

# An Introduction to Spatial Regression Analysis in R

Luc Anselin

University of Illinois, Urbana-Champaign

<http://sal.agecon.uiuc.edu>

May 23, 2003

## Introduction

This note contains a brief introduction and tutorial on the spatial regression functionality contained in the R package “spdep” developed by Roger Bivand and collaborators. No previous knowledge of R is assumed, although the more you know, the more you can exploit the powerful features of spdep and customize it to your needs. Detailed information on the data and functions in the spdep package is contained in the spdep help file on which much of these notes are based. To get started with R, the document “An Introduction to R” by Venables, Smith et al is highly recommended. The specific illustrations will be for a Windows platform, although it should be noted that R is cross-platform and runs equally well on Unix/Linux and MacOS.

## Preliminaries

This note uses the latest version of R, R1.7.0. You can download the “base” installation from <http://cran.r-project.org/>. Specifically, download the file rw1070.exe and double click to go through the installation procedure. This will install the R program as well as a basic set of packages. You will also need the spdep package. The latest version of spdep is 0.1-10 (4/29/03). Either download the binaries directly (as the spdep.zip file from the “contributed” directory) or download the source, index and pdf manual from the CRAN package sources directory.

Note: CRAN does not always have the latest version of spdep as a binary for windows. You can always find that at <http://spatial.nhh.no/R/spdep/> (don't forget the last slash) as well as some other late breaking stuff in the archive main page <http://spatial.nhh.no/R/> .

In Windows, the easiest way to start R is to create a working directory and change the properties of the R shortcut icon to start in that directory. For example, first right click the R icon and select Properties, as in Figure 1. Next enter you working directory as the “Start In” property. In Figure 2, this is set to C:\Stats\work, but you should specify the full pathname for the working directory on your system. Put any file that needs to be read in by R, or any file with R code to run (“source”) in this directory to minimize problems with unrecognized paths.

## Installing the Package

You can either download the spdep.zip file, or install the package directly from within R. Double click on the R icon to start the program. In the Packages menu, select:

```
Packages > Install package from CRAN
```

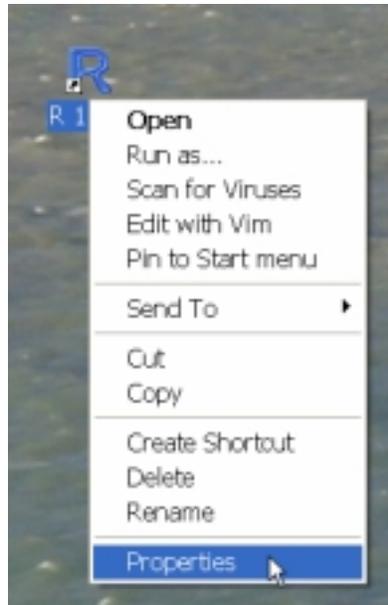


Figure 1. R Properties

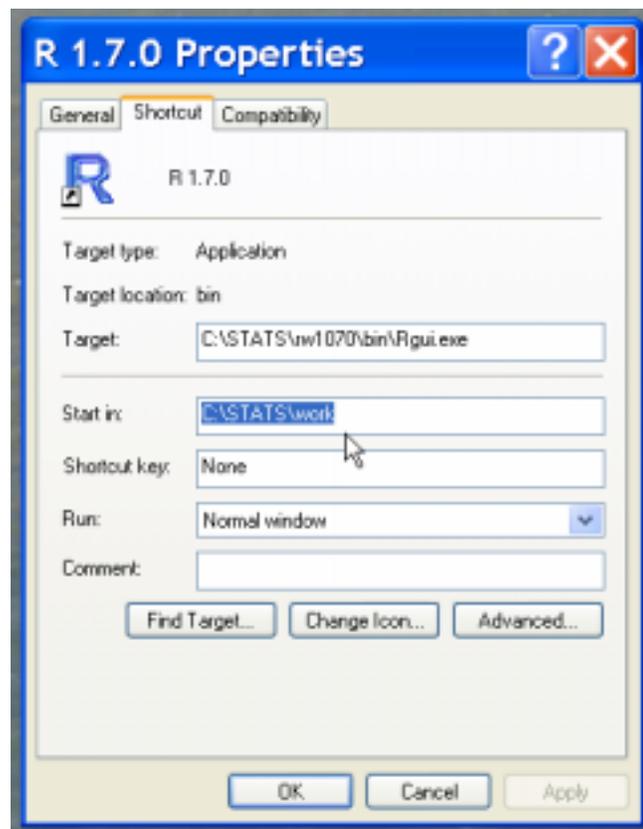


Figure 2. Starting Directory.

select `spdep` from the list, R takes care of the rest. Alternatively, if you have the `spdep.zip` file downloaded (with the latest version), use:

```
Packages > Install package from local zip file
```

Use the open file dialog to locate `spdep.zip`. Finally, make sure you have the latest version, either manually or using the menu:

```
Packages > Update packages from CRAN
```

If you also want the source code to the R and C functions used in the package, download the file `spdep.0.1-10.tar.gz` from the package source archive and extract the files in the R and src directories. Be careful when you extract the R source files since the binary `spdep` file will be overwritten by the source file. Make sure the binary is there (if necessary, reinstall from `spdep.zip`, the source files won't be touched by this procedure).

### Launching `spdep`

To access the functionality and data sets in `spdep`, it must be “loaded”. In the windows menu, use:

```
Packages > Load Package
```

and select `spdep` from the list, or from the command line:

```
>library(spdep)
```

Note, **do not use** `>load(spdep)` even though it may seem intuitive.

Ignore the warning (`Spatial object identity integrity check: FALSE`).

If all goes well, the functions and sample data sets contained in the package are now available. To make sure everything works as advertised, enter

```
>data(columbus)
>summary(columbus)
```

If you get a series of summary descriptive statistics for the variables in the Columbus sample data set, then everything is there. If you get an error message, such as

```
>Error in summary(columbus) : Object "columbus" not found
```

then the package was not installed or loaded properly. Make sure all the files are where they are supposed to be, or start over.

## Sample Data Sets

The `spdep` package contains several sample data sets that have the necessary “spatial” information (weights files, coordinates, boundary files) to carry out spatial regression analysis.

