Prof. Dr. Reinhold Kosfeld October 2018

Spatial Econometrics with R
- Spatial Data Analysis of the 5-Region Script Example

Table of contents

1. Introduction

The statistical software and programming environment R is developed by an international community of developers as open source software and can be freely used and disseminated as such by everyone according to the GNU General Public License version 2 (June 1991). The software is provided on the homepage of the R project (http://www.r-project.org) for download. Source code and binary files for various operating systems are provided by CRAN (http://CRAN.R-project.org).

R is a highly flexible, interpreted programming language and environment for statistical and graphical data analysis. It is not a complete graphical user interface (GUI), but tools are available to their development. In the R-Commander some methods of data analysis can already be run menu driven. During the execution of R-functions normally there is no output of all calculated values. Rather, some results are initially cached into objects. They can later be retrieved at any time. On the other hand R saves no results if no object is specified for this purpose.

R can be used both interactively or in batch mode. The batch mode should be used with larger instruction sequences, in which a script file is executed. The program is ready when the sign > appears. Commands can be separated by a semicolon (;) or by the beginning of a new line. In case of an incomplete command, the plus sign (+) appears. Already used commands can be retrieved with the arrow keys (↑ und ↓). The sign # initiates a comment. As the decimal character the point (.) not the comma (,) is used. The ESC key interrupts the currently running computing process.

All present objects in an R session can be displayed with the command ls(). An object obj can be deleted with the remove command rm(obj). Texts such as "Spatial Econometrics makes fun" can be assigned to an object obj using quotation marks with an equal sign (=) or arrow (<-):

```
> obj = "Spatial Econometrics makes fun"
or
> obj <- "Spatial Econometrics makes fun"
>obj
[1] "Spatial Econometrics makes fun".
```

Here we only make use of the equal sign (=) in assignments. In the above case, the object obj is of type character (string object (text string)).

The available objects in the current R environment are displayed with the command ls() in R the console:

```
>ls()
  [1] "obj"
```

The current working directory can checked with the getwd() command:

```
>getwd()
[1] "C:/Users/Kosfeld/Dokumente/Spatial Econometrics/LV Spatial Econometrics/R
```

It can be changed by the R menu File - Change Dir ... or the setwd() command:

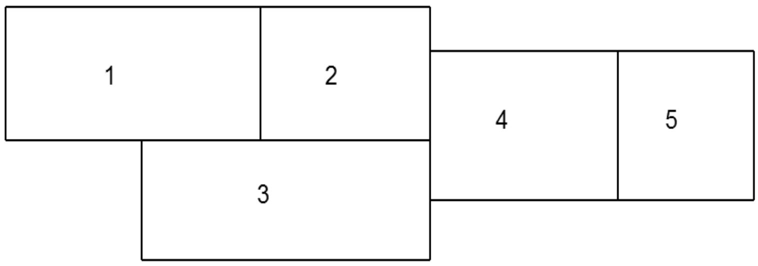
```
> setwd("C:/Users/Kosfeld/ Dokumente/Spatial Econometrics/LV Spatial
Econometrics/R")
```

The R program can be closed through the quit command q() or by clicking the icon x of the program window RGui (64-bit).

The script focuses its attention specifically on spatial data structures and functions for spatial econometric analysis. We use a simple 5-region example in explaining spatial data analysis with R. First of all, the creation of list weight objects and reading area data into R are considered (ch. 2). Then we draw our attention to spatial autocorrelation analysis and mapping of spatial data (ch. 3). Chapter 4 is devoted to cross-sectional regression analysis. Although special R functions of spatial econometric methods are available, for some calculations user-defined functions and matrix operations are introduced. The standard econometric model constitutes the starting point of the spatial variants. Chapter 5 deals with the pooling as wells as fixed and random effects panel data models. Spatial panel data models are instroduced in chapter 6. Finally, I chapter 7, shape files, weights matrices and area data are provided for real applications of spatial data analysis.

2. Creating spatial weights matrices, variable vectors and spatial lags
2.1 First-order contiguity-based weights matrix and weights list object

An irregular arrangement of five spatial units (regions) is given as follows:



Original (unstandardized) spatial weights matrix as a matrix object

```
> WS5EX = matrix(c(0,1,1,0,0,1,0,1,1,0,1,1,0,1,0,0,1,1,0,1,0,0,0,1,0), nrow=5,
ncol=5, byrow=TRUE)
> class(WS5EX)
[1] "matrix"
> WS5EX
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    1    1    0    0
[2,]    1    0    1    1    0
[3,]    1    1    0    1    0
[4,]    0    1    1    0    1
[5,]    0    0    0    1    0
> colnames(WS5EX)
NULL
> colnames(WS5EX) = 1:5
> colnames(WS5EX)
[1] "1" "2" "3" "4" "5"
> rownames(WS5EX) = 1:5
> rownames(WS5EX)
[1] "1" "2" "3" "4" "5"
> WS5EX
  1 2 3 4 5
1 0 1 1 0 0
2 1 0 1 1 0
3 1 1 0 1 0
4 0 1 1 0 1
5 0 0 0 1 0
```

Row-standardizing the spatial weights matrix

```
> WS5EX.sums.rows = apply(WS5EX, 1, sum)
# Argument MARGIN=1: The function FUN=sum is applied over rows
> WS5EX.sums.rows
1 2 3 4 5
2 3 3 3 1
```

```
> W5EX = WS5EX/WS5EX.sums.rows
# Row-standardized spatial weights matrix as a matrix object W5EX
> W5EX
          1         2         3         4         5
1 0.0000000 0.5000000 0.5000000 0.0000000 0.0000000
2 0.3333333 0.0000000 0.3333333 0.3333333 0.0000000
3 0.3333333 0.3333333 0.0000000 0.3333333 0.0000000
4 0.0000000 0.3333333 0.3333333 0.0000000 0.3333333
5 0.0000000 0.0000000 0.0000000 1.0000000 0.0000000
```

Converting the spatial weights matrix to a weights list object

```
> library(spdep)
> W5EX.lw = mat2listw(W5EX, style="W")
> class(W5EX.lw)
[1] "listw" "nb"
# Component nb (W5EX.lw$neighbours): list of neighbours
# Component listw (W5EX.lw$weights): list of weights

> W5EX.lw
Characteristics of weights list object:
Neighbour list object:
Number of regions: 5
Number of nonzero links: 12
Percentage nonzero weights: 48
Average number of links: 2.4
Weights style: W
Weights constants summary:
  N nn S0  S1     S2
W 5 25  5   4.5 21.05556
```

n: number of regions, nn: squared number of regions, S0: sum of weights w_{ij}
S1: half of the double sum of $(w_{ij} + w_{ji})^2$, S2: sum of $(w_{i\bullet} + w_{\bullet i})^2$

```
> summary(W5EX.lw)
Characteristics of weights list object:
Neighbour list object:
Number of regions: 5
Number of nonzero links: 12
Percentage nonzero weights: 48
Average number of links: 2.4
Link number distribution:
1 2 3
1 1 3
1 least connected region:
5 with 1 link
3 most connected regions:
2 3 4 with 3 links
Weights style: W
Weights constants summary:
  N nn S0   S1     S2
W 5 25  5    4.5 21.05556
```

```
> str(W5EX.lw)
List of 3
 $ style     : chr "W"
 $ neighbours:List of 5
  ..$ : Named int [1:2] 2 3
  .. ..- attr(*, "names")= chr [1:2] "2" "3"
  ..$ : Named int [1:3] 1 3 4
  .. ..- attr(*, "names")= chr [1:3] "1" "3" "4"
  ..$ : Named int [1:3] 1 2 4
  .. ..- attr(*, "names")= chr [1:3] "1" "2" "4"
  ..$ : Named int [1:3] 2 3 5
  .. ..- attr(*, "names")= chr [1:3] "2" "3" "5"
  ..$ : Named int 4
  .. ..- attr(*, "names")= chr "4"
  ..- attr(*, "class")= chr "nb"
  ..- attr(*, "region.id")= chr [1:5] "1" "2" "3" "4" ...
  ..- attr(*, "call")= logi NA
  ..- attr(*, "sym")= logi TRUE
 $ weights   :List of 5
  ..$ : num [1:2] 0.5 0.5
  ..$ : num [1:3] 0.333 0.333 0.333
  ..$ : num [1:3] 0.333 0.333 0.333
  ..$ : num [1:3] 0.333 0.333 0.333
  ..$ : num 1
  ..- attr(*, "mode")= chr "general"
  ..- attr(*, "glist")= chr [1:4] "list(c(0.5, 0.5), c(0.333333333333333, 
0.333333333333333, 0.333333333333333" "), c(0.333333333333333, 
0.333333333333333, 0.333333333333333), " "    c(0.333333333333333, 
0.333333333333333, 0.333333333333333" "    ), 1)"
  ..- attr(*, "glistsym")= atomic [1:1] FALSE
  .. ..- attr(*, "d")= num 0.667
  ..- attr(*, "W")= logi TRUE
  ..- attr(*, "comp")=List of 1
  .. ..$ d: num [1:5] 1 1 1 1 1
 - attr(*, "class")= chr [1:2] "listw" "nb"
 - attr(*, "region.id")= chr [1:5] "1" "2" "3" "4" ...
 - attr(*, "call")= language nb2listw(neighbours = res$neighbours, glist = res$weights, 
style = style,      zero.policy = TRUE)

# Extracting contiguity neighbours
> W5EX.nb = W5EX.lw$neighbours
> class(W5EX.nb)
[1] "nb"
> W5EX.nb
Neighbour list object:
Number of regions: 5 
Number of nonzero links: 12 
Percentage nonzero weights: 48 
Average number of links: 2.4
```

```
> summary(W5EX.nb)
Neighbour list object:
Number of regions: 5
Number of nonzero links: 12
Percentage nonzero weights: 48
Average number of links: 2.4
Link number distribution:
1 2 3
1 1 3
1 least connected region:
5 with 1 link
3 most connected regions:
2 3 4 with 3 links

> str(W5EX.nb)
List of 5
 $ : Named int [1:2] 2 3
  ..- attr(*, "names")= chr [1:2] "2" "3"
 $ : Named int [1:3] 1 3 4
  ..- attr(*, "names")= chr [1:3] "1" "3" "4"
 $ : Named int [1:3] 1 2 4
  ..- attr(*, "names")= chr [1:3] "1" "2" "4"
 $ : Named int [1:3] 2 3 5
  ..- attr(*, "names")= chr [1:3] "2" "3" "5"
 $ : Named int 4
  ..- attr(*, "names")= chr "4"
 - attr(*, "class")= chr "nb"
 - attr(*, "region.id")= chr [1:5] "1" "2" "3" "4" ...
 - attr(*, "call")= logi NA
 - attr(*, "sym")= logi TRUE
```

<u>Spatial neighbour sparse representation</u>

```
> W5EX.df = listw2sn(W5EX.lw)
>  class(W5EX.df)
[1] "data.frame"        "spatial.neighbour"
> W5EX.df
   from to   weights
1     1  2 0.5000000
2     1  3 0.5000000
3     2  1 0.3333333
4     2  3 0.3333333
5     2  4 0.3333333
6     3  1 0.3333333
7     3  2 0.3333333
8     3  4 0.3333333
9     4  2 0.3333333
10    4  3 0.3333333
11    4  5 0.3333333
12    5  4 1.0000000
```

```
>  str(W5EX.df)
Classes ‘spatial.neighbour’ and 'data.frame':   12 obs. of  3 variables:
 $ from   : int  1 1 2 2 2 3 3 3 4 4 ...
 $ to     : int  2 3 1 3 4 1 2 4 2 3 ...
 $ weights: num  0.5 0.5 0.333 0.333 0.333 ...
 - attr(*, "n")= int 5
 - attr(*, "region.id")= chr  "1" "2" "3" "4" ...
 - attr(*, "neighbours.attrs")= chr  "class" "region.id" "call" "sym"
 - attr(*, "weights.attrs")= chr  "mode" "glist" "glistsym" "W" ...
 - attr(*, "listw.call")= language nb2listw(neighbours = res$neighbours, glist = 
res$weights, style = style,       zero.policy = TRUE)
```

2.2 Cumulated second-order contiguity-based weights matrix and list weights object

<u>Squared Weights Matrix</u>

```
> WS5EXL2 = WS5EX%*%WS5EX
> WS5EXL2
  1 2 3 4 5
1 2 1 1 2 0
2 1 3 2 1 1
3 1 2 3 1 1
4 2 1 1 3 0
5 0 1 1 0 1
```

<u>Sum of 1st Order Contiguity Matrix and Squared Weights Matrix</u>

```
> WS5EXCUM2 = WS5EX + WS5EXL2
> WS5EXCUM2
  1 2 3 4 5
1 2 2 2 2 0
2 2 3 3 2 1
3 2 3 3 2 1
4 2 2 2 3 1
5 0 1 1 1 1

> diag(WS5EXCUM2) = 0
> WS5EXCUM2
  1 2 3 4 5
1 0 2 2 2 0
2 2 0 3 2 1
3 2 3 0 2 1
4 2 2 2 0 1
5 0 1 1 1 0
> for (i in 1:5) {for (j in 1:5) {if (WS5EXCUM2[i,j]>1) {WS5EXCUM2[i,j]=1}}}
```

```
> WS5EXCUM2
  1 2 3 4 5
1 0 1 1 1 0
2 1 0 1 1 1
3 1 1 0 1 1
4 1 1 1 0 1
5 0 1 1 1 0
> WS5EXCUM2.sums.rows = apply(WS5EXCUM2, 1, sum)
> WS5EXCUM2.sums.rows
1 2 3 4 5
3 4 4 4 3
```

