

## **Session 3 Exercise: Spatial regression models**

Often GeoDa is perfectly fine as the sole software for spatial statistics (and it is user-friendly). Yet, sometimes a more versatile statistical environment will be desirable. The “R Project for Statistical Programming” is one such software. Please note that it is not necessarily better than GeoDa or other software packages, but it does have a few commonly cited merits: wide user base with lots of online sources of help, guidebooks, and tutorials; additional modules for specific purposes ready to be added to the core; fast computing and nice programming environment.

If you correctly *type* (i.e. do not copy/paste) the R commands found in this exercise, you will be okay (most of the time errors are due to misspelling or forgetting to set your working directory). Below are some additional helpful pieces of advice, if you would like to know more about R:

- R is case sensitive (to a very meticulous amount of detail);
- Having your data in `.csv` format is a good practice; however, R can also read shapefiles (their `.dbf` component);
- Data are imported in R usually as a “data-frame” (holding numeric and string variables, similar to Excel);
- R has a graphical user interface (GUI),
  - in which there are buttons to import, save, install packages, etc.
  - but you will not find buttons for statistical operations inside this GUI. For that purpose, you type the commands using a certain (not complicated) syntax.
- A nice online place to find how to do things with R is <http://www.statmethods.net/>
- Although R is fine, RStudio is an extra program that wraps around basic R and makes things much easier and fancier to work with (you have to install both R and RStudio).

You will be learning the main ways to do things in R as we progress with this exercise. Before getting into R, we will do a few introductory regressions with GeoDa.

### **A. Introductory Spatial Regression with GeoDa**

The creation of alternative spatial weights files and the utilization of spatial (auto)correlation tests help to understand what sort of factors may influence the phenomenon under investigation, as well as how does the phenomenon look in geographical terms. A main outcome of such efforts should be the identification of a plausible model specification.

Generally, if there is sufficient evidence for spatial autocorrelation in the dependent variable, then it is very likely that spatial regressions are needed versus standard non-spatial regressions. However, a more formal procedure needs to be followed in order to confer whether this is indeed the case, and in fact what might be the best spatial regression specification among a few commonly used alternatives.

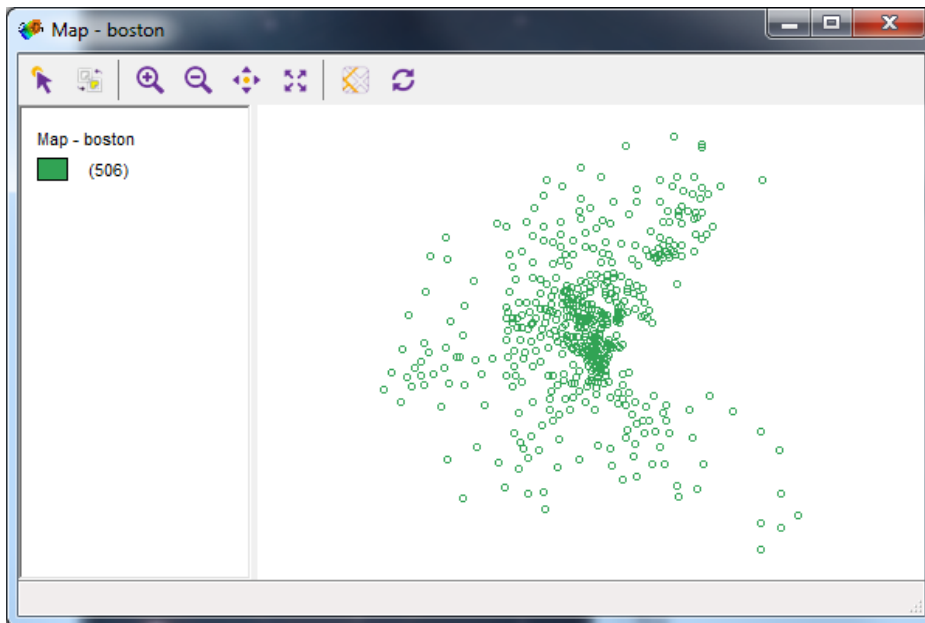
This procedure starts with a standard, non-spatial OLS (ordinary least squares) regression. The behavior of this model is then assessed through a set of tests. The results of the tests guide you through the selection of the most appropriate spatial model. However, it is always possible that the tests indicate that there is no need to employ a spatial model, which should be respected if you are sure that there are no other problems with the appropriateness of the employed spatial weights, the quality of the data, or the selection of explanatory variables.

In other words, it is still very important that you begin with sufficiently investigated weights files and autocorrelation tests, and with a solid OLS model.

## 1 – Preparing point data

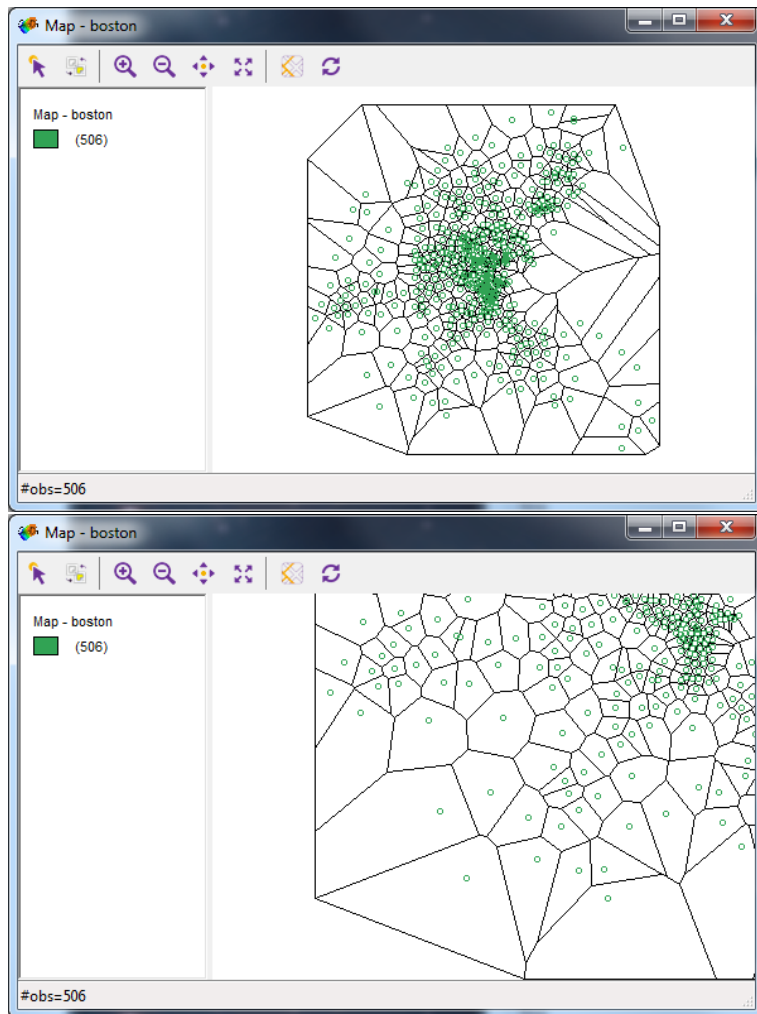
We will work with the file `boston.shp`. The focus of this exercise is only on following a simple procedure from OLS towards spatial models with GeoDa.

1.1 In GeoDa, open the file `boston.shp`:



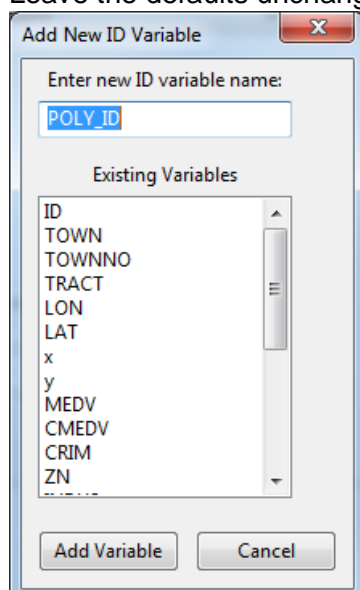
**Notice that we are dealing now with a different kind of geographic entity: sparse points (rather than contiguous polygons).** One way to proceed is to calculate nearest neighbor or other distance-based weights. However, GeoDa will not accept to use distance-based weights (partly due to theoretical concerns). Another way to proceed is to utilize the **Thiessen polygons** of the points, transforming the problem of loosely connected points into a problem of contiguous polygons. From an economic geography point of view, Thiessen polygons represent the “market area” of an entity surrounded by competitor entities of equal potential, over homogeneous geography. More generally, it is a way to approximate *the area of influence* for each point. This is done on-the-fly by GeoDa. You can save the resulting polygons in a new shapefile, but it is not necessary – GeoDa will use the Thiessen polygons whenever needed and you can keep the points file as your working file).

1.2 As an illustration (not needed in the analysis), right-click anywhere on the points map and select `Thiessen Polygons > Display Thiessen Polygons`. The underlying polygon structure will be displayed:

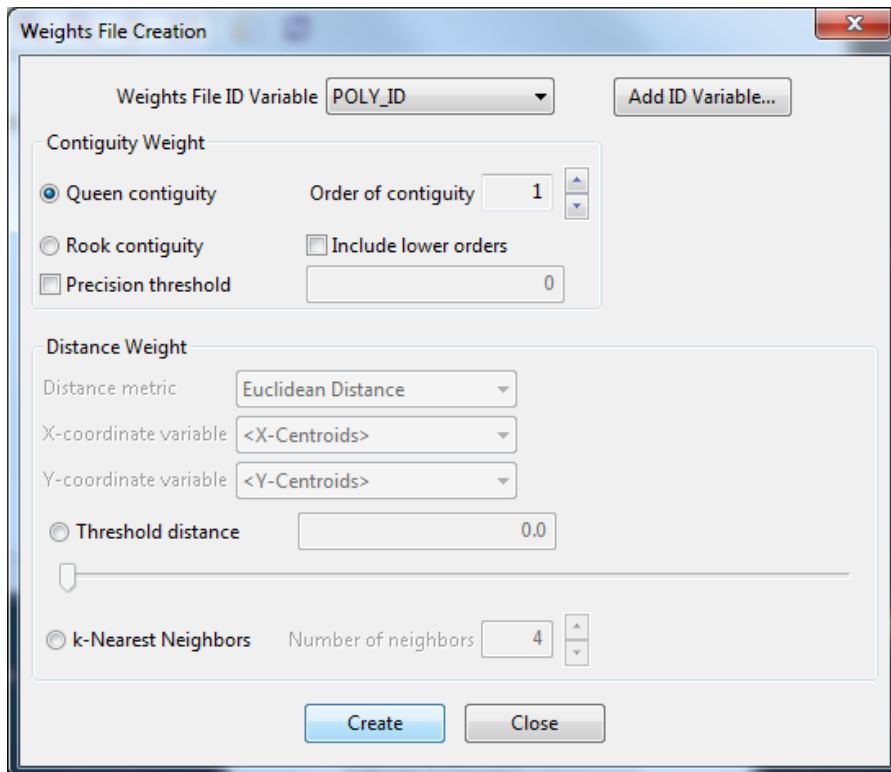


1.3 Now create a queen contiguity weights file by bringing up Tools > Weights Manager:

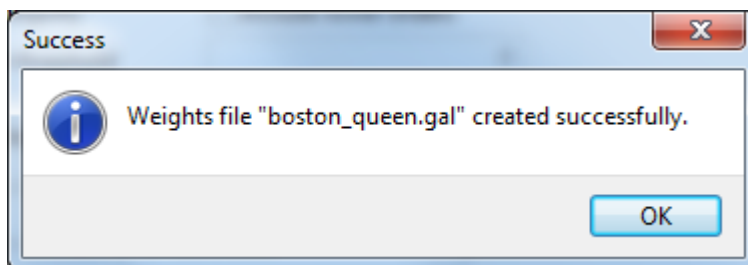
- Click Create in the Weights Manager window
- In the Weights File Creation window that opens, click Add ID Variable...
- Leave the defaults unchanged and press Add Variable:



- Back in the Weights File Creation Window, select Queen contiguity, leave the Order of contiguity to value 1, and press Create and given the weights file the name boston\_queen.gal



- After you select the save directory and name for the file, a confirmation dialog will appear:



You are now set for your regression workflows.

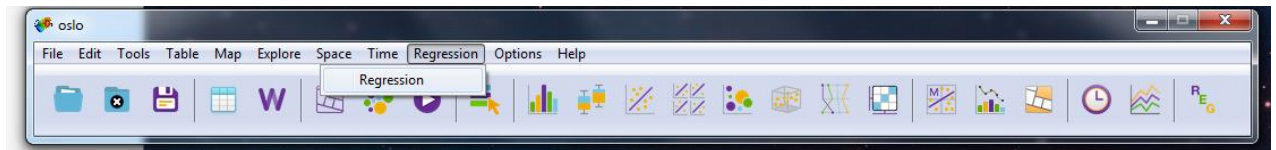
## 2 – Non-spatial OLS and LM diagnostics

Here, we are interested on air quality. Our dependent variable is `CMEDV` (median house value in \$1000). We also control for other factors possibly connected to environmental quality: proximity to water (`CHAS`), and proximity to employment nodes (`DIS`) and to radial highways (`RAD`) that theoretically may have higher pollution levels. We also control for the size of dwellings by approximating it with the number of rooms (`RM`), and for a few other factors:




<b>CMEDV</b>	A numeric vector of corrected median values of owner-occupied housing in USD 1000
<b>CRIM</b>	A numeric vector of per capita crime
<b>CHAS</b>	A factor with levels 1 if tract borders Charles River; 0 otherwise
<b>NOX</b>	A numeric vector of nitric oxides concentration (parts per 10 million) per town

<b>RM</b>	A numeric vector of average numbers of rooms per dwelling
<b>DIS</b>	A numeric vector of weighted distances to five Boston employment centers
<b>RAD</b>	A numeric vector of an index of accessibility to radial highways per town (constant for all Boston tracts)
<b>TAX</b>	A numeric vector full-value property-tax rate per USD 10,000 per town (constant for all Boston tracts)
<b>PTRATIO</b>	A numeric vector of pupil-teacher ratios per town (constant for all Boston tracts)
<b>B</b>	A numeric vector of $1000 \cdot (B_k - 0.63)^2$ where $B_k$ is the proportion of blacks
<b>LSTAT</b>	A numeric vector of % less well-off population (notice the original metadata description though...)

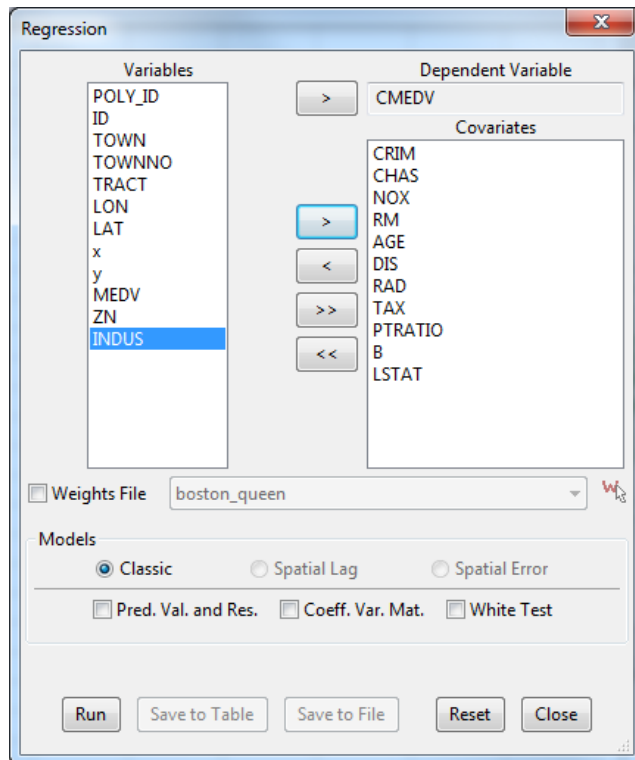
**2.1** You will first construct a standard OLS specification. On the main menu bar, click `Regression` > `Regression`:



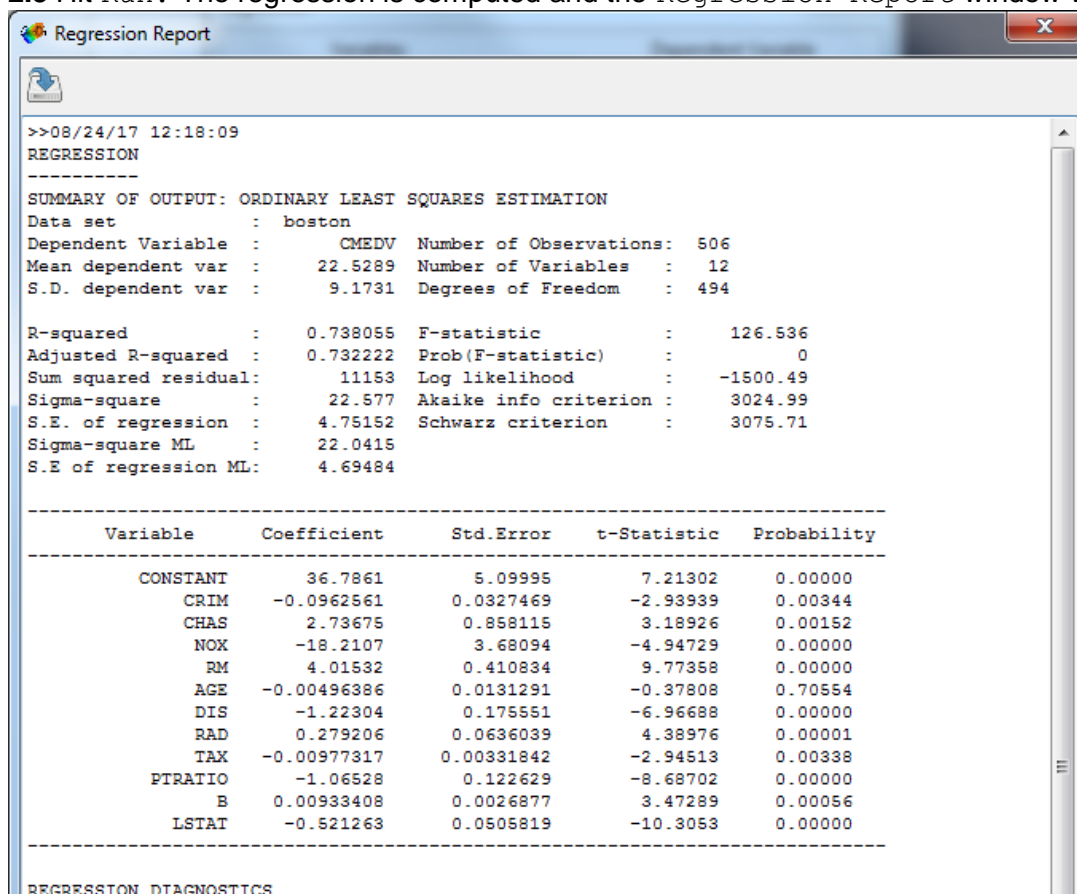
**2.2** A window will now open, in which you select the dependent and independent variables, the weights file, and the type of regression you wish to run:

- On the upper left-hand panel you can see the list of available variables (contained in the shapefile's database).
- Click on the variable `CMEDV` to highlight it. Then click on the  button which is located right on the left of the `Dependent Variable` slot, in order to assign `sqm_price` as the dependent variable.
- Similarly, highlight (one-by-one) variables `CRIM`, `CHAS`, `NOX`, `RM`, `AGE`, `DIS`, `RAD`, `TAX`, `PTRATIO`, `B`, `LSTAT` on the left-hand panel and then click on a different  button, the one responsible for the `Independent Variables` slot.
- Next, check the `Weights File` option and select the `boston_queen` file that you created previously to serve as the weights file. It should be already available in the drop-down list, but if not you can click the  button to navigate to its folder and select it.
- Finally, in the `Models` area, select the `Classic` radio button.
- Hit `Run`. The green progress bar will fill pretty fast, and then hit `View Results`.

The above settings in summary:



2.3 Hit Run . The regression is computed and the Regression Report window will appear:



If you focus for now only to the upper part of report (i.e. before and excluding the part which starts with Regression Diagnostics), you will notice a high R-squared value of 0.7. You will also

notice that most of the variables are highly significant (Probability) and their signs are as expected, while some others give surprises.