The sample data sets are (note the data set names are case sensitive):

- `oldcol`: Columbus crime data from Anselin (1988) book
- `afcon`: African conflict data set from Anselin and O’Loughlin (1988) and Anselin (1995)
- `auckland`: Auckland infant mortality data set used in Bailey and Gatrell (1985)
- `boston`: data set on Boston house prices originally used by Harrison and Rubinfeld (1978) and in spatial models (coordinates added) by Pace et al. (1997)
- `columbus`: Columbus crime data set from Anselin’s web site (same as `oldcol`, but with the observations in a different order)
- `eire`: Irish county data set from Cliff and Ord (1981) and many others
- `getisord`: grid cell remote sensing data from Getis and Ord (1996)
- `nc.sids`: North Carolina county SIDS deaths from Cressie (1993)
- `used.cars`: used car prices for 48 US states used in Hepple (1976).

The tutorial will use the `columbus` data for illustrations and the `boston` data set for practice. Feel free to experiment with the other data sets as well, particularly `afcon`, `eire` and `used.cars`, which have been used to illustrate the properties of spatial regression techniques in a number of publications (see the references in the `spdep` manual).

To make a sample data set active and its contents available, start with

```
>data(datasetname)
```

For example, to load the Columbus sample data set in `spdep` (after the `spdep` library is activated), enter:

```
>data(columbus)
```

The characteristics of the data set can be checked with a number of R commands:

```
>columbus
```

prints the contents as a data frame, with an ID for each row and the variable name as the column header; same as

```
>print(columbus)
```

Alternatively,

```
>summary(columbus)
```

gives basic descriptive statistics for each variable in the data frame (min, 1<sup>st</sup> quartile, median, mean, 3<sup>rd</sup> quartile, max)

Once `columbus` is loaded with the `data(columbus)` command, five “objects” become available – show available objects using `>objects()` or `>ls()` – not just the data frame:

- `columbus`: the data frame, contains 22 variables
- `col.gal.nb`: is an object of class “nb”, a list of vectors, one for each spatial unit, and containing the sequence numbers of the neighbors (this contiguity file uses the queen definition for Columbus)
- `coords`: centroid coordinates that can be used to construct distance-based weights
- `polys` and `bbs`: the boundary file and bounding boxes for the polygons, used to construct contiguity information

### *Practice*

Load the `boston` data set and check what objects have been added to your work space. Check some basic descriptive statistics with the `summary` command. If it did not work, check the contents of your work space and note the “surprising” name for the data frame (the data set) with the `boston` data. Use this name in the `summary` command and it should work. It is good practice to check this with `ls()` since the name of the data set and the data frame need not be the same.

## **Running a Regression**

R consists of expressions that operate on objects. This interactive approach usually requires you to enter a series of expressions where each creates the results of an analysis as a new object constructed from an existing object. For example, a linear regression is a function that creates an object. Written as a function, it simply prints out some results. Written as an assignment to a new object it seems to do nothing, but you can then print, summarize or otherwise manipulate the new object.

For example, to run the “classic” Columbus crime regression (of `CRIME` on `INC` and `HOVAL`) you need the “linear model” function, `lm`. You specify the “model” as:  $y \sim x_1 + x_2 + x_3$ , etc., where  $y$  is the dependent variable and the  $x$  are the explanatory variables. You also should specify the data set (there is a default, but it could give you surprising results, it’s good practice to force yourself to specify the data set). If you enter `lm` without an assignment, you get a summary print out, as in:

```
>lm(CRIME ~ INC + HOVAL, data=columbus)
```

The result is:

Call:

```
lm(formula = CRIME ~ INC + HOVAL, data = columbus)
```

Coefficients:

```
(Intercept)          INC          HOVAL
    68.6190       -1.5973       -0.2739
```

In contrast, if you use `lm` with an assignment, you create a new object, but there is no printout. For example, assign the result to the object `columbus.lm`, as in:

```
>columbus.lm <- lm( CRIME ~ INC + HOVAL, data=columbus)
```

Nothing happens... However, you did create a new object. To find out what this object contains, summarize it:

```
>summary(columbus.lm)
```

and the detailed results appear

Call:

```
lm(formula = CRIME ~ INC + HOVAL, data = columbus)
```

Residuals:

```
      Min       1Q   Median       3Q      Max
-34.418  -6.388  -1.580    9.052   28.649
```

Coefficients:

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   68.6190     4.7355  14.490 < 2e-16 ***
INC           -1.5973     0.3341  -4.780 1.83e-05 ***
HOVAL         -0.2739     0.1032  -2.654 0.0109 *
```

---

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 11.43 on 46 degrees of freedom

Multiple R-Squared: 0.5524, Adjusted R-squared: 0.5329

F-statistic: 28.39 on 2 and 46 DF, p-value: 9.34e-09

Note that this contains much more information than the printout without the assignment. Within R, many functions have a default set of methods, such as `print`, `plot` or `summary` that are customized to provide the appropriate information.

### *Practice*

Run a basic regression of the housing values in the Boston data set. The “classic” specification is  $\log(\text{MEDV})$  or  $\log(\text{CMEDV})$  (median house value) on a set of hedonic characteristics, such as `CRIM` (crime), `ZN` (large lot zoning as a proportion), `INDUS` (non-retail business acres per town), `CHAS` (Charles river dummy), `NOX2` (nitric oxides), `RM2` (average number of rooms), `AGE` (proportion pre 1940), `log(DIS)` (log of weighted distance to employment centers), `log(RAD)` (accessibility index), `TAX` (property tax rate), `PTRATIO` (pupil teacher ratio), `B` (black population) and `log(LSTAT)` (lower status population). For details, see the `spdep` manual and the references therein. Also, note that to use the powers in the regression “formula,” you must enclose them in an “identity” `I()` command, for example, as `I(NOX2)`. Make sure to create an `lm` object, say `bo1`. Compare the results for the original median value dependent variable (`MEDV`) and the “corrected” one (`CMEDV`).

## Spatial Weights

The Moran I test statistic for spatial autocorrelation applied to regression residuals is implemented by the function `lm.morantest`. This function operates on an `lm` object and requires that a spatial weights file be specified. The `spdep` package has several different objects to store contiguity information. It is important to know which type is appropriate for which function. The `lm.morantest` (and all other tests for spatial correlation) requires an `listw` object. This is not the same as the `nb` object included in the sample data set. The `listw` object also explicitly contains values for the spatial weights, whereas the `nb` object only has the identifiers for the neighbors. You need to convert a `nb` object to a `listw` object by means of the `nb2listw` function. For example:

```
> col.listw <- nb2listw(col.gal.nb)
```

Note the difference between the two when you do a print statement. For `col.gal.nb`:

```
> print(col.gal.nb)
[[1]]
[1] 2 3

[[2]]
[1] 1 3 4

...