Cumulated second-order contiguity-based weights matrix as a matrix object

```
> W5EXCUM2 = WS5EXCUM2/WS5EXCUM2.sums.rows
> W5EXCUM2
     1         2         3         4    5
1 0.00 0.3333333 0.3333333 0.3333333 0.00
2 0.25 0.0000000 0.2500000 0.2500000 0.25
3 0.25 0.2500000 0.0000000 0.2500000 0.25
4 0.25 0.2500000 0.2500000 0.0000000 0.25
5 0.00 0.3333333 0.3333333 0.3333333 0.00
```

Cumulated second-order contiguity-based weights matrix as a list weights object

```
> W5EXCUM2.lw = mat2listw(W5EXCUM2, style="W")
> summary(W5EXCUM2.lw)
Characteristics of weights list object:
Neighbour list object:
Number of regions: 5
Number of nonzero links: 18
Percentage nonzero weights: 72
Average number of links: 3.6
Link number distribution:
3 4
2 3
2 least connected regions:
1 5 with 3 links
3 most connected regions:
2 3 4 with 4 links
Weights style: W
Weights constants summary:
  n nn S0       S1       S2
W 5 25  5 2.791667 20.20833
```

2.3 Distance-based weights matrix and list weights object

Locational information is provided by the coordinates of the regional centres:

Region	1	2	3	4	5
Coordinates	(5.5; 5)	(9.5; 5)	(5.5; 3)	(13; 4)	(16; 4)

Coordinates of regional centres

```
> Coord5EX.centres = matrix(c(5.5, 5, 9.5, 5, 5.5, 3, 13, 4, 16, 4), nrow=5, ncol=2,
byrow=TRUE)
> class(Coord5EX.centres)
[1] "matrix"
> Coord5EX.centres
      [,1]  [,2]
[1,]  5.5    5
[2,]  9.5    5
[3,]  5.5    3
[4,] 13.0    4
[5,] 16.0    4
> colnames(Coord5EX.centres)
NULL
> colnames(Coord5EX.centres) = 1:2
> colnames(Coord5EX.centres)
[1] "1" "2"
> rownames(Coord5EX.centres) = 1:5
> rownames(Coord5EX.centres)
[1] "1" "2" "3" "4" "5"
> Coord5EX.centres
     1   2
1  5.5  5
2  9.5  5
3  5.5  3
4 13.0 4
5 16.0 4
```

Euclidean distances between the centres of the regions

```
> Dist5EX.centres = dist(Coord5EX.centres, diag=TRUE, upper=TRUE)
or
> Dist5EX.centres = dist(Coord5EX.centres, method="euclidean", diag=TRUE,
upper=TRUE)
> class(Dist5EX.centres)
[1] "dist"
> Dist5EX.centres
          1         2         3         4         5
1  0.000000  4.000000  2.000000  7.566373 10.547512
2  4.000000  0.000000  4.472136  3.640055  6.576473
3  2.000000  4.472136  0.000000  7.566373 10.547512
4  7.566373  3.640055  7.566373  0.000000  3.000000
5 10.547512  6.576473 10.547512  3.000000  0.000000
```

Inverse Euclidean distances (= unstandardized distance-based spatial weights) between the centres of the regions

```
> Winvdist5EX = 1/Dist5EX.centres
> Winvdist5EX
           1          2          3          4          5
1 0.00000000 0.25000000 0.50000000 0.13216372 0.09480909
2 0.25000000 0.00000000 0.22360680 0.27472113 0.15205718
3 0.50000000 0.22360680 0.00000000 0.13216372 0.09480909
4 0.13216372 0.27472113 0.13216372 0.00000000 0.33333333
5 0.09480909 0.15205718 0.09480909 0.33333333 0.00000000
> class(Winvdist5EX)
[1] "dist"
> Winvdist5EX = as.matrix(Winvdist5EX)
> class(Winvdist5EX)
[1] "matrix"
```

Distance-based weights matrix as a list weights object

```
> Winvdist5EX.lw = mat2listw(Winvdist5EX, style="W")
> summary(Winvdist5EX.lw)
Characteristics of weights list object:
Neighbour list object:
Number of regions: 5
Number of nonzero links: 20
Percentage nonzero weights: 80
Average number of links: 4
Link number distribution:
4
5
5 least connected regions:
1 2 3 4 5 with 4 links
5 most connected regions:
1 2 3 4 5 with 4 links
Weights style: W
Weights constants summary:
  n  nn  S0      S1         S2
W 5  25   5  3.183273  20.08182
> str(Winvdist5EX.lw)
List of 3
 $ style     : chr "W"
 $ neighbours:List of 5
  ..$ : Named int [1:4] 2 3 4 5
  .. ..- attr(*, "names")= chr [1:4] "2" "3" "4" "5"
  ..$ : Named int [1:4] 1 3 4 5
  .. ..- attr(*, "names")= chr [1:4] "1" "3" "4" "5"
  ..$ : Named int [1:4] 1 2 4 5
  .. ..- attr(*, "names")= chr [1:4] "1" "2" "4" "5"
  ..$ : Named int [1:4] 1 2 3 5
  .. ..- attr(*, "names")= chr [1:4] "1" "2" "3" "5"
```

```
  ..$ : Named int [1:4] 1 2 3 4
  .. ..- attr(*, "names")= chr [1:4] "1" "2" "3" "4"
  ..- attr(*, "class")= chr "nb"
  ..- attr(*, "region.id")= chr [1:5] "1" "2" "3" "4" ...
  ..- attr(*, "call")= logi NA
  ..- attr(*, "sym")= logi TRUE
 $ weights   :List of 5
  ..$ : num [1:4] 0.256 0.512 0.135 0.097
  ..$ : num [1:4] 0.278 0.248 0.305 0.169
  ..$ : num [1:4] 0.526 0.2352 0.139 0.0997
  ..$ : num [1:4] 0.151 0.315 0.151 0.382
  ..$ : num [1:4] 0.14 0.225 0.14 0.494
  ..- attr(*, "mode")= chr "general"
  ..- attr(*, "glist")= chr [1:5] "list(c(0.25, 0.5, 0.132163720091018,
0.0948090926279954), c(0.25, " "0.223606797749979, 0.274721127897378,
0.152057184253941), c(0.5, " "0.223606797749979, 0.132163720091018,
0.0948090926279954), c(0.132163720091018, " "0.274721127897378,
0.132163720091018, 0.333333333333333), c(0.0948090926279954, " ...
  ..- attr(*, "glistsym")= atomic [1:1] TRUE
  .. ..- attr(*, "d")= num 0
  ..- attr(*, "W")= logi TRUE
  ..- attr(*, "comp")=List of 1
  .. ..$ d: num [1:5] 0.977 0.9 0.951 0.872 0.675
 - attr(*, "class")= chr [1:2] "listw" "nb"
 - attr(*, "region.id")= chr [1:5] "1" "2" "3" "4" ...
 - attr(*, "call")= language nb2listw(neighbours = res$neighbours, glist = res$weights,
style = style,      zero.policy = TRUE)
```

2.4 Geo-referenced variables and spatial lags

Geo-referenced data on the variables unemployment rate (u), output growth (gx) and productivity growth (gy) are available:

Region	u	gx	gy
1	8	0.6	0.4
2	6	1.0	0.6
3	6	1.6	0.9
4	3	2.6	1.1
5	2	2.2	1.2

<u>Regional unemployment rate and its spatial lag</u>

```
> u5EX = c(8, 6, 6, 3, 2)
> u5EX
[1] 8 6 6 3 2
> Lu5EX = W5EX%*%u5EX
```

```
> Lu5EX
      [,1]
1 6.000000
2 5.666667
3 5.666667
4 4.666667
5 3.000000
or
> Lu5EX = lag.listw(W5EX.lw, u5EX)
# lag.listw: function of library spdep
> Lu5EX
[1] 6.000000 5.666667 5.666667 4.666667 3.000000
```

Output growth and its spatial lag

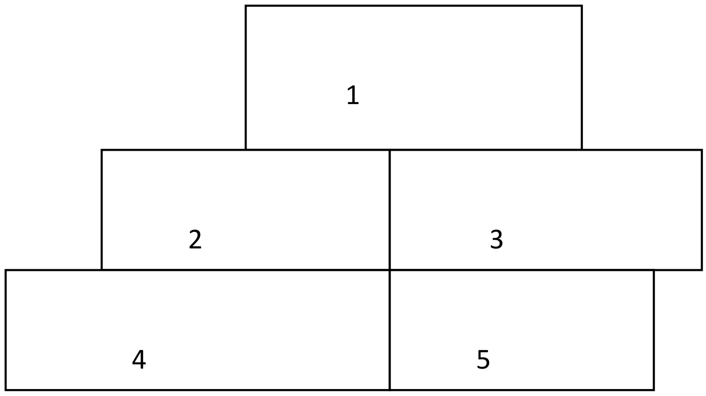
```
> gx5EX = c(0.6, 1.0, 1.6, 2.6, 2.2)
> gx5EX
[1] 0.6 1.0 1.6 2.6 2.2
> Lgx5EX = W5EX%*%gx5EX
> Lgx5EX
  [,1]
1  1.3
2  1.6
3  1.4
4  1.6
5  2.6
or
> Lgx5EX = lag.listw(W5EX.lw, gx5EX)
> Lgx5EX
[1] 1.3 1.6 1.4 1.6 2.6
```

Productivity growth and its spatial lag

```
> gy5EX = c(0.4, 0.6, 0.9, 1.1, 1.2)
> gy5EX
[1] 0.4 0.6 0.9 1.1 1.2
> Lgy5EX = W5EX%*%gy5EX
> Lgy5EX
  [,1]
1 0.75
2 0.80
3 0.70
4 0.90
5 1.10
or
> Lgy5EX = lag.listw(W5EX.lw, gy5EX)
> Ly5EX
[1] 0.75 0.80 0.70 0.90 1.10
```

Exercises

1-1 Five regions are arranged in the following form:



Specify the contiguity matrix (first-order neighbouring matrix) WS5EC and the corresponding row-standardized weights matrix W5EC in R as matrix objects!

1-2 Convert the spatial weights matrix W5EC to a weights list object W5EC.lw and interpret its characteristics!

1-3 The coordinates of the regional centres read:

Region	1	2	3	4	5
Coordinates	(9; 12.5)	(6; 7.5)	(16; 7.5)	(5; 2.5)	(15; 2.5)

Compute the Euclidean distance matrix Dist5EC.centres from the matrix of centres coordinates Coord5EC.centres! Check the type of the R object Dist5EC.centres!

1-4 Use the inverse Euclidean distances to define a distance-based weights matrix Winvdist5EC and convert it to a weights list object Winvdist5EX.lw!

1-5 Regional data on the unemployment rate (u) and the vacancy rate (v) are given as follows:

Region	u	v
1	6	3
2	8	3
3	8	2
4	11	1
5	12	1

Define the variable vectors u5EC and v5EC!

1-6 Form the spatial lags Lu5EC and Lv5EC of the geo-referenced variables u5EC and v5EC with the aid of the

- standardized weights matrix W5EC,
- weights list object W5EC.lw

and interpret them!