## 2.4 Now look at the lower part of the regression report:

```
REGRESSION DIAGNOSTICS
MULTICOLLINEARITY CONDITION NUMBER    82.814494
TEST ON NORMALITY OF ERRORS
TEST          DF          VALUE          PROB
Jarque-Bera      2          877.7597          0.00000

DIAGNOSTICS FOR HETEROSKEDASTICITY
RANDOM COEFFICIENTS
TEST          DF          VALUE          PROB
Breusch-Pagan test  11          182.0169          0.00000
Koenker-Bassett test  11          47.9354          0.00000

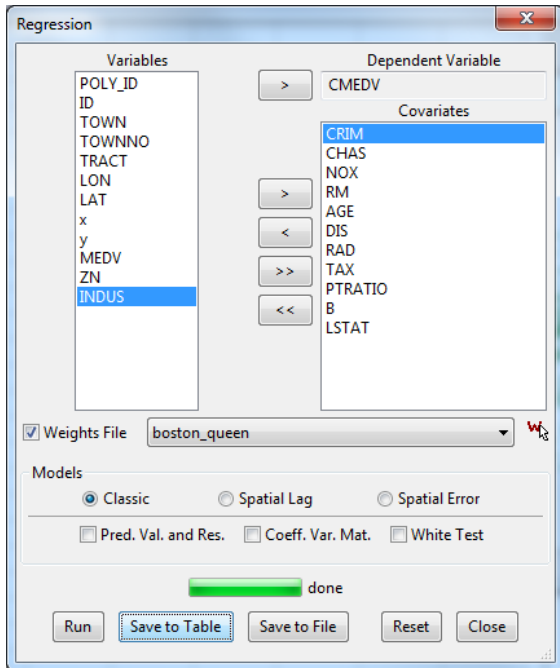
DIAGNOSTICS FOR SPATIAL DEPENDENCE
FOR WEIGHT MATRIX : boston_queen
(row-standardized weights)
TEST          MI/DF          VALUE          PROB
Moran's I (error)    0.3314          13.5498          0.00000
Lagrange Multiplier (lag)    1          79.3355          0.00000
Robust LM (lag)        1          1.1368          0.28633
Lagrange Multiplier (error)  1          159.5580          0.00000
Robust LM (error)      1          81.3592          0.00000
Lagrange Multiplier (SARMA)  2          160.6948          0.00000
===== END OF REPORT =====
```

The section called `DIAGNOSTICS FOR SPATIAL DEPENDENCE` guides you in selecting the most appropriate spatial regression model.

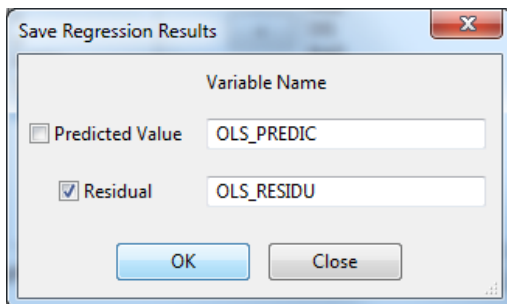
1. You first need to inspect the `Moran's I (error)` significance (named `PROB`). In our case, it is highly significant, indicating that the residuals (errors) of the non-spatial OLS regression that you just estimated are not random. This is a good motivation for running a spatial model.
2. We now want to inspect the significances (`PROB`) of the Lagrange Multipliers (LM) for whether a spatial lag and a spatial error model are (independently from each other) preferred over the OLS model.
3. The LM for the spatial lag model (`Lagrange Multiplier (lag)`) is highly significant with `PROB = 0.00000`.
4. The same holds for the LM of the spatial error model (`Lagrange Multiplier (error)`).
5. Steps 3 and 4 indicate that a spatial specification should be preferred. However, since the LM for both the lag and error models are highly significant, you cannot yet know which spatial specification you should prefer.
6. Move on to inspecting the robust versions of the LM, which assess the lag and error models against each other.
7. The robust LM for the lag model (`Robust LM (lag)`) is insignificant with `PROB = 0.28633`.
8. The robust LM for the error model (`Robust LM (error)`) is highly significant with `PROB = 0.00000`.
9. This indicates that a spatial error specification is preferable.
10. The LM test for the SARMA model (`Lagrange Multiplier (SARMA)`) is also highly significant, but it is because it borrows the significances of the lag and error models.

**2.5** Another way to verify that a spatial specification is needed is to perform autocorrelation tests in the non-spatial OLS residuals. Once you have run your OLS, press the button `Save to Table`:

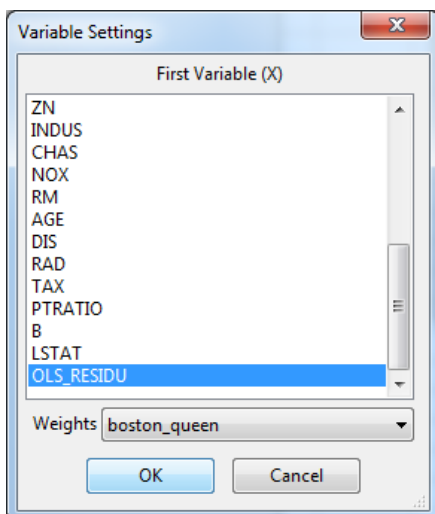




A new window called `Save Regression Results` opens. Select only the `Residual` option, leave the new variable's default name unchanged, and hit `OK`:

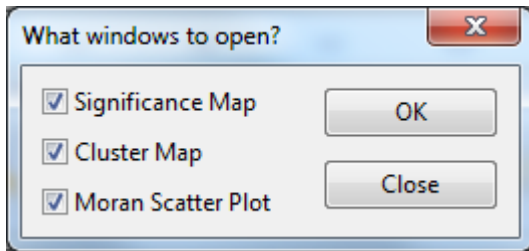


**2.6** Let's now inspect the residuals. Use the tool `Space > Univariate Local Moran's I`. Select the variable `OLS_RESIDU`, use the weights file `boston_queen`, and press `OK`:

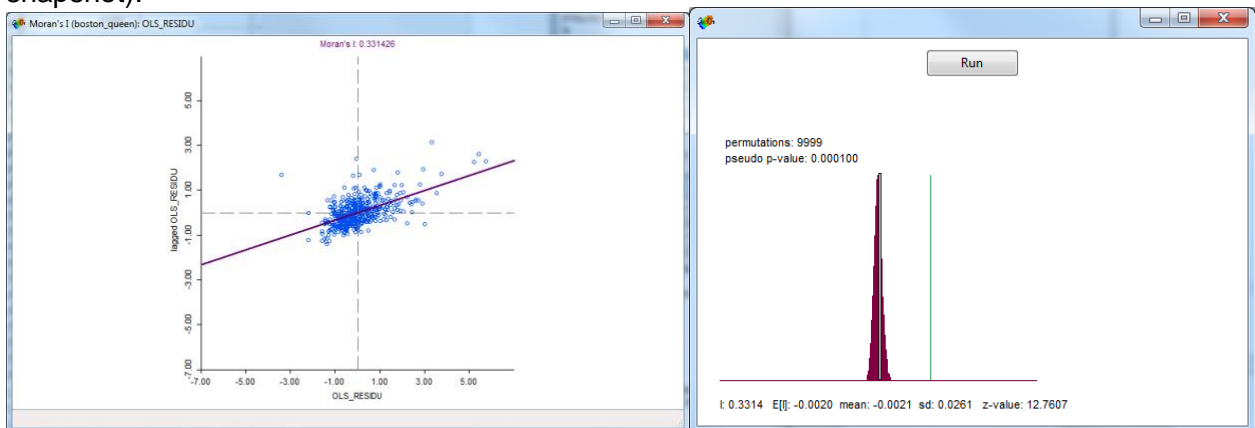




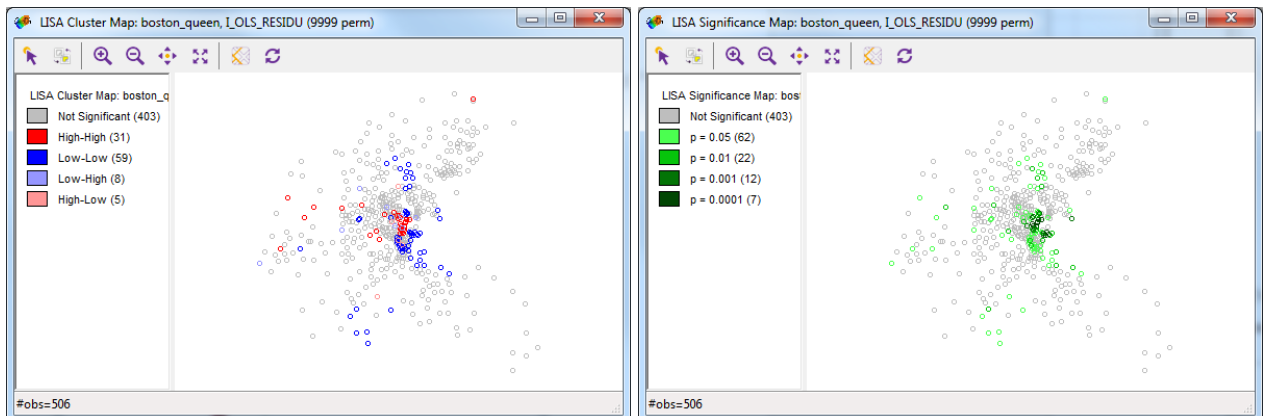
In the What windows to open? pop-up, select all three options and then hit OK:



**2.7** The global Moran's I scatterplot indicates statistically significant positive autocorrelation in the OLS residuals ( $I = 0.3$  in the first snapshot below, and pseudo  $p$ -value = 0.0001 in the second snapshot):

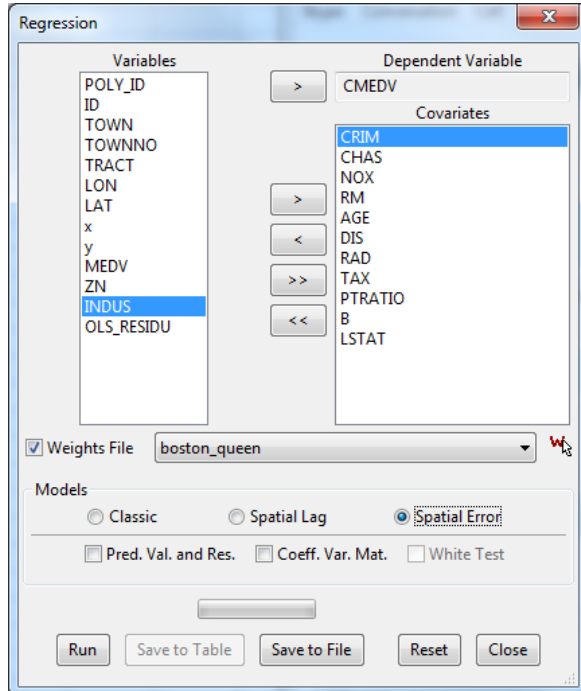


The LISA maps also indicate statistically significant local patterns in the OLS residuals:



### 3 – Spatial error regression

3.1 The LM tests in step 2.4 indicated that we should prefer a spatial error specification. In the Regression window, select the same variables as before, but now select the Spatial Error radio button:



3.2 The regression output is as follows:

```
SUMMARY OF OUTPUT: SPATIAL ERROR MODEL - MAXIMUM LIKELIHOOD ESTIMATION
Data set          : boston
Spatial Weight    : boston_queen
Dependent Variable : CMEDV
Number of Observations: 506
Mean dependent var : 22.528854
Number of Variables : 12
S.D. dependent var : 9.173098
Degrees of Freedom : 494
Lag coeff. (Lambda) : 0.713305

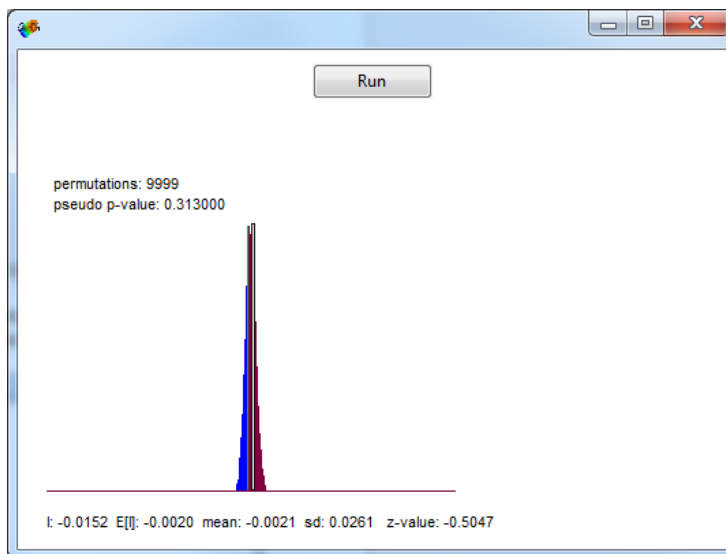
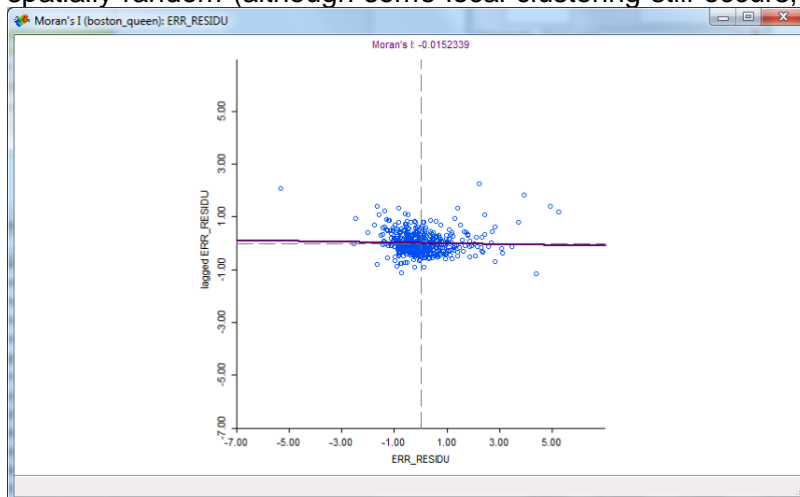
R-squared          : 0.827200
R-squared (BUSE)   : -
Sq. Correlation    : -
Log likelihood     : -1424.942331
Sigma-square       : 14.5404
Akaike info criterion : 2873.88
S.E of regression  : 3.81318
Schwarz criterion  : 2924.6
```

Variable	Coefficient	Std. Error	z-value	Probability
CONSTANT	31.1796	5.37701	5.7987	0.00000
CRIM	-0.12992	0.0274503	-4.73293	0.00000
CHAS	-0.803683	0.871902	-0.921758	0.35665
NOX	-21.7627	4.95447	-4.39254	0.00001
RM	4.39021	0.35663	12.3103	0.00000
AGE	-0.0286732	0.0138338	-2.0727	0.03820
DIS	-1.42358	0.294885	-4.82759	0.00000
RAD	0.306204	0.0762316	4.01676	0.00006
TAX	-0.0123417	0.00337035	-3.66184	0.00025
PTRATIO	-0.680469	0.144133	-4.72113	0.00000
B	0.0104018	0.00306472	3.39405	0.00069
LSTAT	-0.42467	0.0507094	-8.3746	0.00000
LAMBDA	0.713305	0.0399802	17.8415	0.00000

```
REGRESSION DIAGNOSTICS
DIAGNOSTICS FOR HETEROSKEDASTICITY
RANDOM COEFFICIENTS
TEST          DF      VALUE      PROB
Breusch-Pagan test    11      209.7191    0.00000

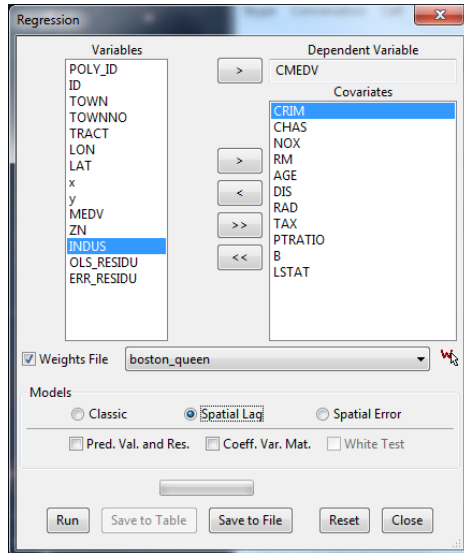
DIAGNOSTICS FOR SPATIAL DEPENDENCE
SPATIAL ERROR DEPENDENCE FOR WEIGHT MATRIX : boston_queen
TEST          DF      VALUE      PROB
Likelihood Ratio Test    1      151.1029    0.00000
===== END OF REPORT =====
```

- There are a few changes in the parameter estimates (any idea why?)
- The beta coefficients can be interpreted as in OLS
- The spatial error coefficient  $\text{LAMBDA}$  is estimated to 0.7, which provides an indication of the strength of the spatial interaction in the data. This is the  $\lambda$  component of  $\lambda \mathbf{W} \mathbf{u}$  in the equations of the lecture slides (it is a beta-coefficient, but the beta-coefficient of the spatially lagged variable is called “lambda”).
- An autocorrelation test in the residuals of the spatial error estimation shows that they are spatially random (although some local clustering still occurs, but much less than in the OLS):



## 4 – Spatial lag regression

4.1 Let's now pretend that we wanted to run a spatial lag model. In the Regression window, select the same variables as before, but switch to the Spatial Lag radio button:



The results:

### SUMMARY OF OUTPUT: SPATIAL LAG MODEL - MAXIMUM LIKELIHOOD ESTIMATION

```
Data set      : boston
Spatial Weight : boston_queen
Dependent Variable : CMEDV   Number of Observations: 506
Mean dependent var : 22.5289 Number of Variables : 13
S.D. dependent var : 9.1731  Degrees of Freedom : 493
Lag coeff. (Rho) : 0.391823

R-squared      : 0.782449   Log likelihood : -1460.96
Sq. Correlation : -        Akaike info criterion : 2947.93
Sigma-square   : 18.306    Schwarz criterion : 3002.87
S.E of regression : 4.27855
```

Variable	Coefficient	Std.Error	z-value	Probability
W_CMEDV	0.391823	0.0394787	9.92491	0.00000
CONSTANT	14.2485	5.07738	2.80627	0.00501
CRIM	-0.0704621	0.0299503	-2.35263	0.01864
CHAS	1.12719	0.7791	1.44678	0.14796
NOX	-11.4057	3.40202	-3.35262	0.00080
RM	3.89061	0.372674	10.4397	0.00000
AGE	-0.00359546	0.0118441	-0.303567	0.76146
DIS	-0.995009	0.162715	-6.11504	0.00000
RAD	0.225173	0.0579965	3.88253	0.00010
TAX	-0.0081511	0.00300582	-2.71177	0.00669
PTRATIO	-0.625859	0.117707	-5.31711	0.00000
B	0.00784121	0.00242738	3.23032	0.00124
LSTAT	-0.361642	0.0480533	-7.52586	0.00000

### REGRESSION DIAGNOSTICS DIAGNOSTICS FOR HETEROSKEDASTICITY RANDOM COEFFICIENTS

```
TEST      DF      VALUE      PROB
Breusch-Pagan test      11      246.0554      0.00000
```

### DIAGNOSTICS FOR SPATIAL DEPENDENCE

SPATIAL LAG DEPENDENCE FOR WEIGHT MATRIX : boston\_queen

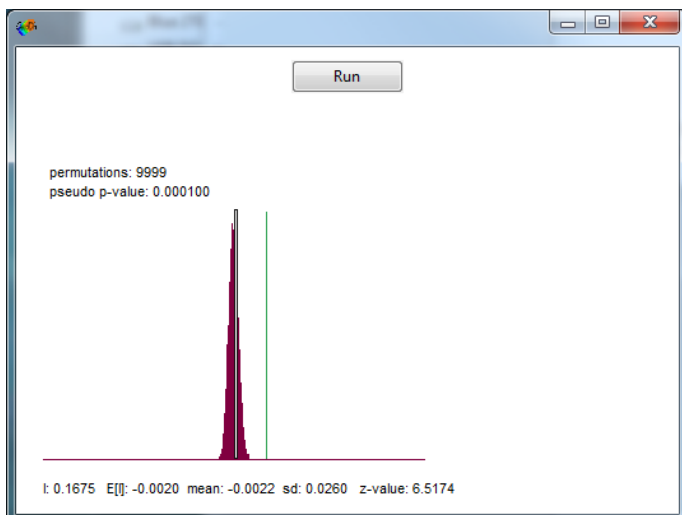
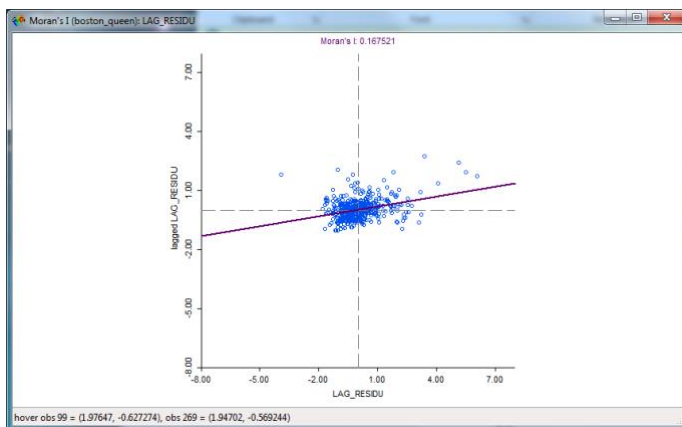
```
TEST      DF      VALUE      PROB
Likelihood Ratio Test      1      79.0597      0.00000
```

===== END OF REPORT =====

## 4.2

- The new variable  $W\_CMEDV$  has been automatically added to the regression and can be seen in the output as the first in the list of regression coefficients. This is the spatially lagged form of your dependent variable.
- The regression coefficient of  $W\_PRICE$  is 0.4, and highly significant (Probability = 0.0000000). This is the  $\rho$  ("rho") component of  $\rho W_y$  in the equations of the lecture slides.
- A good way to start is to see  $W_y$  as the effect that the neighbouring prices have on the price of a particular property.
- Including  $W_y$  (as opposed to  $W_u$  in the error regression) also introduces us to an important concept: endogenous and exogenous parameters.  $W_y$  is endogenous to the dependent variable because they influence each-other bi-directionally. All the rest of the independent variables are exogenous to the dependent variable because they only influence – they do not get influenced.  $W_u$  is also exogenous, because we accepted that it contains unobserved independent variable.
- Endogenous relationships are also a pathway to complex dynamics, and are frequently called "feedback loops". Higher neighbourhood prices increase a single property's price, but this increase is then contributing to the overall neighbourhood level, which in turn contributes to the individual property's price, and so on so forth in a positive feedback loop.
- However, you cannot interpret the regression coefficients of the spatial lag regression at their face value, because they contain both pure and spatial spillover effects. Separation of these two cannot be performed in GeoDa, but is easily done in R.