[[49]]
[1] 44 45 48

attr(,"class")
[1] "nb"
attr(,"region.id")
 [1] 1005 1001 1006 1002 1007 1008 1004 1003 1018 1010 1038 1037 1039
1040 1009
[16] 1036 1011 1042 1041 1017 1043 1019 1012 1035 1032 1020 1021 1031
1033 1034
[31] 1045 1013 1022 1044 1023 1046 1030 1024 1047 1016 1014 1049 1029
1025 1028
[46] 1048 1015 1027 1026
attr(,"gal")
[1] TRUE
attr(,"call")
[1] TRUE
attr(,"sym")
[1] TRUE
```

This is a list of vectors, with each element of the list shown in the `[[ ]]` notation. For example, the first element `[[1]]` contains an integer vector with elements 2 and 3, the identifiers for the neighbors of 1. After the list of neighbors comes information on the “attributes” of the object, the class (`nb`), labels for the neighborhoods (`region.id`) and some technical attributes used internally. In contrast the print command for a `listw` object, `print(col.listw)` yields:

```

$style
[1] "W"

$neighbours
[[1]]
[1] 2 3

[[2]]
[1] 1 3 4

[[3]]
[1] 1 2 4 5

...

[[49]]
[1] 44 45 48

attr(,"class")
[1] "nb"
attr(,"region.id")
 [1] 1005 1001 1006 1002 1007 1008 1004 1003 1018 1010 1038 1037 1039
1040 1009
[16] 1036 1011 1042 1041 1017 1043 1019 1012 1035 1032 1020 1021 1031
1033 1034
[31] 1045 1013 1022 1044 1023 1046 1030 1024 1047 1016 1014 1049 1029
1025 1028
[46] 1048 1015 1027 1026
attr(,"gal")
[1] TRUE
attr(,"call")
[1] TRUE
attr(,"sym")
[1] TRUE

$weights
$weights[[1]]
[1] 0.5 0.5

$weights[[2]]
[1] 0.3333333 0.3333333 0.3333333

...

$weights[[49]]
[1] 0.3333333 0.3333333 0.3333333

attr(,"binary")
[1] TRUE
attr(,"W")
[1] TRUE

attr(,"class")
[1] "listw" "nb"
attr(,"region.id")

```

```

[1] 1005 1001 1006 1002 1007 1008 1004 1003 1018 1010 1038 1037 1039
1040 1009
[16] 1036 1011 1042 1041 1017 1043 1019 1012 1035 1032 1020 1021 1031
1033 1034
[31] 1045 1013 1022 1044 1023 1046 1030 1024 1047 1016 1014 1049 1029
1025 1028
[46] 1048 1015 1027 1026
attr(,"call")
nb2listw(neighbours = col.gal.nb)

```

In addition to the neighbor list, this also includes specific values for the spatial weights (as well as some additional internal attributes). Similarly, the summary function is only defined for nb objects:

```

> summary(col.gal.nb)
Connectivity of col.gal.nb with the following attributes:
List of 5
 $ class      : chr "nb"
 $ region.id : num [1:49] 1005 1001 1006 1002 1007 ...
 $ gal       : logi TRUE
 $ call      : logi TRUE
 $ sym       : logi TRUE
NULL
Number of regions: 49
Number of nonzero links: 230
Percentage nonzero weights: 9.579342
Average number of links: 4.693878
Link number distribution:

 2  3  4  5  6  7  8  9 10
 7  7 13  4  9  6  1  1  1
7 least connected regions:
1005 1008 1045 1047 1049 1048 1015 with 2 links
1 most connected region:
1017 with 10 links

```

**In contrast with**

```

> summary(col.listw)
Error in summary.nb(col.listw) : Not a neighbours list
In addition: Warning message:
the condition has length > 1 and only the first element will be used
in: if (class(nb) != "nb") stop("Not a neighbours list")
>

```

### *Practice*

Check the contiguity information in the boston soi file. Convert the nb object to a listw object. Compare print and summary for these objects.

## Moran's I Test for Residual Spatial Autocorrelation

To carry out a Moran test on the residuals of the `columbus.lm` regression object, pass both the regression object (`columbus.lm`) and the `listw` contiguity information to the `lm.morantest` function. For example:

```
> col.moran <- lm.morantest(columbus.lm,col.listw)
> summary(col.moran)
      Length Class      Mode
statistic  1      -none-  numeric
p.value    1      -none-  numeric
estimate   3      -none-  numeric
method     1      -none-  character
alternative 1      -none-  character
data.name  1      -none-  character
> print(col.moran)
```

Global Moran's I for regression residuals

data:

```
model: lm(formula = CRIME ~ INC + HOVAL, data = columbus)
weights: col.listw
```

Moran I statistic standard deviate = 2.681, p-value = 0.00367

alternative hypothesis: greater

sample estimates:

Observed Moran's I	Expectation	Variance
0.212374153	-0.033268284	0.008394853

The result, `col.moran` is an object of class `htest`, which has its specialized “print” function (but note how “summary” does not quite do what you might have expected). One aspect to note about these results is that the default significance is reported for a one-sided alternative (“greater”). For a two-sided alternative hypothesis, the significance would be twice the reported value. You can specify this with the “alternative” option, as in:

```
> col.moran2 <-
lm.morantest(columbus.lm,col.listw,alternative="two.sided")
> print(col.moran2)
```

Global Moran's I for regression residuals

data:

```
model: lm(formula = CRIME ~ INC + HOVAL, data = columbus)
weights: col.listw
```

Moran I statistic standard deviate = 2.681, p-value = 0.00734

alternative hypothesis: two.sided

sample estimates:

Observed Moran's I	Expectation	Variance
0.212374153	-0.033268284	0.008394853

Check the `spdep` manual for details on other options that can be specified.

### *Practice*

Test for spatial residual autocorrelation in the Boston regression using `boston.soi` as the `nb` object.

## **What Not To Do To Test Residual Spatial Autocorrelation**

It is important to realize that the Moran's I test statistic for residual spatial autocorrelation takes into account the fact that the variable under consideration is a residual, computed from a regression. The usual Moran's I test statistic **does not**. It is therefore incorrect to simply apply a Moran's I test to the residuals from the regression without correcting for the fact that these are residuals. For example, you can extract the residuals from `columbus.lm` by means of the `resid( )` function and apply the `moran.test` function (note the spelling of "randomisation"):

```
> col.e <- resid(columbus.lm)
> col.morane <- moran.test(col.e,col.listw,
+ randomisation=FALSE,alternative="two.sided")
> print(col.morane)

Moran's I test under normality

data: col.e
weights: col.listw

Moran I statistic standard deviate = 2.4774, p-value = 0.01323
alternative hypothesis: two.sided
sample estimates:
Moran I statistic      Expectation      Variance
0.212374153          -0.020833333          0.008860962
```

The specific options used here are to take a normal approximation (the default is `randomisation = TRUE`, which implements a permutation test) and to specify a two sided alternative (default is one-sided). Compare the standardized deviate and the p-values to those obtained with the correct test `lm.morantest` above.

### *Practice*

Extract the residuals from the boston regression and compare a "naïve" Moran's I test to the correct test for residual spatial autocorrelation.

## **Lagrange Multiplier Test Statistics for Spatial Autocorrelation**

The Moran's I test statistic has high power against a range of spatial alternatives. However, it does not provide much help in terms of which alternative model would be most appropriate. The Lagrange Multiplier test statistics do allow a distinction between spatial error models and spatial lag models.

Both tests, as well as their robust forms are included in the `lm.LMtests` function. Again, a regression object and a spatial listw object must be passed as arguments. In addition,

the tests must be specified as a character vector (the default is only LMerror), using the `c( )` operator (concatenate), as illustrated below.

```
> columbus.lagrange <- lm.LMtests(columbus.lm,col.listw,
+ test=c("LMerr","RLMerr","LMlag","RLMlag","SARMA"))
> print(columbus.lagrange)

      Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = CRIME ~ INC + HOVAL, data = columbus)
weights: col.listw

LMerr = 4.6111, df = 1, p-value = 0.03177

      Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = CRIME ~ INC + HOVAL, data = columbus)
weights: col.listw

RLMerr = 0.0335, df = 1, p-value = 0.8547

      Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = CRIME ~ INC + HOVAL, data = columbus)
weights: col.listw

LMlag = 7.8557, df = 1, p-value = 0.005066

      Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = CRIME ~ INC + HOVAL, data = columbus)
weights: col.listw

RLMlag = 3.2781, df = 1, p-value = 0.07021

      Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = CRIME ~ INC + HOVAL, data = columbus)
weights: col.listw

SARMA = 7.8892, df = 2, p-value = 0.01936
```

In a specification search, you would first assess which of the LMerr and LMlag are significant. In this case, both are, which necessitates the comparison of the robust forms. Here, this suggests the lag model as the likely alternative ( $p = 0.07$  compared to  $p = 0.85$ ).

### *Practice*

Test for spatial autocorrelation in the boston regression and suggest the likely alternative model. You can experiment with different model specifications to try and eliminate spatial correlation by including different variables (you could try this with the Columbus data as well, there are in fact a few specifications where there is no remaining spatial residual correlation).

### **Maximum Likelihood Estimation of the Spatial Lag Model**

Maximum Likelihood (ML) estimation of the spatial lag model is carried out with the `lagsarlm()` function. The required arguments are a regression “formula”, a data set and a listw spatial weights object. The default method uses Ord’s eigenvalue decomposition of the spatial weights matrix. There is also a sparse weights method available (but see the cautionary note in the `spdep` manual).

```
> columbus.lag <- lagsarlm(CRIME ~ INC + HOVAL,data=columbus,
+ col.listw)
```

```
> summary(columbus.lag)
```

Call:

```
lagsarlm(formula = CRIME ~ INC + HOVAL, data = columbus, listw =
col.listw)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-37.4497093	-5.4565567	0.0016387	6.7159553	24.7107978

Type: lag

Coefficients: (asymptotic standard errors)

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	46.851431	7.314754	6.4051	1.503e-10
INC	-1.073533	0.310872	-3.4533	0.0005538
HOVAL	-0.269997	0.090128	-2.9957	0.0027381

Rho: 0.40389 LR test value: 8.4179 p-value: 0.0037154

Asymptotic standard error: 0.12071 z-value: 3.3459 p-value: 0.00082027

Log likelihood: -183.1683 for lag model

ML residual variance (sigma squared): 99.164, (sigma: 9.9581)

Number of observations: 49

Number of parameters estimated: 4

AIC: 374.34, (AIC for lm: 382.75)

LM test for residual autocorrelation

test value: 0.19184 p-value: 0.66139

As expected, the spatial autoregressive parameter ( $\rho$ ) is highly significant, as indicated by the p-value of 0.0008 on an asymptotic t-test (based on the asymptotic variance matrix). The Likelihood Ratio test (LR) on this parameter is also highly significant (p-value 0.0037).