2. Spatial autocorrelation analysis
2.1 Global spatial autocorrelation coefficients
2.1.1 Moran coefficient

- with matrix operations

Moran coefficient of regional unemployment rate

```
> u5EX
[1] 8 6 6 3 2
> class(u5EX)
[1] "numeric"
>  u.mean = mean(u5EX)
> u.mean
[1] 5
> u5EX-u.mean
[1]  3  1  1 -2 -3
> MIu_num = (u5EX-u.mean)%*%W5EX%*%(u5EX-u.mean)
> class(MIu_num)
[1] "matrix"
> MIu_num
     [,1]
[1,]  11
> MIu_den = (u5EX-u.mean)%*%(u5EX-u.mean)
> class(MIu_den)
[1] "matrix"
> MIu_den
     [,1]
[1,]   24
> MIu = MIu_num/MIu_den
> MIu
          [,1]
[1,] 0.4583333
```

- with R function moran

Moran coefficient of regional unemployment rate

```
> moran(u5EX, listw=W5EX.lw, n=length(u5EX), S0=Szero(W5EX.lw))
$I
[1] 0.4583333
$K
[1] 1.5625
> str(moran(u5EX, listw=W5EX.lw, n=length(u5EX), S0=Szero(W5EX.lw)))
List of 2
 $ I: num 0.458
 $ K: num 1.56

> moran(u5EX, listw=W5EXCUM2.lw, n=length(u5EX), S0=Szero(W5EXCUM2.lw))
$I
[1] -0.0625
```

```
$K
[1] 1.5625

> moran(u5EX, listw=Winvdist5EX.lw, n=length(u5EX), S0=Szero(Winvdist5EX.lw))
$I
[1] 0.1196157
$K
[1] 1.5625
```

Moran coefficient of output growth

```
> moran(x5EX, listw=W5EX.lw, n=length(x5EX), S0=Szero(W5EX.lw))
$I
[1] 0.3308824
$K
[1] 1.526817
> str(moran(x5EX, listw=W5EX.lw, n=length(x5EX), S0=Szero(W5EX.lw)))
List of 2
 $ I: num 0.331
 $ K: num 1.53
```

Moran coefficient of productivity growth

```
> moran(y5EX, listw=W5EX.lw, n=length(y5EX), S0=Szero(W5EX.lw))
$I
[1] 0.3318584
$K
r[1] 1.521693
>  str(moran(y5EX, listw=W5EX.lw, n=length(y5EX), S0=Szero(W5EX.lw)))
List of 2
 $ I: num 0.332
 $ K: num 1.52
```

2.1.2 Moran test (randomization, normality and permutation test)

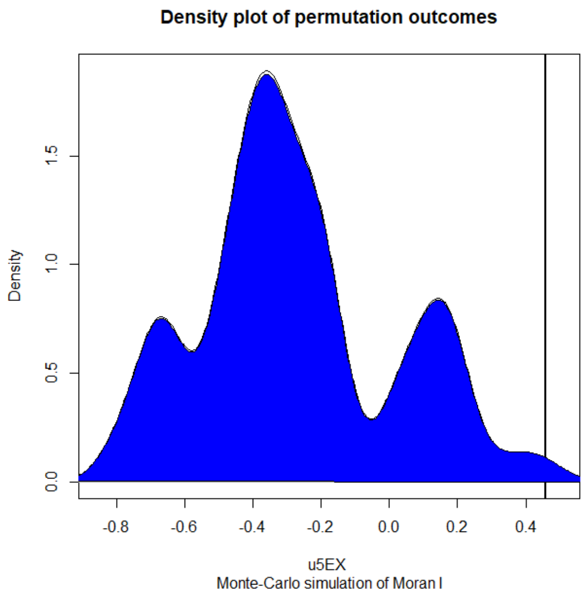
Moran test of regional unemployment rate

```
> moran.test(u5EX, listw=W5EX.lw)
or
> moran.test(u5EX, listw=W5EX.lw, randomisation=TRUE, alternative="greater")

        Moran's I test under randomisation

data:  u5EX
weights: W5EX.lw

Moran I statistic standard deviate = 2.2861, p-value = 0.01113
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
       0.45833333       -0.25000000        0.09600694
```

```
> moran.test(u5EX, listw=W5EX.lw, randomisation=FALSE, alternative="greater")

        Moran's I test under normality

data:  u5EX
weights: W5EX.lw

Moran I statistic standard deviate = 2.5945, p-value = 0.004737
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
       0.45833333       -0.25000000        0.07453704

> set.seed(12345)
> u5EX.MImc = moran.mc(u5EX, listw=W5EX.lw, nsim=99, alternative="greater")
> u5EX.MImc

        Monte-Carlo simulation of Moran's I

data:  u5EX
weights: W5EX.lw
number of simulations + 1: 100

statistic = 0.4583, observed rank = 100, p-value = 0.01
alternative hypothesis: greater
```

A coloured density plot of the simulated distribution of Moran’s I from the permuation test is drawn with the aid of the plot, density and polygon commands:

```
> plot(u5EX.MImc)
> u5EX.MImc.dens = density(u5EX.MImc$res)
> polygon(u5EX.MImc.dens, col="blue")
```



The Moran coefficient of the observed regional unemployment is depicted at the upper tail of the distribution by a vertical bar.

<u>Moran test of output growth</u>

```
> set.seed(12345)
> moran.mc(gx5EX, listw=W5EX.lw, nsim=99, alternative="greater")

        Monte-Carlo simulation of Moran I

data:  gx5EX
weights: W5EX.lw
number of simulations + 1: 100

statistic = 0.3309, observed rank = 92, p-value = 0.08
alternative hypothesis: greater
```

<u>Moran test of productivity growth</u>

```
> set.seed(12345)
> moran.mc(gy5EX, listw=W5EX.lw, nsim=99, alternative="greater")

        Monte-Carlo simulation of Moran I

data:  gy5EX
weights: W5EX.lw
number of simulations + 1: 100

statistic = 0.3319, observed rank = 91, p-value = 0.09
alternative hypothesis: greater
```

2.1.3 Moran scatterplot

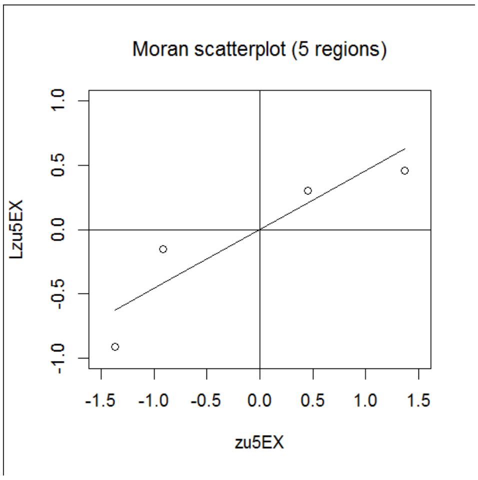
<u>Moran scatterplot of regional unemployment rate</u>

Commands for creating a Moran scatterplot of u5EX in R script file MIscatu5EX.R:

```
# Moran scatterplot of the regional unemployment rate (u5EX)
u5EX.lm = lm(Lu5EX~u5EX)
u5EX.lm
Lu5EX.fit = u5EX.lm$fitted.values
mu5EX = mean(u5EX)
su5EX = sqrt((n-1)/n)*sd(u5EX)
zu5EX = (u5EX-mu5EX)/su5EX
Lzu5EX = lag.listw(W5EX.lw, zu5EX)
Lzu5EX.fit = (Lu5EX.fit-mu5EX)/su5EX
plot(zu5EX, Lzu5EX, main="Moran scatterplot (5 regions)",xlim=c(-1.5, 1.5), ylim=c(-
1, 1), cex.main=1.15, font.main=1)
abline(h=0, v=0)
lines(zu5EX, Lzu5EX.fit)
box(which="figure")
```

Output after running script file MIscatu5EX.R:

```
> u5EX.lm
Call:
lm(formula = Lu5EX ~ u5EX)
Coefficients:
(Intercept)        u5EX
     2.7083      0.4583
```



2.1.4 Geary's C and Geary test

<u>Geary's C of regional unemployment rate</u>

```
> geary(u5EX, listw=W5EX.lw, n=length(u5EX), n1=length(u5EX)-1,
S0=Szero(W5EX.lw))
$C
[1] 0.3333333
$K
[1] 1.5625
```

```
> str(geary(u5EX, listw=W5EX.lw, n=length(u5EX), n1=length(u5EX)-1,
S0=Szero(W5EX.lw)))
List of 2
 $ C: num 0.333
 $ K: num 1.56
```

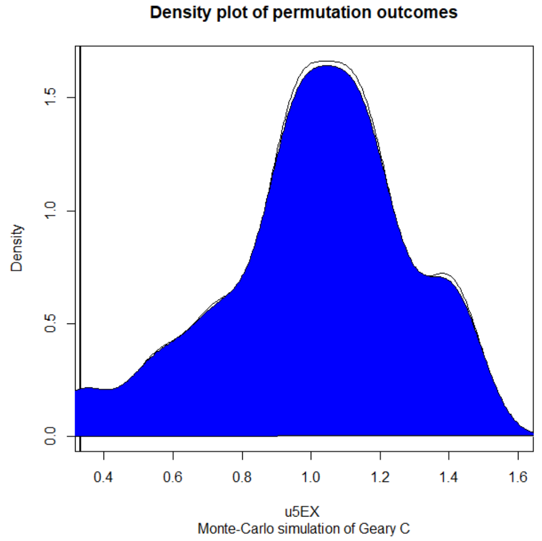
<u>Geary test of regional unemployment rate</u>

```
> geary.test(u5EX, listw=W5EX.lw, randomisation=TRUE, alternative="greater")

        Geary's C test under randomisation

data:  u5EX
weights: W5EX.lw

Geary C statistic standard deviate = 2.4755, p-value = 0.006653
alternative hypothesis: Expectation greater than statistic
```

```
sample estimates:
Geary C statistic      Expectation        Variance
       0.33333333       1.00000000       0.07252778

> geary.test(u5EX, listw=W5EX.lw, randomisation=FALSE, alternative="greater")

        Geary's C test under normality

data:  u5EX
weights: W5EX.lw

Geary C statistic standard deviate = 2.5678, p-value = 0.005118
alternative hypothesis: Expectation greater than statistic
sample estimates:
Geary C statistic      Expectation        Variance
       0.33333333       1.00000000       0.06740741

> set.seed(12345)
> u5EX.gearymc = geary.mc(u5EX, listw=W5EX.lw, nsim=99, alternative="greater")
> u5EX.gearymc

        Monte-Carlo simulation of Geary's C

data:  u5EX
weights: W5EX.lw
number of simulations + 1: 100

statistic = 0.3333, observed rank = 1, p-value = 0.01
alternative hypothesis: greater
```

A coloured density plot of the simulated distribution of Geary's C from the permuation test is drawn with the aid of the plot, density and polygon command:

```
> plot(u5EX.gearymc)
> u5EX.gearymc.dens = density(u5EX.gearymc$res)
> polygon(u5EX.gearymc.dens, col="blue")
```



Geary's C of the observed regional unemployment is depicted at the upper tail of the distribution by a vertical bar.

2.1.5 Getis-Ord global G test

Getis-Ord global G test of regional unemployment rate

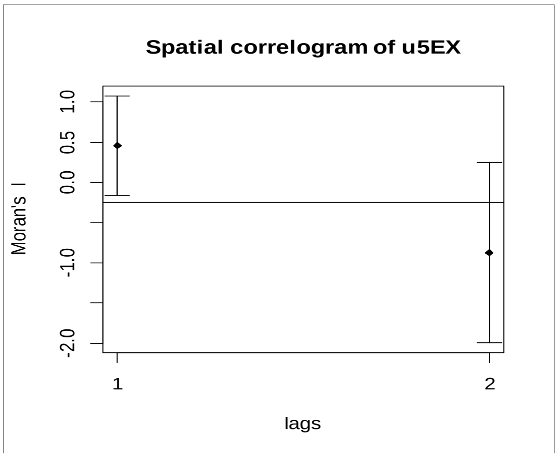
```
> globalG.test(u5EX, listw=W5EX.lw, alternative="greater")

        Getis-Ord global G statistic

data:  u5EX
weights: W5EX.lw

standard deviate = 1.149, p-value = 0.1253
alternative hypothesis: greater
sample estimates:
Global G statistic       Expectation          Variance
      0.2857142857      0.2500000000      0.0009661708

Warning message:
In globalG.test(u5EX, listw = W5EX.lw, alternative = "greater") :
  Binary weights recommended (sepecially for distance bands)

# Binary weights matrix
> WS5EX.lw = mat2listw(WS5EX, style="B")
> globalG.test(u5EX, listw=WS5EX.lw, alternative="greater")

        Getis-Ord global G statistic

data:  u5EX
weights: WS5EX.lw

standard deviate = 1.149, p-value = 0.1253
alternative hypothesis: greater
sample estimates:
Global G statistic       Expectation          Variance
      0.2857142857      0.2500000000      0.0009661708
```