**4.3** Lastly, we can see why (among others) the LM multipliers did not favor a spatial lag model. It failed to address spatial autocorrelation in the residuals:



## B. Spatial Regression Models with R

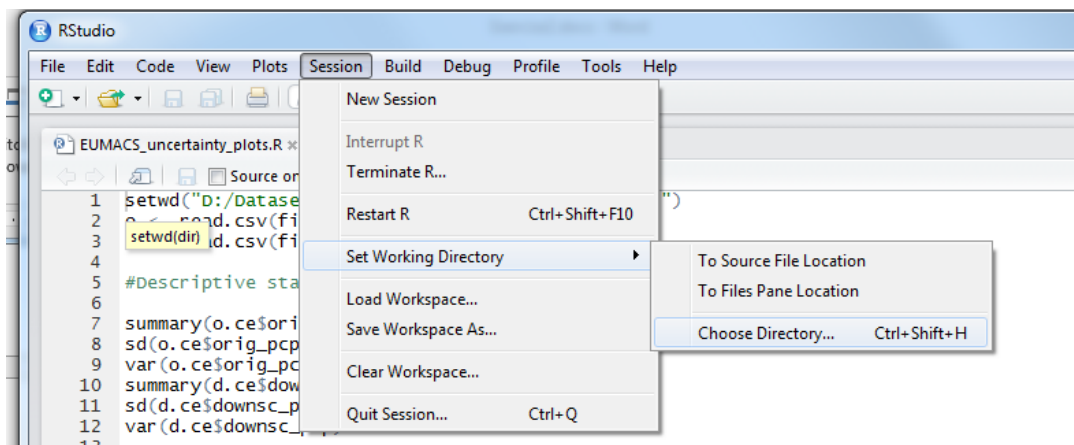
### 1 – Using GeoDa to create spatial weights

**1.1** Currently, the handiest and fastest workflow is to use GeoDa to generate spatial weights files from any working dataset. The weights files generated by GeoDa can be then imported easily in R.

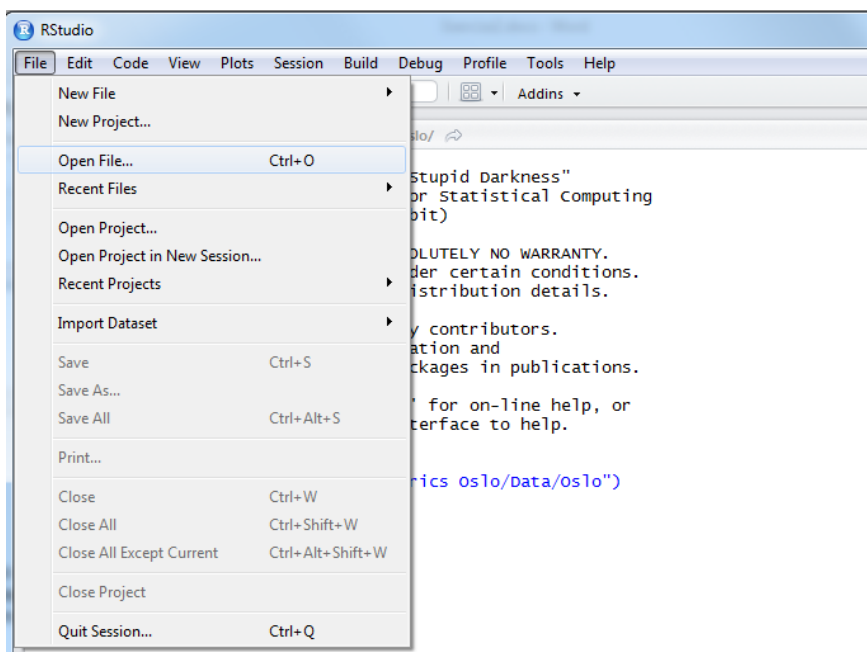
Use the weights file `boston_queen.gal` that you created earlier with GeoDa (step A-1.3 in p. 3).

### 2 – Importing data and weights into R (RStudio)

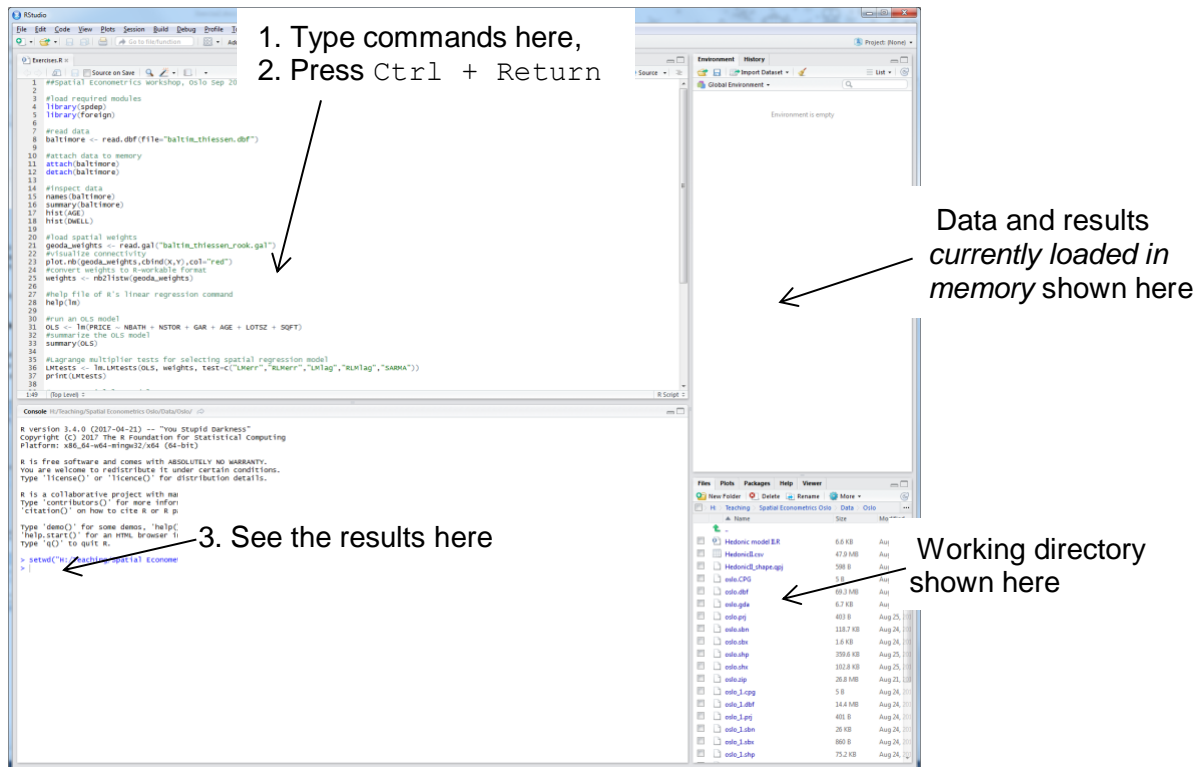
**2.1** After opening RStudio, we need to set the working directory. Go to `Session > Set Working Directory > Choose Directory` and a new window will open asking you to select a working directory. Select your working folder and then press `Select Folder`. You have to set the working directory every time you start R.



**2.2** Go to `File > Open File`. A navigation window will open, so you can browse to your working folder and select to open the file `Session_3.R`.



Your RStudio window will look something like the following picture:



**2.3** In the top-left quarter (where your R script file has been loaded and displayed), type `library(spdep)` and hit Control + Return, which is what you need to press in order to run any command you have typed (it will run one line at a time, or the selected lines if you have selected more than one line).

Now type `library(foreign)` (and then press `Ctrl+Return`); this will load an additional module that is needed to read and load shapefiles. The bottom-left quarter (Console) displays the commands you have just run, information regarding the requested operations, and when it is ready it will display the `>` symbol without anything following it (denoting that it is ready for the next command). In my case:

```
> library(spdep)
Loading required package: sp
Loading required package: Matrix
> library(foreign)
>
```

**2.4** We will load our data by telling R to read the file as the database (`*.dbf`) component of a shapefile. Type:

```
boston <- read.dbf(file="boston.dbf")
```

- The first component [`boston <-`] tells R to load your file into a new data-frame named "boston" (the name need not be the same as the shapefile – it can be chosen at will).
- The second component [`read.dbf(file="boston.dbf")`] is the actual command that tells R to read a dbf file that has the name `boston.dbf` (here the filename matters).



**2.5** In order to work efficiently with the variables in your dataset, you can “attach” the just created data-frame to the memory of R, and then simply use the contained variables with their names. Type:

```
attach(boaton)
```

To verify that the file has loaded fine, type:

```
names(boston)
```

```
> names(boston)
[1] "ID"      "TOWN"    "TOWNNO"  "TRACT"   "LON"     "LAT"     "x"       "y"       "MEDV"    "CMEDV"   "CRIM"    "ZN"
[13] "INDUS"   "CHAS"    "NOX"     "RM"      "AGE"     "DIS"     "RAD"     "TAX"     "PTRATIO" "B"       "LSTAT"
```

The names of the variables of the attached data-frame will show, as in the above image.