For comparison purposes, estimate this model with the inconsistent OLS estimator. First, you need to construct a spatially lagged dependent variable. To make the individual variables available as “objects” for direct manipulation (the simplest way to proceed), you must first “attach” the data set. Next construct a spatially lagged dependent variable using the `lag.listw` function (specify a listw spatial weights object and a variable/vector). Finally, include the spatial lag as an explanatory variable in the OLS regression. Note how the estimate and standard error of the coefficient for `lagCRIME` is quite different from the ML result.

```
> attach(columbus)
> lagCRIME <- lag.listw(col.listw,CRIME)
> wrong.lag <- lm(CRIME ~ lagCRIME + INC + HOVAL)
> summary(wrong.lag)
```

Call:

```
lm(formula = CRIME ~ lagCRIME + INC + HOVAL)
```

Residuals:

```
      Min       1Q   Median       3Q      Max
-38.3932  -6.5409  -0.1902   6.7995  23.4854
```

Coefficients:

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  40.07773     9.43653   4.247 0.000107 ***
lagCRIME      0.52957     0.15612   3.392 0.001454 **
INC          -0.91054     0.36314  -2.507 0.015840 *
HOVAL        -0.26877     0.09312  -2.886 0.005970 **
```

---

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 10.32 on 45 degrees of freedom

Multiple R-Squared: 0.6436, Adjusted R-squared: 0.6198

F-statistic: 27.08 on 3 and 45 DF, p-value: 3.672e-10

After you run these analyses, don’t forget to `detach(columbus)` or you might run into memory problems. Once you detach a data set, its variables are no longer directly available by name, but must be specified as columns in a data frame, e.g., as `columbus$CRIME` (see the Introduction to R guide for further details).

### *Practice*

Estimate a spatial lag model for the boston example using both OLS and ML and compare the results. Comment on the extent to which spatial effects have been taken care of. Experiment with the sparse weights approach (`method=“sparse”`).

## **Maximum Likelihood Estimation of the Spatial Error Model**

ML estimation of the spatial error model is similar to the lag procedure and implemented in the `errorsarlm` function. Again, the formula, data set and a listw spatial weights object must be specified, as illustrated below.

```

> columbus.err <- errorsarlm(CRIME ~ INC + HOVAL,data=columbus,
+ col.listw)

> summary(columbus.err)

Call:
errorsarlm(formula = CRIME ~ INC + HOVAL, data = columbus, listw =
col.listw)

Residuals:
      Min       1Q   Median       3Q      Max
-34.45950  -6.21730  -0.69775   7.65256  24.23631

Type: error
Coefficients: (asymptotic standard errors)
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  61.053618   5.314875  11.4873 < 2.2e-16
INC          -0.995473   0.337025  -2.9537 0.0031398
HOVAL        -0.307979   0.092584  -3.3265 0.0008794

Lambda: 0.52089 LR test value: 6.4441 p-value: 0.011132
Asymptotic standard error: 0.14129 z-value: 3.6868 p-value: 0.00022713

Log likelihood: -184.1552 for error model
ML residual variance (sigma squared): 99.98, (sigma: 9.999)
Number of observations: 49
Number of parameters estimated: 4
AIC: 376.31, (AIC for lm: 382.75)

```

Note how the estimate for the spatial autoregressive coefficient  $\lambda$  is significant, but less so than for the spatial lag model. A direct comparison between the two models can be based on the maximized log-likelihood (**not** on measures of fit such as  $R^2$ ). In this example, the log-likelihood for the error model of -184.16 is worse (less than) the one for the lag model of -183.17. The actual values are not that meaningful and the difference between the likelihoods cannot be compared more formally in a LR test because the two models are non-nested (that is, one cannot be obtained from the other by setting coefficients or functions of coefficients to zero). More specifically, one **should not** use the `LR.sarlm` method in `spdep` for such a comparison, this is only valid for nested hypotheses, as in the common factor hypothesis discussed next.

### *Practice*

Estimate a spatial error model for the boston data. Compare the coefficient estimates and their standard errors to the OLS results and to the results for the spatial lag model. How does the model fit (log likelihood) compare?

## **Spatial Durbin Model**

An important specification test in the spatial error model is a test on the so-called spatial common factor hypothesis. This exploits the property that a spatial error model can also be specified in spatial lag form, with the spatially lagged explanatory variables included, but with constraints on the parameters (the common factor constraints). The spatial lag

form of the error model is also referred to as the spatial Durbin specification. In `spdep`, this is implemented by specifying the “mixed” option for `type` in the spatial lag estimation (the spatially lagged explanatory variables need not be specified). For the Columbus example, this yields:

```
> columbus.durbin <- lagsarlm(CRIME ~ INC+HOVAL,data=columbus,
+ col.listw,type="mixed")

> summary(columbus.durbin)

Call:
lagsarlm(formula = CRIME ~ INC + HOVAL, data = columbus, listw =
col.listw, type = "mixed")

Residuals:
      Min       1Q   Median       3Q      Max
-37.15904  -6.62594  -0.39823   6.57561  23.62757

Type: mixed
Coefficients: (asymptotic standard errors)
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  45.592894  13.128679   3.4728 0.0005151
INC          -0.939088   0.338229  -2.7765 0.0054950
HOVAL        -0.299605   0.090843  -3.2980 0.0009736
lag.INC      -0.618375   0.577052  -1.0716 0.2838954
lag.HOVAL     0.266615   0.183971   1.4492 0.1472760

Rho: 0.38251 LR test value: 4.1648 p-value: 0.041272
Asymptotic standard error: 0.16237 z-value: 2.3557 p-value: 0.018488

Log likelihood: -182.0161 for mixed model
ML residual variance (sigma squared): 95.051, (sigma: 9.7494)
Number of observations: 49
Number of parameters estimated: 6
AIC: 376.03, (AIC for lm: 380.2)
LM test for residual autocorrelation
test value: 0.101 p-value: 0.75063
```

A test on the common factor hypothesis is a test on the constraints that each coefficient of a spatially lagged explanatory variable (e.g., `lag.INC`) equals the negative of the product of the spatial autoregressive coefficient (`Rho`) and the matching regression coefficient of the unlagged variable (`INC`). In `spdep`, this can be illustrated by means of the `LR.sarlm` function. This is a simple likelihood ratio test, comparing two objects for which a `logLIK` function exists. In our example, we have `columbus.durbin` for the “unconstrained” model (the model with no restrictions on the parameters) and `columbus.err` for the “constrained” model (the constraints have been enforced). The fit (log-likelihood) of the unconstrained model will always be greater than that of the constrained model. A likelihood ratio test is:

```
> durbin.test1 <- LR.sarlm(columbus.durbin,columbus.err)
> print(durbin.test1)
```

Likelihood ratio for spatial linear models