2.1.6 Spatial correlogram

Spatial correlogram of regional unemployment rate

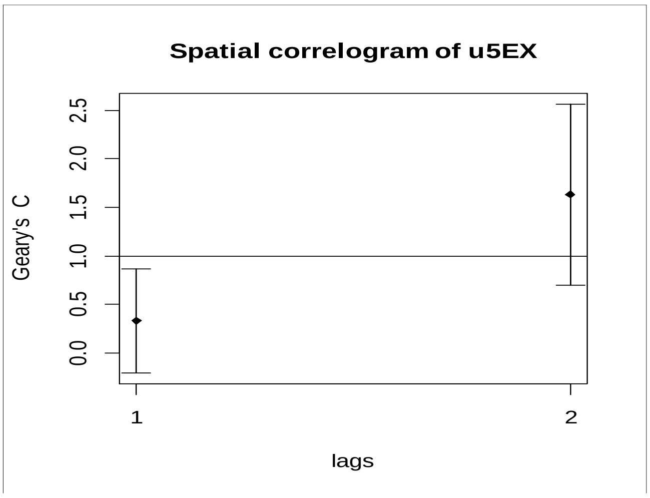
```
> u5EX.spcorrI = sp.correlogram(W5EX.nb, u5EX, order = 2, method = "I", style =
"W", randomisation = TRUE)
> class(u5EX.spcorrI)
[1] "spcor"
> u5EX.spcorrI
Spatial correlogram for u5EX
method: Moran's I
         estimate    expectation    variance  standard deviate Pr(I) two sided
1 (5)  0.458333    -0.250000     0.096007           2.2861         0.02225 *
2 (5) -0.875000    -0.250000     0.314062          -1.1152         0.26474
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> plot(u5EX.spcorrI, main="Spatial correlogram of u5EX")
> box(which="figure")
```




```
> print(u5EX.spcorrI, p.adj.method="none")

Spatial correlogram for u5EX
method: Moran's I
      estimate expectation  variance standard deviate Pr(I) two sided
1 (5)  0.458333   -0.250000  0.096007           2.2861        0.02225 *
2 (5) -0.875000   -0.250000  0.314062          -1.1152        0.26474
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> print(u5EX.spcorrI, p.adj.method="bonferroni")
Spatial correlogram for u5EX
method: Moran's I
         estimate     expectation    variance   standard deviate  Pr(I) two sided
1 (5)  0.458333     -0.250000    0.096007           2.2861           0.0445 *
2 (5) -0.875000     -0.250000    0.314062          -1.1152           0.5295
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> print(u5EX.spcorrI, p.adj.method="holm")
Spatial correlogram for u5EX
method: Moran's I
        estimate    expectation   variance  standard deviate  Pr(I) two sided
1 (5)  0.458333   -0.250000    0.096007           2.2861           0.0445 *
2 (5) -0.875000   -0.250000    0.314062          -1.1152           0.2647
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> print(u5EX.spcorrI, p.adj.method="BY")
Spatial correlogram for u5EX
method: Moran's I
        estimate    expectation  variance  standard deviate  Pr(I) two sided
1 (5)  0.458333   -0.250000   0.096007           2.2861          0.06675 .
2 (5) -0.875000   -0.250000   0.314062          -1.1152          0.39712
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> u5EX.spcorrC = sp.correlogram(W5EX.nb, u5EX, order = 2, method = "C", style =
"W", randomisation = TRUE)
> class(u5EX.spcorrC)
[1] "spcor"
> u5EX.spcorrC

Spatial correlogram for u5EX
method: Geary's C
       estimate   expectation  variance  standard deviate  Pr(I) two sided
1 (5) 0.333333    1.000000   0.072528         -2.4755          0.01331 *
2 (5) 1.633333    1.000000   0.216750          1.3604          0.17372
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> plot(u5EX.spcorrC, main="Spatial correlogram of u5EX")
> box(which="figure")
```



Exercises

2-1-1 Compute Moran's I of the unemployment rate (u5EC) with the contiguity-based weights matrix by matrix calculation!

2-1-2 Use the R function moran to compute Moran's I and the kurtosis of the unemployment rate (u5EC) and vacancy rate (v5EC) with the

- contiguity-based weights matrix,
- distance-based weights matrix

and interpret the outcomes!

2-1-3 Test Moran's I of the unemployment rate (u5EC) and vacancy rate (v5EC) for significance with the row-standardized contiguity matrix

- under normality,
- under randomization,
- by Monte Carlo simulation!

How can the results be interpreted?

2-1-4 Create a Moran scatterplot of the unemployment rate (u5EC) and interpret it!

2-1-5 Compute Geary's c for the unemployment rate (u5EC) and vacancy rate (v5EC) and test the significance by Monte Carlo simulation! Compare the outcomes with those obtained for Moran's I!

2-1-6 Compute the Getis-Ord global G statistic for the unemployment rate (u5EC) and vacancy rate (v5EC) and interpret it! Are the figures statistically significant?

2-1-7 Create spatial correlograms of Moran's I for the unemployment rate (u5EC) and vacancy rate (v5EC) with a maximum lag order of 2!

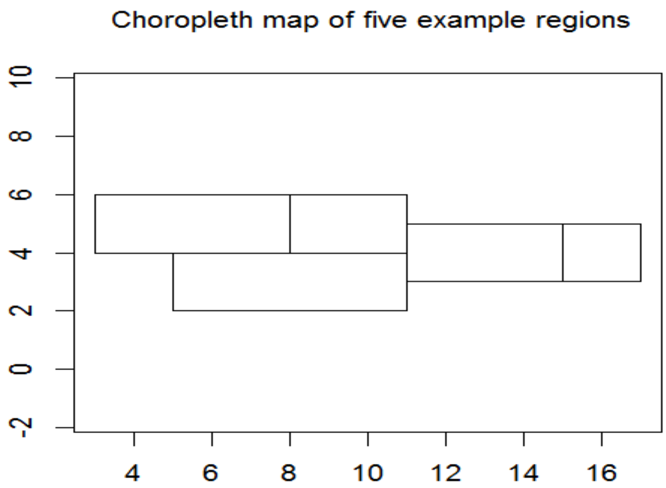
2.2 Local indicators of spatial autocorrelation
2.2.1 Choropleth maps of irregular grids

Coordinates of regions boundaries: textfile R1_R5ppt_coord.txt

Contents of coordinates file R1_R5ppt_coord.txt				
3.0 6.0 8.0 6.0 8.0 4.0 5.0 4.0 3.0 4.0 3.0 6.0	8.0 6.0 11.0 6.0 11.0 5.0 11.0 4.0 8.0 4.0 8.0 6.0	5.0 4.0 8.0 4.0 11.0 4.0 11.0 3.0 11.0 2.0 5.0 2.0 5.0 4.0	11.0 5.0 15.0 5.0 15.0 3.0 11.0 3.0 11.0 4.0 11.0 5.0	15.0 5.0 17.0 5.0 17.0 3.0 15.0 3.0 15.0 5.0

Commands for creating a choropleth map of five example regions in R script file SpatPoly5EX.R:

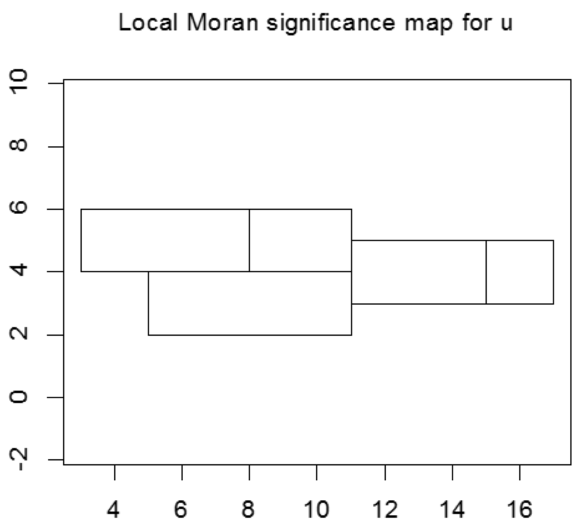
```
# SpatialPolygonsDataFrame object from textfiles of coordinates
# ppt example of 5 regions
R1_R5ppt.coord = read.table("R1_R5ppt_coord.txt")
# Extract the coordinates and create Polygons objects
R1ppt.poly = Polygon(R1_R5ppt.coord[1:6,])
R1ppt.polys = Polygons(list(Polygon(R1ppt.poly)), ID="R1")
R2ppt.polys = Polygons(list(Polygon(R1_R5ppt.coord[7:12,])), ID="R2")
R3ppt.polys = Polygons(list(Polygon(R1_R5ppt.coord[13:19,])), ID="R3")
R4ppt.polys = Polygons(list(Polygon(R1_R5ppt.coord[20:25,])), ID="R4")
R5ppt.polys = Polygons(list(Polygon(R1_R5ppt.coord[26:30,])), ID="R5")
# Converting the Polygons objects into a SpatialPolygons object
R1_R5ppt.SP =
SpatialPolygons(list(R1ppt.polys,R2ppt.polys,R3ppt.polys,R4ppt.polys,R5ppt.polys))
# Drawing a map of the regional system

> plot(R1_R5ppt.SP, main=" Choropleth map of five example regions", cex.main=1.0,
font.main=1, axes=TRUE)
```



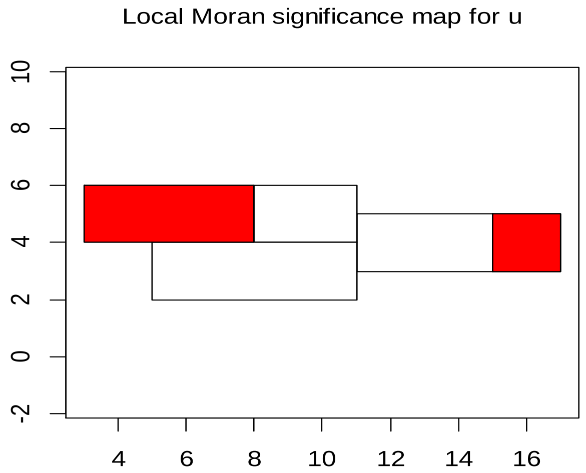
Contents of file	
R1ppt.poly	R1ppt.polys
An object of class "Polygon" Slot "labpt": [1] 5.5 5.0 Slot "area": [1] 10 Slot "hole": [1] FALSE Slot "ringDir": [1] 1 Slot "coords": V1 V2 1 3 6 2 8 6 3 8 4 4 5 4 5 3 4 6 3 6	An object of class "Polygons" Slot "Polygons": [[1]] An object of class "Polygon" Slot "labpt": [1] 5.5 5.0 Slot "area": [1] 10 Slot "hole": [1] FALSE Slot "ringDir": [1] 1 Slot "coords": V1 V2 1 3 6 2 8 6 3 8 4 4 5 4 5 3 4 6 3 6 Slot "plotOrder": [1] 1 Slot "labpt": [1] 5.5 5.0 Slot "ID": [1] "R1" Slot "area": [1] 10

2.2.2 Local Moran coefficient and test

```
# Performing local Moran tests
> u5EX.Iitest = localmoran(u5EX, listw=W5EX.lw, alternative = "greater")
> class(u5EX.Iitest)
[1] "localmoran" "matrix"
> str(u5EX.Iitest)
 localmoran [1:5, 1:5] 0.625 0.139 0.139 0.139 1.25 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:5] "1" "2" "3" "4" ...
  ..$ : chr [1:5] "Ii" "E.Ii" "Var.Ii" "Z.Ii" ...
 - attr(*, "call")= language localmoran(x = u5EX, listw = W5EX.lw, alternative =
"greater")
 - attr(*, "class")= chr [1:2] "localmoran" "matrix"
> u5EX.Iitest
         Ii  E.Ii    Var.Ii     Z.Ii   Pr(z > 0)
1 0.6250000 -0.25 0.2890625 1.627467 0.05181898
2 0.1388889 -0.25 0.1197917 1.123601 0.13059110
3 0.1388889 -0.25 0.1197917 1.123601 0.13059110
4 0.1388889 -0.25 0.1197917 1.123601 0.13059110
5 1.2500000 -0.25 0.7968750 1.680336 0.04644597
attr(,"call")
localmoran(x = u5EX, listw = W5EX.lw, alternative = "greater")
attr(,"class")
[1] "localmoran" "matrix"
> u5EX.localIi = u5EX.Iitest[1:5,1]
> class(u5EX.localIi)
[1] "numeric"
> u5EX.localIi
        1         2         3         4         5
0.6250000 0.1388889 0.1388889 0.1388889 1.2500000

# Drawing a map of the regional system
> plot(R1_R5ppt.SP, main="Local Moran significance map for u", cex.main=1.0,
font.main=1, axes=TRUE)
```



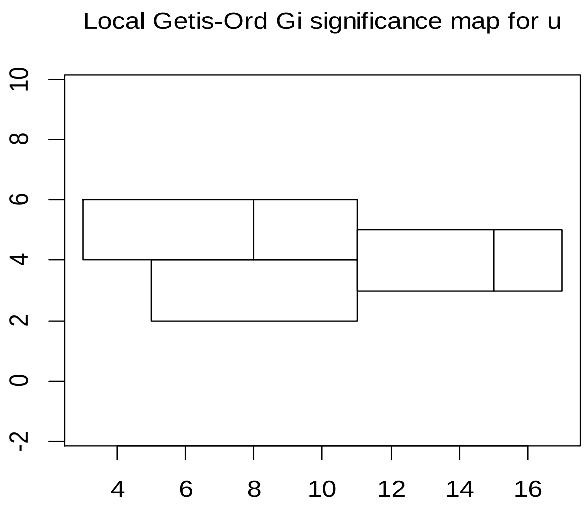
```
# Identifying critical Ii values (p<0.10)
> u5EX.Iisign = u5EX.Iitest[1:5,5]<0.10
> u5EX.Iisign
    1     2     3     4     5
 TRUE FALSE FALSE FALSE  TRUE

# Marking regions with significant Ii values (p<0.10) by red areas
> plot(R1_R5ppt.SP[u5EX.Iisign, ], col="red", add=TRUE)
```



