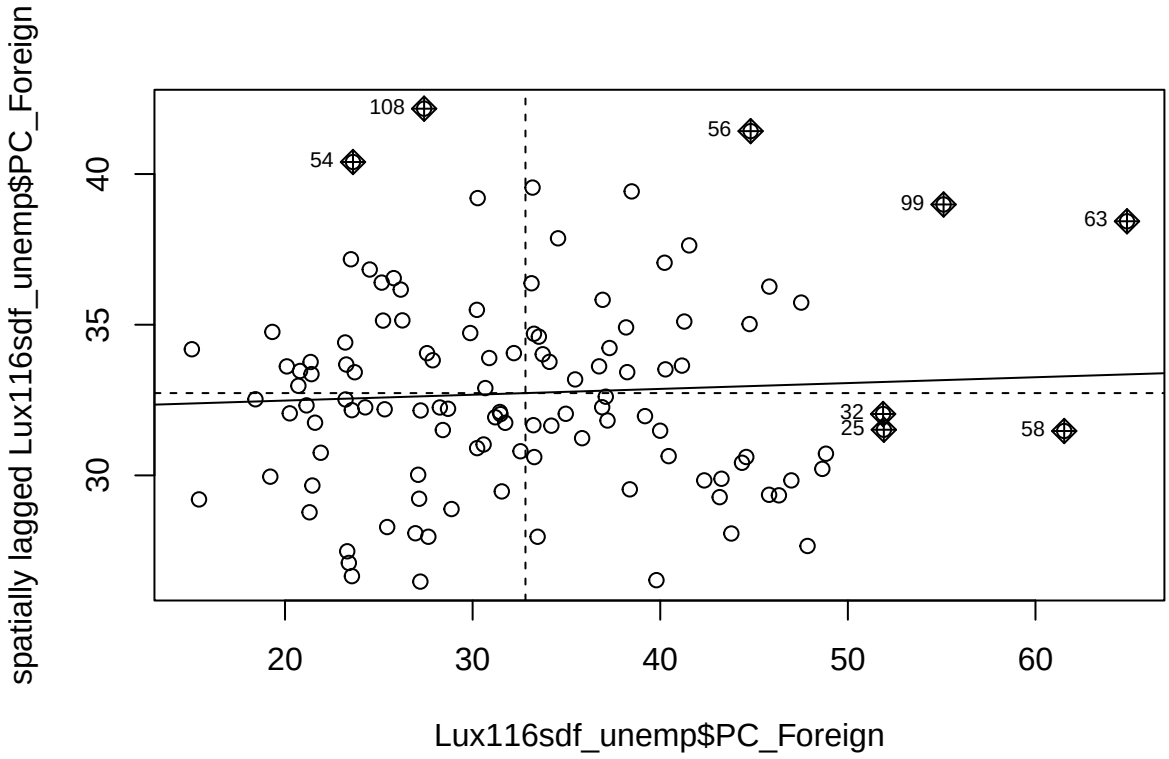
The map shows there is little case for clusters of residual spatial autocorrelation.

We can of course also apply the global and local analysis to input data such as population density or the percentage of foreigners

```
moran.test(Lux116sdf_unemp$PC_Foreign, listw =nb2listw_k10 )
```

```
##
##  Moran I test under randomisation
##
## data:  Lux116sdf_unemp$PC_Foreign
## weights: nb2listw_k10
##
## Moran I statistic standard deviate = 0.73824, p-value = 0.2302
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.019380036      -0.008695652       0.001446318
```

```
moran.plot(Lux116sdf_unemp$PC_Foreign, listw =nb2listw_k10)
```




```
lisa<-localmoran(Lux116sdf_unemp$PC_Foreign,nb2listw_k10)
plotmaplisa(Lux116sdf_unemp,Lux116sdf_unemp$PC_Foreign,nb2listw_k10)
```

Lux116sdf_unemp$PC_Foreign