For more information, the summary statistics of the dataset can be displayed by typing:

```
summary(boston)
```

The following statistics are computed:

```
> summary(boston)
      ID      TOWN      TOWNNO      TRACT      LON      LAT      x      y
Min.   : 1.0   Min.   :0   Min.   : 0.00   Min.   : 1   Min.   : -71.29   Min.   :42.03   Min.   :310.9   Min.   :4655
1st Qu.:127.2   1st Qu.:0   1st Qu.:26.25   1st Qu.:1303   1st Qu.: -71.09   1st Qu.:42.18   1st Qu.:327.2   1st Qu.:4672
Median :253.5   Median :0   Median :42.00   Median :3394   Median : -71.05   Median :42.22   Median :330.5   Median :4676
Mean   :253.5   Mean   :0   Mean   :47.53   Mean   :2700   Mean   : -71.06   Mean   :42.22   Mean   :330.3   Mean   :4676
3rd Qu.:379.8   3rd Qu.:0   3rd Qu.:78.00   3rd Qu.:3740   3rd Qu.: -71.02   3rd Qu.:42.25   3rd Qu.:333.4   3rd Qu.:4680
Max.   :506.0   Max.   :0   Max.   :91.00   Max.   :5082   Max.   : -70.81   Max.   :42.38   Max.   :350.2   Max.   :4694

      MEDV      CMEDV      CRIM      ZN      INDUS      CHAS      NOX
Min.   : 5.00   Min.   : 5.00   Min.   : 0.00632   Min.   : 0.00   Min.   : 0.46   Min.   :0.00000   Min.   :0.3850
1st Qu.:17.02   1st Qu.:17.02   1st Qu.: 0.08204   1st Qu.: 0.00   1st Qu.: 5.19   1st Qu.:0.00000   1st Qu.:0.4490
Median :21.20   Median :21.20   Median : 0.25651   Median : 0.00   Median : 9.69   Median :0.00000   Median :0.5380
Mean   :22.53   Mean   :22.53   Mean   : 3.61352   Mean   :11.36   Mean   :11.14   Mean   :0.06917   Mean   :0.5547
3rd Qu.:25.00   3rd Qu.:25.00   3rd Qu.: 3.67708   3rd Qu.:12.50   3rd Qu.:18.10   3rd Qu.:0.00000   3rd Qu.:0.6240
Max.   :50.00   Max.   :50.00   Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000   Max.   :0.8710

      RM      AGE      DIS      RAD      TAX      PTRATIO      B
Min.   :3.561   Min.   : 2.90   Min.   : 1.130   Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   : 0.32
1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100   1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38
Median :6.208   Median : 77.50   Median : 3.207   Median : 5.000   Median :330.0   Median :19.05   Median :391.44
Mean   :6.285   Mean   : 68.57   Mean   : 3.795   Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67
3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188   3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
Max.   :8.780   Max.   :100.00   Max.   :12.127   Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90

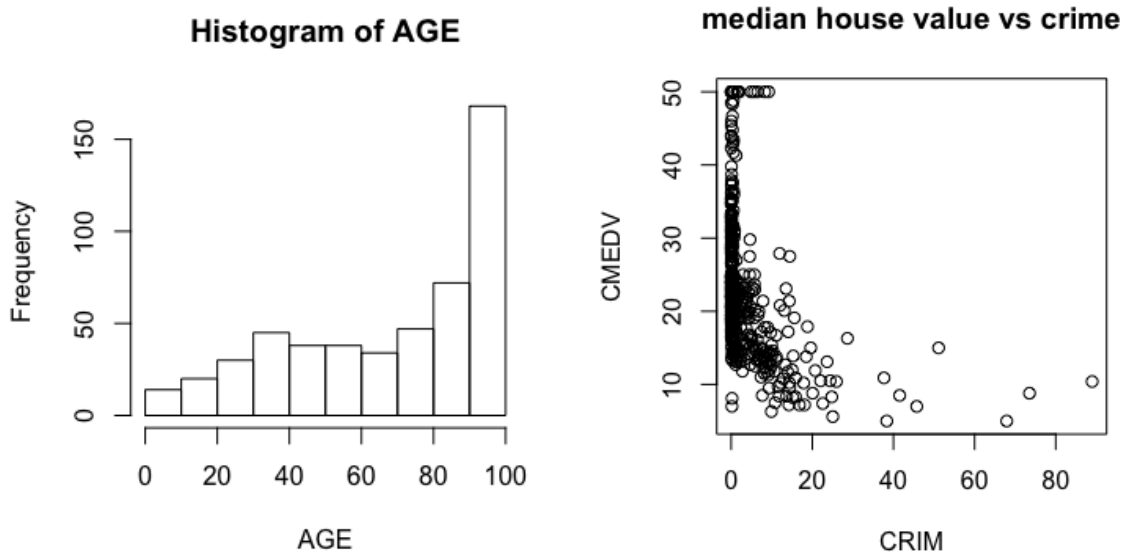
      LSTAT
Min.   : 1.73
1st Qu.: 6.95
Median :11.36
Mean   :12.65
3rd Qu.:16.95
Max.   :37.97
```

You can also display the summary statistics of just one variable, for instance:

```
> summary(AGE)
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      2.90  45.02   77.50   68.57  94.07   100.00
```

Additional functionalities for quickly inspecting the data are, for instance:

```
hist(AGE) and plot(AGE, CMEDV, main="median house value vs crime")
```



All the above inspections indicate that the information contained in the shapefile has been loaded fine, and so we are all set to begin analysis.

### 3 Importing weights into R

While the weights files generated in GeoDa can be imported easily into R, there are two points that need attention:

- You need to tell R a bit of info about the weights file, namely to indicate the unique ID variable that you used in GeoDa to generate the weights.
- Once the weights are imported, they need to be converted within R into a slightly different format.

Below are the necessary steps to import and convert the weights file:

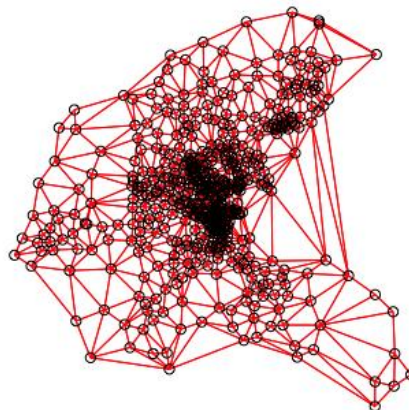
**3.1** Import the queen weights file created with GeoDa. Type:

```
geoda_weights <- read.gal("boston_queen.gal", region.id=POLY_ID)
```

This has created an object called "geoda\_weights"; in this object, R has imported the weights file that you created earlier with GeoDa.

**3.2** If you would like to visualize the connectivity between the geographic entities to which the weights file refers, you can use the following command:

```
plot.nb(geoda_weights, cbind(x, y), col="red")
```



**3.3** Lastly, the imported weights are in fact something called a “neighbors list”. R understands a different version called “listw”, and so you need to make this conversion. To achieve that for the two weight files imported above, type:

```
weights <- nb2listw(geoda_weights)
```

## 4 – Lagrange multiplier tests for specification selection

**4.1** In our context, Lagrange Multiplier (LM) tests are a set of statistics that evaluate alternative regression specifications. Based on the values of LM tests, you have to select the most appropriate spatial (or non-spatial) model. LM tests can only be obtained if you have first estimated and stored the results of a **good** non-spatial OLS model. We will be using the following variables:

Dependent variable

**CMEDV** A numeric vector of corrected median values of owner-occupied housing in USD 1000

Explanatory variables

**CRIM** A numeric vector of per capita crime

**CHAS** A factor with levels 1 if tract borders Charles River; 0 otherwise

**NOX** A numeric vector of nitric oxides concentration (parts per 10 million) per town

**RM** A numeric vector of average numbers of rooms per dwelling

**DIS** A numeric vector of weighted distances to five Boston employment centers

**RAD** A numeric vector of an index of accessibility to radial highways per town (constant for all Boston tracts)

**TAX** A numeric vector full-value property-tax rate per USD 10,000 per town (constant for all Boston tracts)

**PTRATIO** A numeric vector of pupil-teacher ratios per town (constant for all Boston tracts)

**B** A numeric vector of  $1000 \cdot (B_k - 0.63)^2$  where  $B_k$  is the proportion of blacks

**LSTAT** A numeric vector of % less well-off population (notice the original metadata description though...)

In R, the command `lm` with syntax `lm(dependent variable ~ independent variables separated by the PLUS sign)` is used to run a linear regression model on the specified variables. In our case, type the following:

```
OLS <- lm(CMEDV ~ CRIM + CHAS + NOX + RM + DIS + RAD + TAX + PTRATIO + B + LSTAT)
```

The above command will create a new memory object called “OLS” and it will store in this object the results of the specified linear regression. Note that if you simply run `lm(CMEDV ~ CRIM + CHAS + NOX + RM + DIS + RAD + TAX + PTRATIO + B + LSTAT)`, then the results will be displayed temporarily, without the possibility of calling them or processing them at a later stage. So, it is good practice to store the results of any operation into its own object in the memory.

You can check the summary of the OLS regression by typing:

```
summary(OLS)
```

```
Call:
lm(formula = CMEDV ~ CRIM + CHAS + NOX + RM + DIS + RAD + TAX +
    PTRATIO + B + LSTAT)

Residuals:
    Min       1Q   Median       3Q      Max
-15.8778  -2.8156  -0.7443   1.8680  26.6949

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  36.957367   5.075385   7.282 1.31e-12 ***
CRIM         -0.096158   0.032718  -2.939 0.003446 **
CHAS          2.720186   0.856253   3.177 0.001582 **
NOX         -18.608719   3.524104  -5.280 1.93e-07 ***
RM           3.985602   0.402895   9.892 < 2e-16 ***
DIS         -1.196916   0.161236  -7.423 5.02e-13 ***
RAD           0.280896   0.063392   4.431 1.16e-05 ***
TAX          -0.009786   0.003315  -2.952 0.003311 **
PTRATIO     -1.071172   0.121530  -8.814 < 2e-16 ***
B             0.009271   0.002680   3.459 0.000589 ***
LSTAT       -0.527757   0.047536 -11.102 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.747 on 495 degrees of freedom
Multiple R-squared:  0.738,    Adjusted R-squared:  0.7327
F-statistic: 139.4 on 10 and 495 DF,  p-value: < 2.2e-16
```

**4.2** To compute the LM tests for an object in which you stored an OLS regression, you have to use the `lm.LMtests` command with a syntax that can be found through `help(lm.LMtests)`. But now, you need to additionally specify a weights file to be used in the evaluations. In our case, for the object “OLS” that you stored previously, you need to type:

```
LMtests <- lm.LMtests(OLS, weights.rook, test="all")
```