2.2.3 Local Getis-Ord Gi and Gi* statistics

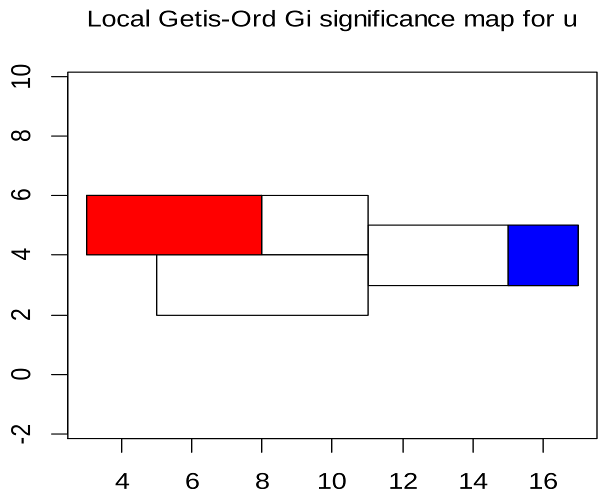
```
# Creating a binary weights list object with zero diagonal elements
> class(W5EX.nb)
[1] "nb"
> summary(W5EX.nb)
Neighbour list object:
Number of regions: 5
Number of nonzero links: 12
Percentage nonzero weights: 48
Average number of links: 2.4

Link number distribution:
1 2 3
1 1 3
1 least connected region:
5 with 1 link
3 most connected regions:
2 3 4 with 3 links
> WS5EX.lwdiag0 = nb2listw(neighbours=W5EX.nb, style="B")
> class(WS5EX.lwdiag0)
[1] "listw" "nb"
> summary(WS5EX.lwdiag0)
Characteristics of weights list object:
Neighbour list object:
Number of regions: 5
Number of nonzero links: 12
Percentage nonzero weights: 48
```

```
Average number of links: 2.4
Link number distribution:
1 2 3
1 1 3
1 least connected region:
5 with 1 link

3 most connected regions:
2 3 4 with 3 links
Weights style: B
Weights constants summary:
  n nn  S0 S1 S2
B 5 25 12 24 128

# Performing local Getis-Ord Gi tests
> u5EX.Gi = localG(u5EX, listw=WS5EX.lwdiag0)
or
> u5EX.Gi = localG(u5EX, listw=W5EX.lw)
> u5EX.Gi
[1]  1.697749  1.153113  1.153113 -1.147079 -1.540308
attr(,"gstari")
[1] FALSE
attr(,"call")
localG(x = u5EX, listw = WS5EX.lwdiag0)
attr(,"class")
[1] "localG"

# Drawing a map of the regional system
> plot(R1_R5ppt.SP, main="Local Getis-Ord Gi significance map for u",
cex.main=1.0, font.main=1, axes=TRUE)
```




```
# Identifying critical Gi values (p<0.10)
> zkrit10 = qnorm(0.90)
> zkrit10
[1] 1.281552

> u5EX.GisignH = u5EX.Gi>zkrit10
```

```
> u5EX.GisignH
[1]  TRUE FALSE FALSE FALSE FALSE
> u5EX.GisignL = -zkrit10>u5EX.Gi
> u5EX.GisignL
[1] FALSE FALSE FALSE FALSE  TRUE

# Plotting Gi hot spots in red
> plot(R1_R5ppt.SP[u5EX.GisignH, ], col="red", add=TRUE)
# Plotting Gi cold spots in blue
> plot(R1_R5ppt.SP[u5EX.GisignL, ], col="blue", add=TRUE)
```

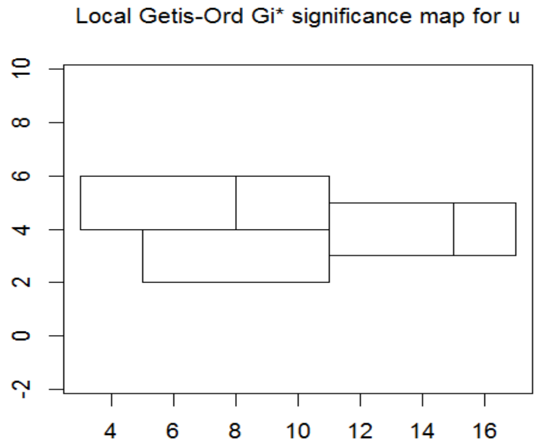



```
# Creating a binary weights list object with diagonal elements of one
> WS5EX.lwdiag1 = nb2listw(include.self(W5EX.nb), style="B")
> class(WS5EX.lwdiag1)
[1] "listw" "nb"
> summary(WS5EX.lwdiag1)
Characteristics of weights list object:
Neighbour list object:
Number of regions: 5
Number of nonzero links: 17
Percentage nonzero weights: 68
Average number of links: 3.4
Link number distribution:
2 3 4
1 1 3
1 least connected region:
5 with 2 links
3 most connected regions:
2 3 4 with 4 links
Weights style: B
Weights constants summary:
  n nn S0 S1  S2
B 5 25 17 34 244

# Performing local Getis-Ord Gi* tests
> u5EX.Gstari = localG(u5EX, listw=WS5EX.lwdiag1)
> u5EX.Gstari
[1]  1.863390  1.369306  1.369306 -1.369306 -1.863390
```

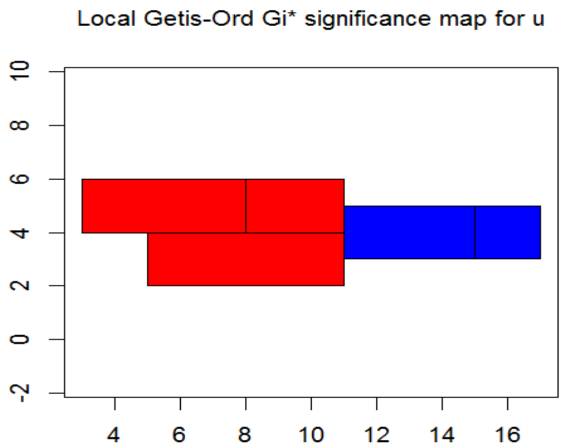
attr(,"gstari")
[1] TRUE
attr(,"call")

localG(x = u5EX, listw = WS5EX.lwdiag1)
attr(,"class")
[1] "localG"

```
# Drawing a map of the regional system
> plot(R1_R5ppt.SP, main="Local Getis-Ord Gi* significance map for u",
cex.main=1.0, font.main=1, axes=TRUE)
```




```
# Identifying critical Gi* values (p<0.10)
> zkrit10 = qnorm(0.90)
> zkrit10
[1] 1.281552
> u5EX.GstarisignH = u5EX.Gstari>zkrit10
> u5EX.GstarisignH
[1]  TRUE  TRUE  TRUE FALSE FALSE
> u5EX.GstarisignL = -zkrit10>u5EX.Gstari
> u5EX.GstarisignL
[1] FALSE FALSE FALSE TRUE TRUE

# Plotting Gi* hot spots in red
> plot(R1_R5ppt.SP[u5EX.GstarisignH, ], col="red", add=TRUE)
# Plotting Gi* cold spots in blue
> plot(R1_R5ppt.SP[u5EX.GstarisignL, ], col="blue", add=TRUE)
```



Exercises

2-2-1 The coordinates of regions boundaries are given as follows:

Contents of coordinates file R1_R5ec_coord.txt				
3.0,6.0 8.0,6.0 8.0,4.0 5.5,4.0 3.0,4.0 3.0,6.0	2.0,4.0 3.0,4.0 5.5,4.0 5.5,2.0 2.0,2.0 2.0,4.0	5.5,4.0 8.0,4.0 10.0,4.0 10.0,2.0 9.0,2.0 5.5,2.0 5.5,4.0	1.0,2.0 2.0,2.0 5.5,2.0 5.5,0.0 1.0,0.0 1.0,2.0	5.5,2.0 9.0,2.0 9.0,0.0 5.5,0.0 5.5,2.0

Store the coordinates in a textfile R1_R5ec_coord.txt and create a SpatialPolygonsDataFrame object R1_R5ec_coord! Plot the choropleth map for the exercise regions!

2-2-2 Compute local Moran coefficients of the unemployment rate (u5EC) and the vacancy rate (v5EC) and test them for significance. Interpret the output tables!

2-2-3 Create a choropleth map of significant I_i values ($p<0.10$) for the unemployment rate (u5EC) and the vacancy rate (v5EC)!

2-2-4 Create a binary weights list object of the exercise regions and compute local Getis-Ord Gi* statistics for the unemployment rate (u5EC) and the vacancy rate (v5EC)! Test the Gi* values for significance and interpret the output tables!

2-2-5 Create a choropleth map of Gi* hot spots (red) and cold spots (blue) ($p<0.10$) for the unemployment rate (u5EC) and the vacancy rate (v5EC)!

3. Standard regression model and spatial dependence tests

3.1 Standard regression model

The standard regression model is given by

(1) $y_r = \beta_0 + \sum_{j=1}^{m} \beta_j \cdot x_{jr} + \varepsilon_r$, r=1,2,…n,

where y is the dependent variable, x_j the j*th* explanatory variable, j=1,2,…,m, and r the region index. β_0 is the intercept and β_j are the regression coefficients of texplanatory variables x_j. Under the standard assumptions, the disturbances ε has an expectation of zero, a constant varianc (homoscedasticity) and free of autocor-relation.

Regression of productivity growth on output growth (Verdoorn's law)

- with matrix operations

Ordinary least-squares (OLS) estimator:

```
> n = length(gx5EX)
>n
[1] 5
>one = rep(1,n)
> one
[1] 1 1 1 1 1
> X5EX = cbind(one, gx5EX)
> X5EX
     one gx5EX
[1,]   1   0.6
[2,]   1   1.0
[3,]   1   1.6
[4,]   1   2.6
[5,]   1   2.2
> XtX5EX = t(X5EX)%*%X5EX
> XtX5EX
         one  gx5EX
one       5    8.00
gx5EX     8   15.52
> XtX5EXi = solve(XtX5EX)
> XtX5EXi
                 one         gx5EX
one        1.1411765   -0.5882353
gx5EX     -0.5882353    0.3676471
> Xtgy5EX = t(X5EX)%*%gy5EX
> Xtgy5EX
           [,1]
one        4.20
gx5EX      7.78
> b = XtX5EXi%*%Xtgy5EX
```

```
> b
             [,1]
one      0.2164706
gx5EX 0.3897059
```

- with R function lm

Ordinary least-squares (OLS) estimator, significance tests and goodness of fit:

```
> Verdoorn.lm = lm(gy5EX~gx5EX)

> class(Verdoorn.lm)
[1] "lm"
> summary(Verdoorn.lm)
Call:
lm(formula = gy5EX ~ gx5EX)

Residuals:
       1          2          3          4          5
-0.050294 -0.006176  0.060000 -0.129706  0.126176

Coefficients:
            Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  0.21647     0.12166    1.779    0.173
gx5EX        0.38971     0.06906    5.643    0.011 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1139 on 3 degrees of freedom
Multiple R-squared:  0.9139,    Adjusted R-squared:  0.8852
F-statistic: 31.85 on 1 and 3 DF,  p-value: 0.01101
```

- **Computing general statistics and overall test**

Maximum likelihood (ML) estimator of the error variance:

```
> n = length(Verdoorn.lm$residuals)
> n
[1] 5
> se2.ML = sum(Verdoorn.lm$residuals^2)/n
> se2.ML
[1] 0.007782353
```

Unbiased estimator of the error variance:

```
> Verdoorn.lm.df.resid = Verdoorn.lm$df.residual
> Verdoorn.lm.df.resid
[1] 3
> se2 = sum(Verdoorn.lm$residuals^2)/Verdoorn.lm.df.resid
or
> se2 = (summary(Verdoorn.lm)$sigma)^2
```

```
> se2
[1] 0.01297059
```

Standard error of regression (SER):

```
> SER = sqrt(se2)
```

or

```
> SER = summary(Verdoorn.lm)$sigma
> SER
[1] 0.1138885
```

Coefficient of determination (R^2):

```
# The R function var uses the denominator n-1
> R2 = 1- var(Verdoorn.lm$residuals)/var(Verdoorn.lm$model$gy5EX)
```

or

```
> R2 = 1- var(Verdoorn.lm$residuals)/var(gy5EX)
```

or

```
> R2 = summary(Verdoorn.lm)$r.squared
> R2
[1] 0.913912
```

Adjusted coefficient of determination:

```
> R2adj = 1 - (sum(Verdoorn.lm$residual^2)/Verdoorn.lm.df.resid)/
var(Verdoorn.lm$model$gy5EX)
```

or

```
> R2adj = 1 - (sum(Verdoorn.lm$residual^2)/Verdoorn.lm$df.residual)/var(gy5EX)
```

or

```
> R2adj = 1 - (1 - R2)*(n-1)/Verdoorn.lm$df.residual
```

or

```
> R2adj = summary(Verdoorn.lm)$adj.r.squared
> R2adj
[1] 0.885216
```

F test on overall fit:

```
> summary(Verdoorn.lm)$fstatistic
   value    numdf    dendf
31.84807  1.00000  3.00000
> fstat = summary(Verdoorn.lm)$fstatistic[1]
> fstat
   value
31.84807
> f0.95 = qf(0.95,1,3)    # 95% quantile of F distribution with 1 and 3 df
> f0.95
[1] 10.12796
> f0.99 = qf(0.99,1,3)    # 99% quantile of F distribution with 1 and 3 df
> f0.99
[1] 34.11622
> pvalue.fstat = 1 - pf(fstat,1,3)     # Actual significance level
> pvalue.fstat
     value
0.01101058
```