```
data:
Likelihood ratio = 4.2782, df = 1, p-value = 0.03860
sample estimates:
Log likelihood of columbus.durbin      Log likelihood of columbus.err
                -182.0161                      -184.1552
```

This would suggest that the common factor hypothesis is rejected with a p-value of 0.038. However, this is **incorrect** since the proper number for the degrees of freedom is not 1 (as implemented in LR.sarlm), but 2 (the number of nonlinear constraints equals the number of explanatory variables, excluding the constant). To get the proper p-value, you need to delve into some R technicalities. First, the brute force way, by typing the value for the LR into a pchisq function (the cumulative density function for a chi square) with the correct degrees of freedom:

```
> 1 - pchisq(4.2782,2)
[1] 0.1177608
```

With the correct degrees of freedom, there is much weaker evidence to reject the null hypothesis. A little more elegantly, by extracting the value of the statistic from the htest object created by LR.sarlm

```
> 1 - pchisq(durbin.test1[[1]][1],2)
Likelihood ratio
      0.1177622
```

The slight difference between the two results is due to rounding (the second approach is more precise).

### *Practice*

Check the spatial common factor hypothesis for the boston model.

## Appendix A

Answers to practice puzzles.

### *Boston Data Set*

```
> data(boston)
> ls( )
```

gives (your results may be slightly different)

```
> ls()
[1] "bbs"          "boston.c"      "boston.soi"    "boston.utm"
"col.gal.nb"
[6] "columbus"     "columbus.lm"  "coords"        "polys"
```

the data frame is `boston.c`, the census tract X and Y (as a matrix) are `boston.utm`, and `boston.soi` is a neighbor list of class `nb`.

```
> summary(boston)
Error in summary(boston) : Object "boston" not found
> summary(boston.c)
      TOWN      TOWNNO      TRACT
Cambridge   : 30   Min.    : 0.00   Min.    : 1
Boston Savin Hill: 23 1st Qu.:26.25 1st Qu.:1303
Lynn        : 22   Median :42.00  Median :3394
Boston Roxbury : 19   Mean   :47.53  Mean   :2700
Newton      : 18   3rd Qu.:78.00 3rd Qu.:3740
Somerville  : 15   Max.    :91.00  Max.    :5082
(Other)     :379
```

### *Boston OLS Regression*

For example

```
> bo1 <- lm(log(MEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) + I(RM^2) +
+ AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
+ data=boston.c)
> summary(bo1)
```

```
Call:
lm(formula = log(MEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +
    I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B +
    log(LSTAT), data = boston.c)
```

```
Residuals:
    Min       1Q   Median       3Q      Max
-0.71176 -0.09169 -0.00566  0.09895  0.79780
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.558e+00  1.544e-01  29.512 < 2e-16 ***
CRIM        -1.186e-02  1.245e-03  -9.532 < 2e-16 ***
```

```

ZN          8.016e-05  5.056e-04  0.159 0.874105
INDUS       2.395e-04  2.364e-03  0.101 0.919318
CHAS1       9.140e-02  3.320e-02  2.753 0.006129 **
I(NOX^2)   -6.380e-01  1.131e-01  -5.639 2.88e-08 ***
I(RM^2)     6.328e-03  1.312e-03  4.823 1.89e-06 ***
AGE         9.074e-05  5.263e-04  0.172 0.863179
log(DIS)    -1.913e-01  3.339e-02  -5.727 1.78e-08 ***
log(RAD)    9.571e-02  1.913e-02  5.002 7.91e-07 ***
TAX         -4.203e-04  1.227e-04  -3.426 0.000664 ***
PTRATIO     -3.112e-02  5.013e-03  -6.208 1.14e-09 ***
B           3.637e-04  1.031e-04  3.527 0.000460 ***
log(LSTAT)  -3.712e-01  2.501e-02 -14.841 < 2e-16 ***

```

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1825 on 492 degrees of freedom  
Multiple R-Squared: 0.8059, Adjusted R-squared: 0.8008  
F-statistic: 157.1 on 13 and 492 DF, p-value: < 2.2e-16

### ***Boston Neighborhood Information***

```

> summary(boston.soi)
Connectivity of boston.soi with the following attributes:
List of 4
 $ region.id: chr [1:506] "1" "2" "3" "4" ...
 $ call      : language soi.graph(tri.nb = tri2nb(boston.utm), coords =
boston.utm)
 $ class     : chr "nb"
 $ sym       : logi TRUE
NULL
Number of regions: 506
Number of nonzero links: 2152
Percentage nonzero weights: 0.8405068
Average number of links: 4.252964
Link number distribution:

  1  2  3  4  5  6  7  8
10 39 93 143 143 52 22  4
10 least connected regions:
6 36 65 195 357 365 370 371 481 506 with 1 link
4 most connected regions:
176 234 326 336 with 8 links

> boston.listw <- nb2listw(boston.soi)
> summary(boston.listw)
Error in summary.nb(boston.listw) : Not a neighbours list
In addition: Warning message:
the condition has length > 1 and only the first element will be used
in: if (class(nb) != "nb") stop("Not a neighbours list")
> print(boston.listw)
$style
[1] "W"

$neighbours
[[1]]
[1] 3 30 32 35

```

...

### ***Boston Moran's I for Residual Spatial Autocorrelation***

```
> boston.moran <-  
lm.morantest(bol,boston.listw,alternative="two.sided")  
> print(boston.moran)
```

Global Moran's I for regression residuals

```
data:  
model: lm(formula = log(MEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +  
I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),  
data = boston.c)  
weights: boston.listw
```

Moran I statistic standard deviate = 14.5085, p-value = < 2.2e-16  
alternative hypothesis: two.sided  
sample estimates:

Observed Moran's I	Expectation	Variance
0.4364296993	-0.0168870829	0.0009762383

### ***Boston Naïve Moran's I***

```
> boston.e <- resid(bol)  
> boston.morane <- moran.test(boston.e,boston.listw,  
+ randomisation=FALSE,alternative="two.sided")  
> print(boston.morane)
```

Moran's I test under normality

```
data: boston.e  
weights: boston.listw
```

Moran I statistic standard deviate = 13.7764, p-value = < 2.2e-16  
alternative hypothesis: two.sided  
sample estimates:

Moran I statistic	Expectation	Variance
0.436429699	-0.001980198	0.001012720

### ***Boston Lagrange Multiplier Tests***