Then, to view the results, call the object you created with a print command (some objects are better reported with the `print` command, some with the `summary` command) - `print(LMtests)` :

```
> print(LMtests)

Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = CMEDV ~ CRIM + CHAS + NOX + RM + DIS + RAD + TAX + PTRATIO + B + LSTAT)
weights: weights

LMerr = 158.15, df = 1, p-value < 2.2e-16

Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = CMEDV ~ CRIM + CHAS + NOX + RM + DIS + RAD + TAX + PTRATIO + B + LSTAT)
weights: weights

LMlag = 79.212, df = 1, p-value < 2.2e-16
```

```
Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = CMEDV ~ CRIM + CHAS + NOX + RM + DIS + RAD + TAX + PTRATIO + B + LSTAT)
weights: weights

RLMerr = 80.179, df = 1, p-value < 2.2e-16

Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = CMEDV ~ CRIM + CHAS + NOX + RM + DIS + RAD + TAX + PTRATIO + B + LSTAT)
weights: weights

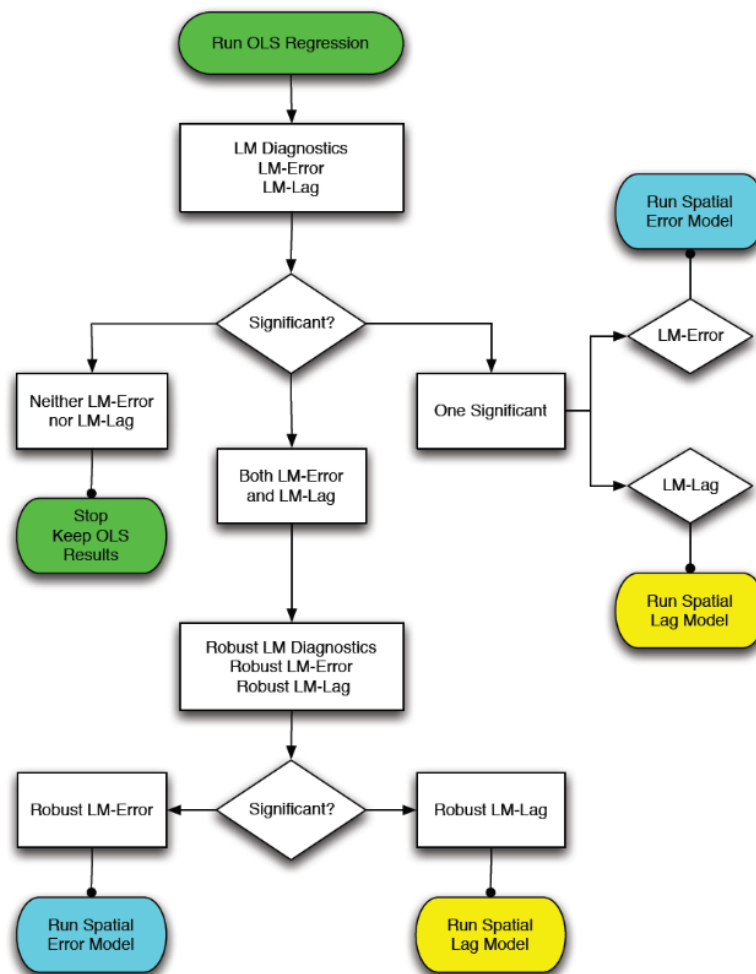
RLMLag = 1.2408, df = 1, p-value = 0.2653

Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = CMEDV ~ CRIM + CHAS + NOX + RM + DIS + RAD + TAX + PTRATIO + B + LSTAT)
weights: weights

SARMA = 159.39, df = 2, p-value < 2.2e-16
```

**4.3** For the interpretation, the best thing to do is to follow Anselin's (2005) workflow:



**4.4** In the case of the LM tests of section 4.2, we can infer that a spatial error regression is the best specification.

## 5 – Spatial error and Spatial Durbin models in R

**5.1** The `spdep` package of R, which we have already been using when importing and transforming weights and when running the LM tests, employs a spatial simultaneous autoregressive error model, by means of maximum likelihood estimation. The model is called by the command `errorsarlm`, with a syntax that you can see by typing `help(errorsarlm)`.

In the case of our dataset, converted weights file, and previously investigated OLS model, the estimation is called by the following syntax. Remember to store the estimation into its own object, so that you retrieve its results and work with them at any time later on.

```
sp.err <- errorsarlm(CMEDV ~ CRIM + CHAS + NOX + RM + DIS + RAD + TAX +
PTRATIO + B + LSTAT, data=boston, weights)
```

Notice the new part `data=boston, weights`. There is also a `method` setting that provides several approximation options for samples that are too large. You can retrieve the estimations with the `summary` command `summary(sp.err)`. Alternatively, for a nicer output, you can run the following:

```
sum.err <- summary(sp.err, Nagelkerke=TRUE)
print(sum.err, signif.stars=TRUE)
```

```
Call:errorsarlm(formula = CMEDV ~ CRIM + CHAS + NOX + RM + DIS + RAD +
TAX + PTRATIO + B + LSTAT, data = boston, listw = weights)
```

```
Residuals:
    Min       1Q   Median       3Q      Max
-19.80138  -2.28990  -0.59735   1.42950  19.88119

Type: error
Coefficients: (asymptotic standard errors)
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  30.7041086   5.3810526   5.7060 1.157e-08 ***
CRIM         -0.1320561   0.0276118  -4.7826 1.730e-06 ***
CHAS         -0.7302708   0.8754379  -0.8342 0.4041809
NOX          -22.4013922   4.9317399  -4.5423 5.565e-06 ***
RM           4.2757836    0.3546737  12.0555 < 2.2e-16 ***
DIS          -1.2559918   0.2797953  -4.4890 7.157e-06 ***
RAD           0.3076608   0.0762306   4.0359 5.439e-05 ***
TAX          -0.0126494   0.0033802  -3.7422 0.0001824 ***
PTRATIO      -0.6949224   0.1442842  -4.8163 1.462e-06 ***
B             0.0096308   0.0030505   3.1571 0.0015932 **
LSTAT        -0.4579763   0.0484935  -9.4441 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Lambda: 0.70301, LR test value: 147.03, p-value: < 2.22e-16
Asymptotic standard error: 0.04082
z-value: 17.222, p-value: < 2.22e-16
Wald statistic: 296.6, p-value: < 2.22e-16

Log likelihood: -1427.053 for error model
ML residual variance (sigma squared): 14.725, (sigma: 3.8373)
Nagelkerke pseudo-R-squared: 0.80405
Number of observations: 506
Number of parameters estimated: 13
AIC: 2880.1, (AIC for lm: 3025.1)
```

**5.2** The spatial error model should be your workhorse. Its motivation and interpretation are straightforward and it is the most compatible with standard policy-based analysis. This means that you can focus on causal effects while at the same time controlling for spatial autocorrelation, in a simple way.



The next step, however, can be the exploitation of the spatial interaction information in your data. One way to do that is to check whether the above spatial error model can be extended into a spatial Durbin model.

**5.3** First, prepare a sparse spatial weights matrix and calculate its traces of powers. The traces are sometimes needed if the sample is large: using the full spatial weights matrix will create various computational problems (computational time or inability to estimate significances, among others):

```
W <- as(as_dgRMatrix_listw(weights), "CsparseMatrix")  
  
tr <- trW(W, type="MC")
```

**5.4** Next, run a spatial Durbin model:

```
sp.durb <- lagsarlm(CMEDV ~ CRIM + CHAS + NOX + RM + DIS + RAD + TAX +  
PTRATIO + B + LSTAT, data=boston, weights, type="mixed", method="MC",  
trs=tr)
```

**5.5** Run the spatial common factor hypothesis test and assign it to a data-frame called "CFH":

```
CFH <- LR.sarlm(sp.durb, sp.err)
```

Print the results by typing `print(CFH)`:

```
      Likelihood ratio for spatial linear models  
  
data:  
Likelihood ratio = 27.23, df = 10, p-value = 0.002395  
sample estimates:  
Log likelihood of sp.durb  Log likelihood of sp.err  
      -1413.439             -1427.053
```

Notice the degrees of freedom for the test (`df = 10`) and the p-value for the test (`p-value = 0.002395` [`< 0.05`]). A significant p-value denotes that you can employ a spatial Durbin model (i.e. you reject the common factor hypothesis).

The degrees of freedom must always be the same as the explanatory variables minus the intercept (the *lagged forms* of the variables *do not count as explanatory variables* for the degrees of freedom since they do not represent genuine variables but are lagged derivations of the original ones). Note that all the Likelihood Ratio tests for the common factor hypothesis employ the  $X^2$  (chi-square) distribution.

**5.6** We have rejected the common factor hypothesis and we can this prefer a spatial Durbin specification over the simpler spatial error specification. The summary is shown below:



```
Call:lagsarlm(formula = CMEDV ~ CRIM + CHAS + NOX + RM + DIS + RAD +  
TAX + PTRATIO + B + LSTAT, data = boston, listw = weights, type = "mixed", method = "MC", trs = tr)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-20.68618	-2.25303	-0.40741	1.55617	20.12499

Type: mixed

Coefficients: (asymptotic standard errors)

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	13.3306105	7.1674143	1.8599	0.0629009
CRIM	-0.1268187	0.0278380	-4.5556	5.224e-06
CHAS	-1.5171938	0.8996429	-1.6864	0.0917111
NOX	-27.0689858	5.9693165	-4.5347	5.769e-06
RM	4.2857361	0.3538629	12.1113	< 2.2e-16
DIS	-0.7236244	0.5322005	-1.3597	0.1739301
RAD	0.2973144	0.0865416	3.4355	0.0005914
TAX	-0.0130750	0.0034808	-3.7563	0.0001724
PTRATIO	-0.5255637	0.1584581	-3.3167	0.0009108
B	0.0100981	0.0032425	3.1143	0.0018441
LSTAT	-0.4489788	0.0507482	-8.8472	< 2.2e-16
lag.CRIM	0.1224338	0.0523023	2.3409	0.0192380
lag.CHAS	5.7146275	1.4761676	3.8713	0.0001083
lag.NOX	20.1509362	7.4611761	2.7008	0.0069179
lag.RM	-2.9118373	0.6907786	-4.2153	2.494e-05
lag.DIS	0.3420404	0.5745476	0.5953	0.5516287
lag.RAD	-0.2809071	0.1180636	-2.3793	0.0173462
lag.TAX	0.0147112	0.0056191	2.6181	0.0088431
lag.PTRATIO	0.0979089	0.2286962	0.4281	0.6685654
lag.B	-0.0068153	0.0046115	-1.4779	0.1394416
lag.LSTAT	0.2281859	0.0876260	2.6041	0.0092119

Rho: 0.63167, LR test value: 124.91, p-value: < 2.22e-16

Asymptotic standard error: 0.045517

z-value: 13.877, p-value: < 2.22e-16

Wald statistic: 192.58, p-value: < 2.22e-16

Log likelihood: -1413.439 for mixed model

ML residual variance (sigma squared): 14.34, (sigma: 3.7868)

Number of observations: 506

Number of parameters estimated: 23

AIC: 2872.9, (AIC for lm: 2995.8)

LM test for residual autocorrelation

test value: 0.29508, p-value: 0.58698

**5.5** Remember, however, that you cannot now interpret the coefficients at their face value; they contain both pure and spatial interaction effects. You will have to simulate the effects into direct, indirect, and total spatial impacts (LeSage 2008):

A simple way:

Impact measures (mixed, exact):

	Direct	Indirect	Total
CRIM	-0.11979857	0.107893932	-0.011904640
CHAS	-0.72834120	12.124089156	11.395747956
NOX	-26.56273535	7.780699517	-18.782035833
RM	4.25178871	-0.521746484	3.730042223
DIS	-0.74270586	-0.293268707	-1.035974571
RAD	0.28187273	-0.237327840	0.044544893
TAX	-0.01200485	0.016447058	0.004442207
PTRATIO	-0.56438583	-0.596667899	-1.161053727
B	0.01002569	-0.001112993	0.008912699
LSTAT	-0.45817035	-0.141267396	-0.599437745

But notice that the significances are missing! A more detailed way:

```
impacts.durb <- impacts(sp.durb, tr=tr, R=100, zstats=TRUE)

summary(impacts.durb, zstats=TRUE)
```

The summary will provide a lot of output – the most important pieces of information are the first and last tables, which give the impact estimates and their significances, respectively:

```
Impact measures (mixed, trace):
      Direct      Indirect      Total
CRIM   -0.11964826  0.107743631 -0.011904628
CHAS   -0.71145057 12.107186737 11.395736169
NOX   -26.55189536  7.769878952 -18.782016407
RM      4.25106178 -0.521023418  3.730038365
DIS    -0.74311442 -0.292859079 -1.035973500
RAD     0.28154210 -0.236997250  0.044544847
TAX    -0.01198194  0.016424140  0.004442202
PTRATIO -0.56521707 -0.595835460 -1.161052527
B       0.01002414 -0.001111451  0.008912690
LSTAT  -0.45836715 -0.141069976 -0.599437125
=====
.....
Simulated p-values:
      Direct      Indirect      Total
CRIM   6.9438e-05 0.30641  0.98915834
CHAS   0.44071280 4.1268e-05 0.00019905
NOX    1.0423e-07 0.37117  0.05185245
RM     < 2.22e-16 0.80621  0.01831899
DIS    0.12143229 0.74673  0.02644906
RAD    3.9008e-05 0.16987  0.95896636
TAX    0.00059086 0.13185  0.61896822
PTRATIO 0.00089247 0.16806  0.00567124
B      0.00086877 0.93270  0.28082180
LSTAT  < 2.22e-16 0.41586  0.00059914
> |
```

What do these results mean?

- You should distinguish between Direct, Indirect, and Total impacts.
- You should realize a variable can have insignificant direct, indirect, or total impacts.
- LeSage (2008) and Le Sage & Pace (2009) define those impacts.
- The mentality is close to multiplier analysis and its flavor is regional or urban policy analysis: what happens to a location when a marginal change happens at the location or somewhere in its neighborhood, or across the spatial system.

## 6 – Spatial lag model in R

**5.1** Assume that we would like to run a spatial lag model, based on the fact that the LM tests were not clear. The `spdep` package can estimate a spatial simultaneous autoregressive lag model via the command `lagsarlm`, and a syntax that is similar to the spatial error model of steps 5.x above:

```
sp.lag <- lagsarlm(CMEDV ~ CRIM + CHAS + NOX + RM + DIS + RAD + TAX +
PTRATIO + B + LSTAT, data=boston, weights)
```

Similar to the spatial Durbin model, the parameter estimates of the spatial lag model are better interpreted through the spatial impacts framework – in this case the impacts are as follows:

Impact measures (lag, trace):

	Direct	Indirect	Total
CRIM	-0.072651345	-0.043101033	-0.11575238
CHAS	1.150559146	0.682579068	1.83313821
NOX	-12.068566048	-7.159780174	-19.22834622
RM	3.993803919	2.369358381	6.36316230
DIS	-1.007476767	-0.597694221	-1.60517099
RAD	0.233678091	0.138631529	0.37230962
TAX	-0.008422843	-0.004996924	-0.01341977
PTRATIO	-0.650282443	-0.385785629	-1.03606807
B	0.008046124	0.004773432	0.01281956
LSTAT	-0.378099479	-0.224310754	-0.60241023

Simulated p-values:

	Direct	Indirect	Total
CRIM	0.02109607	0.02785550	0.0213276
CHAS	0.15473297	0.15649168	0.1516548
NOX	0.00013308	0.00018582	8.7019e-05
RM	< 2.22e-16	3.1985e-07	1.1102e-15
DIS	9.6612e-13	1.7628e-07	4.8501e-12
RAD	8.4924e-05	0.00048457	8.9900e-05
TAX	0.00346946	0.01154901	0.0046975
PTRATIO	4.3467e-08	8.5947e-07	1.3466e-08
B	0.00166697	0.00398160	0.0018635
LSTAT	3.3307e-15	8.0250e-11	< 2.22e-16

## 7 – Utilizing a SAC/SARMA model

**7.1** You can always employ the SARMA output of the Lagrange Multiplier tests. If the LM test for SARMA turns out significant, then you can run a variant of such higher-order mixed models by using the `sacsarlm` command of `spdep`.

**7.2** Let's first import and convert three alternative knn weights files:

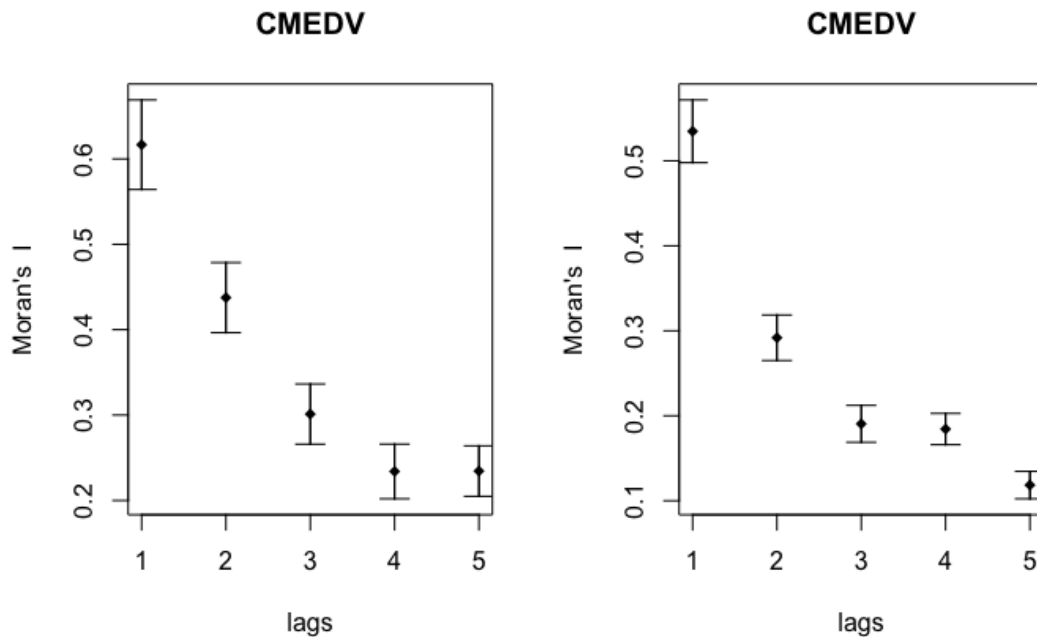
```
geoda_5nn <- read.gwt2nb("boston_5nn.gwt", region.id=POLY_ID)
geoda_10nn <- read.gwt2nb("boston_10nn.gwt", region.id=POLY_ID)

weights.5nn <- nb2listw(geoda_5nn)
weights.10nn <- nb2listw(geoda_10nn)
```

**7.3** You can use a spatial correlogram to get a sense of the lag order that is relevant for your dependent variable. This feature is far less utilized in relation to time series econometrics, but it can provide useful hints:

```
plot(sp.correlogram(geoda_5nn, CMEDV, order=5, method = "I", zero.policy = T))
plot(sp.correlogram(geoda_10nn, CMEDV, order=5, method = "I", zero.policy = T))
```

The resulting correlograms:



### 7.3 We can run a SAC model, but be cautious:

1. You notice that the model estimates both the  $\rho$  and  $\lambda$  coefficients. The problem is that in the optimization procedure of finding the values of  $\rho$  and  $\lambda$  there is more than one optimum. This means that we are not sure whether this particular pair of values for the coefficients is the correct one. The `sacsarlm` command does offer some functionality on selecting the correct optimum, but the uncertainties remain at large.

2. However, the final blow is given by the following question: you use a spatial weights matrix  $\mathbf{W}$ ; how do you know that  $\rho \cdot \mathbf{y} \cdot \mathbf{W}$  is not  $\lambda \cdot \mathbf{u} \cdot \mathbf{W}$  and vice versa? You can employ two different weights matrices to avoid this problem, but then you need to justify the particular neighbourhood conceptualizations for the spatial error and lag terms.

**7.4** To ameliorate the above problems, let's use the 5-nearest-neighbors weights file for the lag ( $\rho$ ) coefficient and the 10-nearest-neighbors weights file for the error ( $\lambda$ ) coefficient. In this case, the syntax will be as follows:

```
SAC <- sacsarlm(CMEDV ~ CRIM + CHAS + NOX + RM + DIS + RAD + TAX +
PTRATIO + B + LSTAT, data=boston, weights.5nn, weights.10nn)
```

## 8 – Before ending a session in R (or before exiting R altogether)

- Make the script editor window active. Go to `File > Save as`
- Save with a name of your liking
- You can also save a workspace image by `Session > Save Workspace As...` This saves all the imported data, dataframes, computed objects and results, making it easy to pick up the work from where you stopped last time. Storing objects can become useful when you have complicated calculations that you do not want to run again. Saving just the script file is much more important, and you can run it again, generate the objects again, and continue with further analysis.