- Computing logLik, AIC and BIC

Logarithmic likelihood (logLik) with maximum likelihood estimators for the regression coefficients β_1, β_2, .., β_k and the error variance σ^2:

$$(3.1.1) \quad \log \mathrm{Lik} = -\frac{n}{2}\log(2\cdot\pi) - \frac{n}{2} - \frac{n}{2}\log(s_e^2)$$

```
> n = length(Verdoorn.lm$residuals)
> se2.ML = sum(Verdoorn.lm$residuals^2)/n
> se2.ML
[1] 0.007782353
> logLik.Verdlm = -(n/2)*log(2*pi)-n/2-(n/2)*log(se2.ML)
> logLik.Verdlm
[1] 5.045049
```

Logarithmic likelihood (logLik) with R function logLik:

```
> logLik.Verdlm = logLik(Verdoorn.lm)
> logLik.Verdlm
'log Lik.' 5.045049 (df=3)
> class(logLik.Verdlm)
[1] "logLik"
> logLik.Verdlm = as.numeric(logLik(Verdoorn.lm))
> class(logLik.Verdlm)
[1] "numeric"
> logLik.Verdlm
[1] 5.045049
```

Table 3.1.1: Akaike's information criterion (AIC) and Bayes' information criterion (BIC)

Information criteria	
Akaike's information criterion (AIC)	Bayes' information criterion (BIC)
$\mathrm{AIC} = -2\cdot\mathrm{logLik} + 2\cdot\mathrm{npar}$	$\mathrm{BIC} = -2\cdot\mathrm{logLik} + \mathrm{npar}\cdot\log(n)$
$\mathrm{AIC} = \mathrm{const.} + n\cdot\log(s_e^2) + 2\cdot\mathrm{npar}$	$\mathrm{BIC} = \mathrm{const.} + n\cdot\log(s_e^2) + \mathrm{npar}\cdot\log(n)$

npar: number of estimated parameters (regression coefficients and error variance)

Computation of AIC and BIC with logLik:

```
> npar = length(Verdoorn.lm$coefficients) + 1
> npar
[1] 3
> AIC.Verdlm = -2*logLik.Verdlm + 2*npar
> AIC.Verdlm
[1] -4.090097
> n
[1] 5
```

```
> BIC.Verdlm = -2*logLik.Verdlm + log(n)*npar
> BIC.Verdlm
[1] -5.261784
```

AIC and BIC with R functions:

```
> AIC.Verdlm = AIC(Verdoorn.lm)
[1] -4.090097
> BIC.Verdlm = BIC(Verdoorn.lm)
[1] -5.261784
```

3.2 Tests on spatial dependence in the errors

3.2.1 Moran test

• Moran test of residuals of the Verdoorn model using the normal approximation

```
> Verdoorn.lmmoran = lm.morantest(Verdoorn.lm, listw=W5EX.lw,
alternative="two.sided")
> Verdoorn.lmmoran

        Global Moran I for regression residuals

data:
model: lm(formula = gy5EX ~ gx5EX)
weights: W5EX.lw

Moran I statistic standard deviate = -1.8149, p-value = 0.06953
alternative hypothesis: two.sided
sample estimates:
Observed Moran I      Expectation        Variance
      -0.7447146        -0.4436275       0.0275205
```

• Moran permuation test applied to residuals

```
> set.seed(12345)
> Verdoorn.moran.mc_err = moran.mc(residuals(Verdoorn.lm), listw=W5EX.lw,
nsim=99, alternative="greater")
> Verdoorn.moran.mc_err

        Monte-Carlo simulation of Moran I

data:  residuals(Verdoorn.lm)
weights: W5EX.lw
number of simulations + 1: 100

statistic = -0.7447, observed rank = 1, p-value = 0.99
alternative hypothesis: greater

> set.seed(12345)
```

```
> Verdoorn.moran.mc_err = moran.mc(residuals(Verdoorn.lm), listw=W5EX.lw,
nsim=99, alternative="less")
> Verdoorn.moran.mc_err

        Monte-Carlo simulation of Moran I

data:  residuals(Verdoorn.lm)
weights: W5EX.lw
number of simulations + 1: 100

statistic = -0.7447, observed rank = 1, p-value = 0.01
alternative hypothesis: less
```

3.2.2 Lagrange multiplier test for spatial error dependence

• Standard LM error test of residuals of the Verdoorn model

```
> Verdoorn.LMerr = lm.LMtests(residuals(Verdoorn.lm), listw=W5EX.lw,
test="LMerr")
or
> Verdoorn.LMerr = lm.LMtests(Verdoorn.lm$resid, listw=W5EX.lw, test="LMerr")
or
> Verdoorn.LMerr = lm.LMtests(Verdoorn.lm, listw=W5EX.lw, test="LMerr")
> Verdoorn.LMerr

        Lagrange multiplier diagnostics for spatial dependence

data:
residuals: Verdoorn.lm$resid
weights: W5EX.lw

LMErr = 3.0811, df = 1, p-value = 0.07921
```

• Robust LM error test of residuals of the Verdoorn model

```
> Verdoorn.RLMerr = lm.LMtests(Verdoorn.lm, listw=W5EX.lw, test="RLMerr")
> Verdoorn.RLMerr

        Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = gy5EX ~ gx5EX)
weights: W5EX.lw

RLMerr = 5.4435, df = 1, p-value = 0.01964
```

3.2.3 Lagrange multiplier test for spatial lag dependence

• Standard LM lag test of residuals of the Verdoorn model

```
> Verdoorn.LMlag = lm.LMtests(Verdoorn.lm, listw=W5EX.lw, test="LMlag")
> Verdoorn.LMlag

        Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = gy5EX ~ gx5EX)
weights: W5EX.lw

LMlag = 0.3825, df = 1, p-value = 0.5363
```

• Robust LM lag test of residuals of the Verdoorn model

```
> Verdoorn.RLMlag = lm.LMtests(Verdoorn.lm, listw=W5EX.lw, test="RLMlag")
> Verdoorn.RLMlag

        Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = gy5EX ~ gx5EX)
weights: W5EX.lw

RLMlag = 2.7449, df = 1, p-value = 0.09756
```

Exercises

3-1 For the five exercise regions, geo-referenced data on the unemployment rate (u) and the vacancy rate (v) are given:

region	u	v
1	6	3
2	8	3
3	8	2
4	11	1
5	12	1

Regress the unemployment rate (u) on the vacancy rate (v) (Beveridge curve)! Compute the OLS-estimators of the regression coefficients

- by matrix operations,
- with the lm function

and interpret them (α=0.05)!

3-2 Compute Moran's I for the residuals of the OLS estimated Beveridge curve and interpret it descriptively!

3-3 Test Moran's I of the residuals of the standard Beveridge curve regression for significance (α=0.05) using the

- normal approximation,
- permutation approach!

Compare your findings!

3-4 Test the residuals of the OLS estimated Beveridge curve for spatial error dependence using the traditional and robust LM test (α=0.05)!

3-5 Test the residuals of the OLS estimated Beveridge curve for spatial lag dependence using the traditional and robust LM test (α=0.05)!

4. Spatial regression models
4.1 SLX model

The spatial cross-regressive model (SLX model) is given by

(2) $$y_r = \beta_0 + \sum_{j=1}^{m} \beta_j \cdot x_{jr} + \sum_{j=1}^{m} \theta_j \cdot SL(x_{jr}) + \varepsilon_r .$$

$SL(x_{jr})$ is the spatial lag of the j*th* explanatory variable and θ_j the corresconding regression coefficient.

Regression of productivity growth on output growth and its spatial lag (spatialized law of Verdoorn)

- with matrix operations

Ordinary least-squares (OLS) estimator:

```
> XS5EX = cbind(one, gx5EX, Lgx5EX)
> XS5EX
     one gx5EX Lgx5EX
[1,]   1   0.6    1.3
[2,]   1   1.0    1.6
[3,]   1   1.6    1.4
[4,]   1   2.6    1.6
[5,]   1   2.2    2.6
> XStXS5EX = t(XS5EX)%*%XS5EX
> XStXS5EX
       one gx5EX Lgx5EX
one    5.0  8.00   8.50
gx5EX  8.0 15.52  14.50
Lgx5EX 8.5 14.50  15.53
> XStXS5EXi = solve(XStXS5EX)
> XStXS5EXi
               one       gx5EX     Lgx5EX
one     2.89298740 -0.09306261 -1.4965219
gx5EX  -0.09306261  0.50761421 -0.4230118
Lgx5EX -1.49652190 -0.42301184  1.2784358
> XStgy5EX = t(XS5EX)%*%gy5EX
> XStgy5EX
       [,1]
one    4.20
gx5EX  7.78
Lgx5EX 7.62

> b.SLX = XStXS5EXi%*%XStgy5EX
> b.SLX
             [,1]
one    0.02302312
gx5EX  0.33502538
Lgx5EX 0.16525663
```

- with R function lm

Ordinary least-squares (OLS) estimator, significance tests and goodness of fit:

```
> Verdoorn.SLX = lm(gy5EX~gx5EX+Lgx5EX)

> summary(Verdoorn.SLX)
Call:
lm(formula = gy5EX ~ gx5EX + Lgx5EX)

Residuals:
       1        2        3        4        5
-0.03887 -0.02246  0.10958 -0.05850  0.01025

Coefficients:
            Estimate  Std. Error t value   Pr(>|t|)
(Intercept)  0.02302   0.15933   0.145   0.8984
gx5EX        0.33503   0.06674   5.020   0.0375 *
Lgx5EX       0.16526   0.10592   1.560   0.2591
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.09367 on 2 degrees of freedom
Multiple R-squared:  0.9612,    Adjusted R-squared:  0.9223
F-statistic: 24.76 on 2 and 2 DF,  p-value: 0.03883
```

- **Computing general statistics and overall test**

Maximum likelihood (ML) estimator of the error variance:

```
> n = length(Verdoorn.SLX$residuals)
> n
[1] 5
> se2.ML = sum(Verdoorn.SLX$residuals^2)/n
> se2.ML
[1] 0.003509983
```

Unbiased estimator of the error variance:

```
> Verdoorn.SLX.df.resid = Verdoorn.SLX$df.residual
> Verdoorn.SLX.df.resid
[1] 2
> se2 = sum(Verdoorn.SLX$residuals^2)/Verdoorn.SLX.df.resid
or
> se2 = (summary(Verdoorn.SLX)$sigma)^2
> se2
[1] 0.008774958
```

Standard error of regression (SER):

```
> SER = sqrt(se2)
or
> SER = summary(Verdoorn.SLX)$sigma
```

```
> SER
[1] 0.0937
```

Coefficient of determination (R^2):

```
# The R function var uses the denominator n-1
> R2 = 1- var(Verdoorn.SLX$residuals)/var(Verdoorn.SLX$model$gy5EX)
or
> R2 = 1- var(Verdoorn.SLX$residuals)/var(gy5EX)
or
> R2 = summary(Verdoorn.SLX)$r.squared
> R2
[1] 0.9611728
```

Adjusted coefficient of determination:

```
> R2adj = 1 - (sum(Verdoorn.SLX$residual^2)/Verdoorn.SLX.df.resid)/
var(Verdoorn.SLX$model$gy5EX)
or
> R2adj = 1 - (sum(Verdoorn.SLX$residual^2)/Verdoorn.SLX$df.residual)/var(gy5EX)
or
> R2adj = 1 - (1 - R2)*(n-1)/Verdoorn.SLX$df.residual
or
> R2adj = summary(Verdoorn.SLX)$adj.r.squared
> R2adj
[1] 0.9223455
```

- **logLik, AIC and BIC**

```
> logLik.VerdSLX = logLik(Verdoorn.SLX)
> logLik.VerdSLX
'log Lik.' 7.035667 (df=4)

> AIC.VerdSLX = AIC(Verdoorn.SLX)
> AIC.VerdSLX
[1] -6.071335
> BIC.VerdSLX = BIC(Verdoorn.SLX)
> BIC.VerdSLX
[1] -7.633583
```

- Moran test

Moran test for residuals:

```
> Verdoorn.SLXmoran = lm.morantest(Verdoorn.SLX, listw=W5EX.lw,
alternative="two.sided")
> Verdoorn.SLXmoran

        Global Moran I for regression residuals

data:
model: lm(formula = gy5EX ~ gx5EX + Lgx5EX)
weights: W5EX.lw
```

```
Moran I statistic standard deviate = -0.7102, p-value = 0.4776
alternative hypothesis: two.sided
sample estimates:
Observed Moran I      Expectation        Variance
    -0.493454111     -0.439650310      0.005739908
```

- Lagrange multiplier test for spatial error dependence

<u>• Standard LM error test of residuals of the spatialized SLX Verdoorn model</u>

```
> Verdoorn.SLXLMerr = lm.LMtests(Verdoorn.SLX, listw=W5EX.lw, test="LMerr")
> Verdoorn.SLXLMerr

        Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = gy5EX ~ gx5EX + Lgx5EX)
weights: W5EX.lw
LMErr = 1.3528, df = 1, p-value = 0.2448
```

<u>• Robust LM error test of residuals of the spatialized SLX Verdoorn model</u>

```
> Verdoorn.SLXRLMerr = lm.LMtests(Verdoorn.SLX, listw=W5EX.lw, test="RLMerr")
> Verdoorn.SLXRLMerr

        Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = gy5EX ~ gx5EX + Lgx5EX)
weights: W5EX.lw
RLMerr = 2.9657, df = 1, p-value = 0.08505
```

- Lagrange multiplier test for spatial lag dependence

<u>• Standard LM lag test of residuals of the spatialized SLX Verdoorn model</u>

```
> Verdoorn.SLXLMlag = lm.LMtests(Verdoorn.SLX, listw=W5EX.lw, test="LMlag")
> Verdoorn.SLXLMlag

        Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = gy5EX ~ gx5EX + Lgx5EX)
weights: W5EX.lw
LMlag = 2.4009, df = 1, p-value = 0.1213
```

• Robust LM lag test of residuals of the spatialized SLX Verdoorn model

```
> Verdoorn.SLXRLMlag = lm.LMtests(Verdoorn.SLX, listw=W5EX.lw, test="RLMlag")
> Verdoorn.SLXRLMlag

        Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = gy5EX ~ gx5EX + Lgx5EX)
weights: W5EX.lw
RLMlag = 4.0138, df = 1, p-value = 0.04513
```

Exercises

4-1-1 Regress the unemployment rate (u) on the vacancy rate (v) and its spatial lag (SLX model of the Beveridge curve)! Compute the OLS-estimators of the regression coefficients

- by matrix operations,
- with the lm function

and interpret them (α=0.05)!