```
> boston.lagrange <- lm.LMtests(bol,boston.listw,  
+ test=c("LMerr","RLMerr","LMlag","RLMlag","SARMA"))  
> print(boston.lagrange)
```

Lagrange multiplier diagnostics for spatial dependence

```
data:  
model: lm(formula = log(MEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +  
I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),  
data = boston.c)  
weights: boston.listw
```

LMerr = 186.5702, df = 1, p-value = < 2.2e-16

Lagrange multiplier diagnostics for spatial dependence

```
data:
model: lm(formula = log(MEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +
I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
data = boston.c)
weights: boston.listw
```

RLMerr = 37.606, df = 1, p-value = 8.658e-10

Lagrange multiplier diagnostics for spatial dependence

```
data:
model: lm(formula = log(MEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +
I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
data = boston.c)
weights: boston.listw
```

LMlag = 190.7132, df = 1, p-value = < 2.2e-16

Lagrange multiplier diagnostics for spatial dependence

```
data:
model: lm(formula = log(MEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +
I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
data = boston.c)
weights: boston.listw
```

RLMlag = 41.749, df = 1, p-value = 1.038e-10

Lagrange multiplier diagnostics for spatial dependence

```
data:
model: lm(formula = log(MEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +
I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
data = boston.c)
weights: boston.listw
```

SARMA = 228.3191, df = 2, p-value = < 2.2e-16

### ***Boston Spatial Lag Model***

#### **OLS (inconsistent!)**

```
> attach(boston.c)
> lagMEDV = lag.listw(boston.listw,log(MEDV))
> bo2 <- lm(log(MEDV) ~ lagMEDV + CRIM + ZN + INDUS + CHAS +
+ I(NOX^2) + I(RM^2) + AGE + log(DIS) + log(RAD) + TAX +
+ PTRATIO + B + log(LSTAT),data=boston.c)
> summary(bo2)
```

Call:

```
lm(formula = log(MEDV) ~ lagMEDV + CRIM + ZN + INDUS + CHAS +
I(NOX^2) + I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO +
```

```
B + log(LSTAT), data = boston.c)
```

```
Residuals:
```

	Min	1Q	Median	3Q	Max
	-0.57709	-0.07490	-0.00451	0.07778	0.71192

```
Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	1.953e+00	1.948e-01	10.029	< 2e-16	***
lagMEDV	5.551e-01	3.232e-02	17.175	< 2e-16	***
CRIM	-6.517e-03	1.033e-03	-6.309	6.26e-10	***
ZN	4.315e-04	4.006e-04	1.077	0.281875	
INDUS	1.537e-03	1.872e-03	0.821	0.411814	
CHAS1	-3.959e-03	2.685e-02	-0.147	0.882840	
I(NOX^2)	-2.210e-01	9.275e-02	-2.383	0.017558	*
I(RM^2)	6.829e-03	1.039e-03	6.575	1.24e-10	***
AGE	-3.026e-04	4.170e-04	-0.726	0.468429	
log(DIS)	-1.479e-01	2.654e-02	-5.572	4.15e-08	***
log(RAD)	7.343e-02	1.519e-02	4.833	1.80e-06	***
TAX	-3.634e-04	9.712e-05	-3.741	0.000205	***
PTRATIO	-1.068e-02	4.141e-03	-2.580	0.010163	*
B	2.717e-04	8.176e-05	3.324	0.000955	***
log(LSTAT)	-2.098e-01	2.190e-02	-9.577	< 2e-16	***

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.1444 on 491 degrees of freedom  
Multiple R-Squared: 0.8787, Adjusted R-squared: 0.8753  
F-statistic: 254.2 on 14 and 491 DF, p-value: < 2.2e-16
```

## ML-Lag

```
> bo.lag <- lagsarlm(log(MEDV) ~ CRIM + ZN + INDUS + CHAS +  
+ I(NOX^2) + I(RM^2) + AGE + log(DIS) + log(RAD) + TAX +  
+ PTRATIO + B + log(LSTAT), data=boston.c, boston.listw)
```

```
> summary(bo.lag)
```

```
Call:
```

```
lagsarlm(formula = log(MEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +  
I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B +  
log(LSTAT), data = boston.c, listw = boston.listw)
```

```
Residuals:
```

	Min	1Q	Median	3Q	Max
	-0.5726328	-0.0757891	-0.0034976	0.0733915	0.7139960

```
Type: lag
```

```
Coefficients: (asymptotic standard errors)
```

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	2.3233e+00	1.7959e-01	12.9368	< 2.2e-16
CRIM	-7.2765e-03	9.9145e-04	-7.3393	2.147e-13
ZN	3.8160e-04	3.9680e-04	0.9617	0.3362071
INDUS	1.3529e-03	1.8533e-03	0.7300	0.4653851
CHAS1	9.5916e-03	2.6191e-02	0.3662	0.7142070
I(NOX^2)	-2.8028e-01	9.0681e-02	-3.0908	0.0019960

I(RM^2)	6.7583e-03	1.0343e-03	6.5340	6.402e-11
AGE	-2.4669e-04	4.1278e-04	-0.5976	0.5500854
log(DIS)	-1.5405e-01	2.6301e-02	-5.8572	4.709e-09
log(RAD)	7.6598e-02	1.5078e-02	5.0802	3.770e-07
TAX	-3.7144e-04	9.6589e-05	-3.8456	0.0001202
PTRATIO	-1.3589e-02	4.0900e-03	-3.3225	0.0008921
B	2.8481e-04	8.1775e-05	3.4828	0.0004962
log(LSTAT)	-2.3272e-01	2.1001e-02	-11.0816	< 2.2e-16

Rho: 0.47619 LR test value: 199.22 p-value: < 2.22e-16

Asymptotic standard error: 0.030145 z-value: 15.797 p-value: < 2.22e-16

Log likelihood: 249.5643 for lag model

ML residual variance (sigma squared): 0.020466, (sigma: 0.14306)

Number of observations: 506

Number of parameters estimated: 15

AIC: -469.13, (AIC for lm: -269.91)

LM test for residual autocorrelation

test value: 11.394 p-value: 0.00073662

### ***Boston Spatial Error Model***