4-1-2 Test Moran's I of the residuals of the SLX model of Beveridge curve regression for significance (α=0.05) using the

- normal approximation,
- permutation approach!

Compare your findings!

4-1-3 Test the residuals of the OLS estimated SLX Beveridge curve for spatial error dependence using the traditional and robust LM test (α=0.05)!

4-1-4 Test the residuals of the OLS estimated SLX Beveridge curve for spatial lag dependence using the traditional and robust LM test (α=0.05)!

4.2 Spatial lag model

The spatial autoregressive (SAR) model (spatial lag model) is given by

(3) $$y_r = \beta_0 + \lambda \cdot SL(y_r) + \sum_{j=1}^{m} \beta_j \cdot x_{jr} + \varepsilon_r \,.$$

$SL(y_r)$ is the spatial lag of the dependent variable and the corresponding regression coefficient λ the spatial autoregressive parameter.

Regression of productivity growth on its spatial lag and output growth (spatialized law of Verdoorn)

```
> Verdoorn.sar = lagsarlm(gy5EX~gx5EX, listw=W5EX.lw ,type="lag")

> summary(Verdoorn.sar)
Call:lagsarlm(formula = gy5EX ~ gx5EX, listw = W5EX.lw, type = "lag")

Residuals:growth
      Min        1Q    Median       3Q       Max
-0.109224 -0.056418 -0.012722  0.075287  0.103076

Type: lag
Coefficients: (asymptotic standard errors)
             Estimate  Std. Error  z value   Pr(>|z|)
(Intercept)  0.028118   0.164543  0.1709    0.8643
gx5EX        0.354865   0.056629  6.2664    3.694e-10

Rho: 0.28717, LR test value: 0.86648, p-value: 0.35193
Asymptotic standard error: 0.21686
    z-value: 1.3242, p-value: 0.18543
Wald statistic: 1.7536, p-value: 0.18543

Log likelihood: 5.478291 for lag model
ML residual variance (sigma squared): 0.0063135, (sigma: 0.079458)
Number of observations: 5
Number of parameters estimated: 4
AIC: -2.9566, (AIC for lm: -4.0901)
LM test for residual autocorrelation
test value: 3.9562, p-value: 0.046698
```

- **Computing general statistics and overall test**

Maximum likelihood (ML) estimator of the error variance:

```
> n = length(Verdoorn.sar$residuals)
> n
[1] 5
> se2.ML = sum(Verdoorn.sar$residuals^2)/n
or
> se2.ML = sum(Verdoorn.sar$s2
> se2.ML
[1] 0.006313501
```

Unbiased estimator of the error variance:

```
> Verdoorn.sar.df.resid = n - (Verdoorn.sar$parameters - 1)
> Verdoorn.sar.df.resid
[1] 2
> se2 = sum(Verdoorn.sar$residuals^2)/Verdoorn.sar.df.resid
or
> se2 = Verdoorn.sar$SSE/Verdoorn.sar.df.resid
> se2
[1] 0.01578375
```

Standard error of regression (SER):
```
> SER = sqrt(se2)
> SER
[1] 0.1256334
```

Coefficient of determination (R^2):
```
# The R function var uses the denominator n-1
> R2 = 1- var(Verdoorn.sar$residuals)/var(Verdoorn.sar$y)
or
> R2 = 1- var(Verdoorn.sar$residuals)/var(gy5EX)
> R2
[1] 0.9301604
```

Adjusted coefficient of determination:
```
> R2adj = 1 -
(sum(Verdoorn.sar$residual^2)/Verdoorn.sar.df.resid)/var(Verdoorn.sar$y)
or
> R2adj = 1 - (sum(Verdoorn.sar$residual^2)/Verdoorn.sar.df.resid)/var(gy5EX)
or
> R2adj = 1 - (1 - R2)*(n-1)/Verdoorn.sar.df.resid
> R2adj
[1] 0.8603208
```

- **Computing LogLik, AIC and BIC**

```
> Verdoorn.sar$LL
         [,1]
[1,] 5.478291
> AIC.Verdsar = AIC(Verdoorn.sar)
> AIC.Verdsar
[1] -2.956582
> BIC.Verdsar = BIC(Verdoorn.sar)
> BIC.Verdsar
[1] -4.51883
```

• **Impact measures in Spatial Lag Model (Verdoorn's law)**

```
> Imp1Verdoorn.sar = impacts(Verdoorn.sar, listw=W5EX.lw)
> class(Imp1Verdoorn.sar)
[1] "lagImpact"
> Imp1Verdoorn.sar
Impact measures (lag, exact):
        Direct    Indirect     Total
x5EX 0.3683498 0.1294787 0.4978285
> I5 = diag(5)
> I5
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    1    0    0    0
```

```
[3,]    0    0    1    0    0
[4,]    0    0    0    1    0
[5,]    0    0    0    0    1
> I5_rhoW5EX = I5 - Verdoorn.sar$rho*W5EX
> I5_rhoW5EX
           1           2           3           4           5
1  1.00000000 -0.14358722 -0.14358722  0.00000000  0.00000000
2 -0.09572481  1.00000000 -0.09572481 -0.09572481  0.00000000
3 -0.09572481 -0.09572481  1.00000000 -0.09572481  0.00000000
4  0.00000000 -0.09572481 -0.09572481  1.00000000 -0.09572481
5  0.00000000  0.00000000  0.00000000 -0.28717444  1.00000000
> VW5EX = solve(I5_rhoW5EX)
> VW5EX
           1          2          3          4           5
1 1.032041506 0.16736259 0.16736259 0.03294722 0.003153866
2 0.111575061 1.03910964 0.12647174 0.11472893 0.010982405
3 0.111575061 0.12647174 1.03910964 0.11472893 0.010982405
4 0.021964811 0.11472893 0.11472893 1.05085245 0.100592656
5 0.006307732 0.03294722 0.03294722 0.30177797 1.028887640
> Verdoorn.sar$coeff[2]
    x5EX
0.3548649

> SW5EX = Verdoorn.sar$coeff[2]*VW5EX
> SW5EX
           1          2          3          4           5
1 0.366235270 0.05939110 0.05939110 0.01169181 0.001119196
2 0.039594069 0.36874350 0.04488038 0.04071327 0.003897270
3 0.039594069 0.04488038 0.36874350 0.04071327 0.003897270
4 0.007794540 0.04071327 0.04071327 0.37291061 0.035696799
5 0.002238393 0.01169181 0.01169181 0.10709040 0.365116074
> dir5EX = sum(diag(SW5EX))/5
> dir5EX
[1] 0.3683498
> tot5EX = sum(SW5EX)/5
> tot5EX
[1] 0.4978285
> ind5EX = tot5EX - dir5EX
> ind5EX
[1] 0.1294787
```

• Impact measures in Spatial Lag Model with significance test (with listw) (Verdoorn’s law)

```
> set.seed(12345)
> Imp2Verdoorn.sar = impacts(Verdoorn.sar, listw=W5EX.lw, R=1000)
> summary(Imp2Verdoorn.sar)
Impact measures (lag, exact):
        Direct    Indirect     Total
x5EX 0.3683498 0.1294787 0.4978285
===========================================================
```

```
Simulation results (asymptotic variance matrix):
Direct:
Iterations = 1:1000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
          Mean   SD Naive  SE Time-series      SE
x5EX   0.3794   0.05669       0.001793       0.001793

2. Quantiles for each variable:
         2.5%     25%   50%    75%   97.5%
x5EX 0.2742 0.3396 0.377 0.4163 0.4843
==========================================================
Indirect:
Iterations = 1:1000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
         Mean  SD Naive SE Time-series       SE
x5EX   0.158     0.1797       0.005684       0.005684

2. Quantiles for each variable:
            2.5%      25%      50%   75%   97.5%
x5EX -0.06158 0.05399 0.1222 0.211 0.6283
==========================================================
Total:
Iterations = 1:1000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
          Mean  SD Naive SE Time-series       SE
x5EX   0.5374    0.2031       0.006422       0.006422

2. Quantiles for each variable:
         2.5%     25%     50%    75%  97.5%
x5EX 0.3292 0.4198 0.4881 0.5935 1.078
```

• Impact measures in Spatial Lag Model with significance test (with "CsparseMatrix") (Verdoorn's law)

```
> W5EX.sparse = as(W5EX.lw, "CsparseMatrix")
> trMCEX = trW(W5EX.sparse, type="mult")
> set.seed(12345)
> Imp3Verdoorn.sar = impacts(Verdoorn.sar, tr=trMCEX, R=1000)
> Imp3Verdoorn.sumsar = summary(Imp3Verdoorn.sar, zstats=TRUE, short=TRUE)
> Imp3Verdoorn.sumsar
Impact measures (lag, trace):
            Direct      Indirect      Total
x5EX 0.3683498 0.1294787 0.4978285
[ With argument type="MC") in trW command slightly different results:
Impact measures (lag , trace):
            Direct      Indirect      Total
x5EX 0.3687716 0.1290569 0.4978285 ]
========================================================
Simulation results (asymptotic variance matrix):
========================================================
Simulated z-values:
           Direct    Indirect    Total
x5EX 6.712046 0.885079 2.668765
Simulated p-values:
            Direct     Indirect      Total
x5EX 1.9192e-11 0.37611  0.0076131
```