```
> boston.err <- errorsarlm(log(MEDV) ~ CRIM + ZN + INDUS + CHAS +
+ I(NOX^2) + I(RM^2) + AGE + log(DIS) + log(RAD) + TAX +
+ PTRATIO + B + log(LSTAT),data=boston.c,boston.listw)
```

```
> summary(boston.err)
```

Call:

```
errorsarlm(formula = log(MEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +
I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B +
log(LSTAT), data = boston.c, listw = boston.listw)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-0.6476342	-0.0676007	0.0011091	0.0776939	0.6491629

Type: error

Coefficients: (asymptotic standard errors)

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	3.85706016	0.16083868	23.9809	< 2.2e-16
CRIM	-0.00545832	0.00097262	-5.6120	2.000e-08
ZN	0.00049195	0.00051835	0.9491	0.3425906
INDUS	0.00019244	0.00282240	0.0682	0.9456389
CHAS1	-0.03303429	0.02836929	-1.1644	0.2442464
I(NOX^2)	-0.23369330	0.16219195	-1.4408	0.1496288
I(RM^2)	0.00800078	0.00106472	7.5145	5.707e-14
AGE	-0.00090974	0.00050116	-1.8153	0.0694827
log(DIS)	-0.10889419	0.04783715	-2.2764	0.0228249
log(RAD)	0.07025730	0.02108181	3.3326	0.0008604
TAX	-0.00049870	0.00012072	-4.1311	3.611e-05
PTRATIO	-0.01907770	0.00564160	-3.3816	0.0007206
B	0.00057442	0.00011101	5.1744	2.286e-07
log(LSTAT)	-0.27212779	0.02323159	-11.7137	< 2.2e-16

Lambda: 0.70175 LR test value: 211.88 p-value: < 2.22e-16

Asymptotic standard error: 0.032698 z-value: 21.461 p-value: < 2.22e-16

Log likelihood: 255.8946 for error model  
ML residual variance (sigma squared): 0.018098, (sigma: 0.13453)  
Number of observations: 506  
Number of parameters estimated: 15  
AIC: -481.79, (AIC for lm: -269.91)

### ***Boston Spatial Common Factor Hypothesis***

#### **ML estimation spatial Durbin model**

```
> boston.durbn <- lagsarlm( log(MEDV) ~ CRIM + ZN + INDUS + CHAS +  
+ I(NOX^2) + I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO +  
+ B + log(LSTAT),data=boston.c,listw=boston.listw,type="mixed")
```

```
> summary(boston.durbn)
```

Call:

```
lagsarlm(formula = log(MEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +  
I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B +  
log(LSTAT), data = boston.c, listw = boston.listw, type = "mixed")
```

Residuals:

	Min	1Q	Median	3Q	Max
	-0.6364278	-0.0640563	-0.0028092	0.0712728	0.6938596

Type: mixed

Coefficients: (asymptotic standard errors)

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	1.95278292	0.23318988	8.3742	< 2.2e-16
CRIM	-0.00585087	0.00096535	-6.0608	1.354e-09
ZN	0.00072312	0.00053544	1.3505	0.176853
INDUS	-0.00087549	0.00317281	-0.2759	0.782597
CHAS1	-0.03815722	0.02826977	-1.3498	0.177095
I(NOX^2)	-0.00295140	0.19985674	-0.0148	0.988218
I(RM^2)	0.00796469	0.00105360	7.5595	4.041e-14
AGE	-0.00115867	0.00050497	-2.2945	0.021760
log(DIS)	-0.11456383	0.09818152	-1.1669	0.243268
log(RAD)	0.05632375	0.02330111	2.4172	0.015640
TAX	-0.00049108	0.00012537	-3.9171	8.963e-05
PTRATIO	-0.01323334	0.00614623	-2.1531	0.031312
B	0.00054285	0.00011446	4.7427	2.109e-06
log(LSTAT)	-0.25496722	0.02334504	-10.9217	< 2.2e-16
lag.CRIM	-0.00501244	0.00178601	-2.8065	0.005008
lag.ZN	-0.00040563	0.00072865	-0.5567	0.577742
lag.INDUS	0.00013376	0.00398366	0.0336	0.973214
lag.CHAS1	0.12050389	0.04202611	2.8674	0.004139
lag.I(NOX^2)	-0.41786543	0.22877096	-1.8266	0.067765
lag.I(RM^2)	-0.00460612	0.00157869	-2.9177	0.003526
lag.AGE	0.00128725	0.00070634	1.8224	0.068390
lag.log(DIS)	-0.01455930	0.10366694	-0.1404	0.888310
lag.log(RAD)	0.00421832	0.03212137	0.1313	0.895519
lag.TAX	0.00039204	0.00018447	2.1252	0.033572
lag.PTRATIO	-0.00184503	0.00815813	-0.2262	0.821078
lag.B	-0.00047673	0.00014613	-3.2625	0.001104

```
lag.log(LSTAT) 0.11255101 0.03490285 3.2247 0.001261
```

```
Rho: 0.58793 LR test value: 171.98 p-value: < 2.22e-16
```

```
Asymptotic standard error: 0.038997 z-value: 15.076 p-value: < 2.22e-16
```

```
Log likelihood: 285.3799 for mixed model
```

```
ML residual variance (sigma squared): 0.017062, (sigma: 0.13062)
```

```
Number of observations: 506
```

```
Number of parameters estimated: 28
```

```
AIC: -514.76, (AIC for lm: -342.78)
```

```
LM test for residual autocorrelation
```

```
test value: 26.539 p-value: 2.5824e-07
```

### Test on common factor hypothesis

```
> boston.cf <- LR.sarlm(boston.durbn,boston.err)
```

```
> boston.cf
```

```
Likelihood ratio for spatial linear models
```

```
data:
```

```
Likelihood ratio = 58.9705, df = 1, p-value = 1.599e-14
```

```
sample estimates:
```

```
Log likelihood of boston.durbn    Log likelihood of boston.err  
                285.3799                255.8946
```

```
> 1 - pchisq(boston.cf[[1]][1],11)
```

```
Likelihood ratio
```

```
1.439262e-08
```