• Impact measures in Spatial Lag Model for neighbours up to 4th order (Verdoorn's law)

```
> W5EX.sparse = as(W5EX.lw, "CsparseMatrix")
> trMCEX = trW(W5EX.sparse, type="mult")
> ImpQ4AVerdoorn.sar = impacts(Verdoorn.sar, tr=trMCEX, Q=4)
> ImpQ4AVerdoorn.sar
Impact measures (lag, trace):
            Direct       Indirect     Total
x5EX 0.3684686 0.1293599 0.4978285
> str(ImpQ4AVerdoorn.sar)
List of 3
 $ direct  : num 0.368
 $ indirect: num 0.129
 $ total   : num 0.498
 - attr(*, "Qres")=List of 3
  ..$ direct  : num [1:4] 0.354865 0 0.011706 0.000883
  ..$ indirect: num [1:4] 0 0.10191 0.01756 0.00752
  ..$ total   : num [1:4] 0.3549 0.1019 0.0293 0.0084
 - attr(*, "method")= chr "trace"
 - attr(*, "type")= chr "lag"
 - attr(*, "bnames")= chr "x5EX"
 - attr(*, "haveQ")= logi TRUE
 - attr(*, "timings")= Named num [1:2] 0 0
```

```
  ..- attr(*, "names")= chr [1:2] "user.self" "elapsed"
 - attr(*, "class")= chr "lagImpact"
 - attr(*, "insert")= logi FALSE
 - attr(*, "iClass")= chr "sarlm"

> attr(ImpQ4AVerdoorn.sar, "Qres")
$direct
[1] 0.3548648653 0.0000000000 0.0117061631 0.0008834217
$indirect
[1] 0.000000000 0.101908120 0.017559245 0.007520855
$total
[1] 0.354864865 0.101908120 0.029265408 0.008404277

> sum(attr(ImpQ4AVerdoorn.sar, "Qres")$direct)
[1] 0.3674545
> ImpQ4AVerdoorn.sar$direct
[1] 0.3684686
> sum(attr(ImpQ4AVerdoorn.sar, "Qres")$direct)/ImpactQ4A.EXsar$direct
[1] 0.9972477
> sum(attr(ImpQ4AVerdoorn.sar, "Qres")$indirect)
[1] 0.1269882
> ImpQ4AVerdoor.sar$indirect
[1] 0.1293599
> sum(attr(ImpQ4AVerdoorn.sar, "Qres")$indirect)/ImpactQ4A.EXsar$indirect
[1] 0.9816659
> sum(attr(ImpQ4AVerdoorn.sar, "Qres")$total)
[1] 0.4944427
> ImpQ4AVerdoor.sar$total
[1] 0.4978285
> sum(attr(ImpQ4AVerdoorn.sar, "Qres")$total)/ImpactQ4A.EXsar$total
[1] 0.9931988

> Verdoorn.sar$coeff[2]
     x5EX
0.3548649

# Direct, indirect and total impact 1st power
> I5*Verdoorn.sar$coeff[2]
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.3548649 0.0000000 0.0000000 0.0000000 0.0000000
[2,] 0.0000000 0.3548649 0.0000000 0.0000000 0.0000000
[3,] 0.0000000 0.0000000 0.3548649 0.0000000 0.0000000
[4,] 0.0000000 0.0000000 0.0000000 0.3548649 0.0000000
[5,] 0.0000000 0.0000000 0.0000000 0.0000000 0.3548649
> dirimpVerd.Q1 = sum(diag(I5*Verdoorn.sar$coeff[2]))/5
> dirimpVerd.Q1
[1] 0.3548649
> totimpVerd.Q1 = sum(I5*Verdoorn.sar$coeff[2])/5
> totimpVerd.Q1
[1] 0.3548649
> indimpVerd.Q1 = totimpVerd.Q1 - dirimpVerd.Q1
```

```
> indimpVerd.Q1
[1] 0

# Direct, indirect and total impact 2nd power
> Verdoorn.sar$rho*W5EX*Verdoorn.sar$coeff[2]
           1          2          3          4          5
1 0.00000000 0.05095406 0.05095406 0.00000000 0.00000000
2 0.03396937 0.00000000 0.03396937 0.03396937 0.00000000
3 0.03396937 0.03396937 0.00000000 0.03396937 0.00000000
4 0.00000000 0.03396937 0.03396937 0.00000000 0.03396937
5 0.00000000 0.00000000 0.00000000 0.10190812 0.00000000
> dirimpVerd.Q2 = sum(diag(Verdoorn.sar$rho*W5EX*Verdoorn.sar$coeff[2]))/5
> dirimpVerd.Q2
[1] 0
> totimpVerd.Q2 = sum(Verdoorn.sar$rho*W5EX*Verdoorn.sar$coeff[2])/5
> totimpVerd.Q2
[1] 0.1019081
> indimpVerd.Q2 = totimpVerd.Q2 - dirimpVerd.Q2
> indimp.EXQ2
[1] 0.1019081

# Direct impact 3rd power
> W5EX2 = W5EX%*%W5EX
> Verdoorn.sar$rho^2*W5EX2*Verdoorn.sar$coeff[2]
            1           2           3           4           5
1 0.009755136 0.004877568 0.004877568 0.009755136 0.000000000
2 0.003251712 0.011380992 0.008129280 0.003251712 0.003251712
3 0.003251712 0.008129280 0.011380992 0.003251712 0.003251712
4 0.006503424 0.003251712 0.003251712 0.016258560 0.000000000
5 0.000000000 0.009755136 0.009755136 0.000000000 0.009755136
> dirimpVerd.Q3 = sum(diag(Verdoorn.sar$rho^2*W5EX2*Verdoorn.sar$coeff[2]))/5
> dirimpVerd.Q3
[1] 0.01170616
> totimpVerd.Q3 = sum(Verdoorn.sar$rho^2*W5EX2*Verdoorn.sar$coeff[2])/5
> totimp.EXQ3
[1] 0.02926541
> indimpVerd.Q3 = totimpVerd.Q3 - dirimpVerd.Q3
> indimpVerd.Q3
[1] 0.01755924

# Direct impact 4th power
> W5EX3 = W5EX2%*%W5EX
> Verdoorn.sar$rho^3*W5EX3*Verdoorn.sar$coeff[2]
             1            2            3            4            5
1 0.0009338086 0.0028014257 0.0028014257 0.0009338086 0.0009338086
2 0.0018676171 0.0015563476 0.0018676171 0.0028014257 0.0003112695
3 0.0018676171 0.0018676171 0.0015563476 0.0028014257 0.0003112695
4 0.0006225390 0.0028014257 0.0028014257 0.0006225390 0.0015563476
5 0.0018676171 0.0009338086 0.0009338086 0.0046690429 0.0000000000
> dirimpVerd.Q4 = sum(diag(Verdoorn.sar$rho^3*W5EX3*Verdoorn.sar$coeff[2]))/5
```

```
> dirimpVerd.Q4
[1] 0.0009338086    # Deviation from R value of 0.0008834217
> totimpVerd.Q4 = sum(Verdoorn.sar$rho^3*W5EX3*Verdoorn.sar$coeff[2])/5

> totimpVerd.Q4
[1] 0.008404277     # Identical with R value of 0.008404277
> indimpVerd.Q4 = totimpVerd.Q4 - dirimpVerd.Q4
> indimpVerd.Q4
[1] 0.007470469     # Deviation from R value of 0.007520855
```

• Impact measures in Spatial Lag Model for neighbours up to 4th order with significance test (Verdoorn)

```
> W5EX.sparse = as(W5EX.lw, "CsparseMatrix")
> set.seed(12345)
> trMCEX = trW(W5EX.sparse, type="MC")
> ImpQ4BVerd.sar = impacts(Verdoorn.sar, tr=trMCEX, R=1000, Q=4)
> ImpQ4BVerd.sumsar = summary(ImpactQ4B.EXsar, zstats=TRUE, reportQ=TRUE,
short=TRUE)
> ImpQ4BVerd.sumsar
Impact measures (lag, trace):
          Direct      Indirect     Total
x5EX 0.3684686 0.1293599 0.4978285
==================================
Impact components
$direct
        x5EX
Q1 0.3548648653
Q2 0.0000000000
Q3 0.0117061631
Q4 0.0008834217
$indirect
       x5EX
Q1 0.000000000
Q2 0.101908120
Q3 0.017559245
Q4 0.007520855
$total
       x5EX
Q1 0.354864865
Q2 0.101908120
Q3 0.029265408
Q4 0.008404277

=========================================================
Simulation results (asymptotic variance matrix):
=========================================================
Simulated z-values:
        Direct     Indirect     Total
x5EX 6.58114 0.8942537 2.668765
```

```
Simulated p-values:
         Direct      Indirect    Total
x5EX 4.6686e-11 0.37119  0.0076131
=============================================================
Simulated impact components z-values:
$Direct
      x5EX
Q1 6.1573673
Q2       NaN
Q3 0.9747153
Q4 0.6521343
$Indirect
      x5EX
Q1       NaN
Q2 1.3006856
Q3 0.9747153
Q4 0.6521343
$Total
      x5EX
Q1 6.1573673
Q2 1.3006856
Q3 0.9747153
Q4 0.6521343

Simulated impact components p-values:
$Direct
  x5EX
Q1 7.3964e-10
Q2 NA
Q3 0.32970
Q4 0.51431
$Indirect
  x5EX
Q1 NA
Q2 0.19337
Q3 0.32970
Q4 0.51431
$Total
  x5EX
Q1 7.3964e-10
Q2 0.19337
Q3 0.32970
Q4 0.51431
```

Exercises

4-2-1 Estimate the spatial lag model of the Beveridge curve by the method of maximum likelihood (ML) and interpret the regression coefficients (α=0.01)!

4-2-2 Determine impact measures for the spatial lag model of the Beveridge curve and interpret them (α=0.01)!

4-2-3 Compare the standard error of regression (SER) between the SLX and the spatial lag model!

4-2-4 Compare the Akaike information criterion (AIC) between the standard regression and spatial lag model and interpret it!

4.3 Spatial error model

The spatial error model (SEM) is given by

(4) $y_{rt} = \beta_0 + \sum_{j=1}^{m} \beta_j \cdot x_{jrt} + \varepsilon_{rt}$,

$$\varepsilon_{rt} = \rho \cdot SL(\varepsilon_{rt}) + v_{rt}$$

$SL(\varepsilon_{rt})$ is the spatial lag of the error ε and ρ spatial autoregressive coefficient of the disturbance process.

Regression of productivity growth on output growth with spatially autocorrelated errors (spatialized law of Verdoorn)

```
> Verdoorn.sem = errorsarlm(gy5EX~gx5EX, listw=W5EX.lw)
or
> Verdoorn.sem = errorsarlm(gy5EX~gx5EX, listw=W5EX.lw, method="eigen")

> summary(Verdoorn.sem)
Call:errorsarlm(formula = gy5EX ~ gx5EX, listw = W5EX.lw)

Residuals:
      Min        1Q     Median        3Q       Max
-0.0225187 -0.0196610  0.0098388  0.0134012  0.0189397
Type: error
Coefficients: (asymptotic standard errors)
              Estimate   Std. Error  z value   Pr(>|z|)
(Intercept) 0.2080323  0.0119740  17.374   < 2.2e-16
gx5EX       0.3852800  0.0069093  55.763   < 2.2e-16

Lambda: -1.1931, LR test value: 11.613, p-value: 0.00065488
Asymptotic standard error: 0.081045
    z-value: -14.721, p-value: < 2.22e-16
Wald statistic: 216.71, p-value: < 2.22e-16
```

Log likelihood: 10.85161 for error model
ML residual variance (sigma squared): 0.00030575, (sigma: 0.017486)
Number of observations: 5
Number of parameters estimated: 4
AIC: -13.703, (AIC for lm: -4.0901)

- **Computing general statistics and overall test**

Maximum likelihood (ML) estimator of the error variance:
> n = length(Verdoorn.sem$residuals)
> n
[1] 5
> se2.ML = sum(Verdoorn.sem$residuals^2)/n
or
> se2.ML = sum(Verdoorn.sem$s2
> se2.ML
[1] 0.0003057513

Unbiased estimator of the error variance:
> Verdoorn.sem.df.resid = n - (Verdoorn.sem$parameters - 1)
> Verdoorn.sem.df.resid
[1] 2
> se2 = sum(Verdoorn.sem$residuals^2)/Verdoorn.sem.df.resid
or
> se2 = Verdoorn.sem$SSE/Verdoorn.sem.df.resid
> se2
[1] 0.0007643782

Standard error of regression (SER):
> SER = sqrt(se2)
> SER
[1] 0.02764739

Coefficient of determination (R^2):
The R function var uses the denominator n-1
> R2 = 1- var(Verdoorn.sem$residuals)/var(Verdoorn.sem$y)
or
> R2 = 1- var(Verdoorn.sem$residuals)/var(gy5EX)
> R2
[1] 0.9966178

Adjusted coefficient of determination:
> R2adj = 1 -
(sum(Verdoorn.sem$residual^2)/Verdoorn.sem.df.resid)/var(Verdoorn.sem$y)
or
> R2adj = 1 - (sum(Verdoorn.sem$residual^2)/Verdoorn.sem.df.resid)/var(gy5EX)
or
> R2adj = 1 - (1 - R2)*(n-1)/Verdoorn.sem.df.resid
> R2adj
[1] 0.9932356

- **Computing logLik, AIC and BIC**

```
> logLik.Verdsem = Verdoorn.sem$LL
> logLik.Verdsem
        [,1]
[1,] 10.85161
> AIC.Verdsem = AIC(Verdoorn.sem)
> AIC.Verdsem
[1] -13.70323
> BIC.Verdsem = BIC(Verdoorn.sem)
> BIC.Verdsem
[1] -15.26548
```

Exercises

4-3-1 Estimate the spatial error model of the Beveridge curve by the method of maximum likelihood (ML) and interpret the regression coefficients (α=0.01)!

4-3-3 Compare the standard error of regression (SER) between the spatial lag and the spatial error model!

4-3-4 Compare the Akaike information criterion (AIC) between the standard regression and spatial error model and interpret it!

4.4 Spatial Durbin Model

The spatial Durbin model (SDM) is given by

(5) $$y_r = \beta_0 + \lambda \cdot SL(y_r) + \sum_{j=1}^{m} \beta_j \cdot x_{jr} + \sum_{j=1}^{m} \theta_j \cdot SL(x_{jr}) + \varepsilon_r \,.$$

Regression of productivity growth on its spatial lag and output growth and its spatial lag (spatialized law of Verdoorn)

```
> Verdoorn.Durbin = lagsarlm(gy5EX~gx5EX, listw=W5EX.lw ,type="mixed")
or
> Verdoorn.Durbin = lagsarlm(gy5EX~gx5EX+Lgx5EX, listw=W5EX.lw ,type="lag")
> summary(Verdoorn.Durbin)

Call:lagsarlm(formula = gy5EX ~ gx5EX, listw = W5EX.lw, type = "mixed")


Residuals:
      Min        1Q     Median        3Q       Max
-0.0253298 -0.0133893  0.0059794  0.0069515  0.0257882

Type: mixed
```

```
Coefficients: (asymptotic standard errors)
             Estimate  Std. Error  z value    Pr(>|z|)
(Intercept)  0.431002  0.043141    9.9906     < 2.2e-16
gx5EX        0.376790  0.012834   29.3584     < 2.2e-16
lag.gx5EX    0.459985  0.040516   11.3533     < 2.2e-16

Rho: -1.148, LR test value: 8.2174, p-value: 0.0041491
Asymptotic standard error: 0.1102
   z-value: -10.418, p-value: < 2.22e-16
Wald statistic: 108.54, p-value: < 2.22e-16

Log likelihood: 11.14436 for mixed model
ML residual variance (sigma squared): 0.000314, (sigma: 0.01772)
Number of observations: 5
Number of parameters estimated: 5
AIC: -12.289, (AIC for lm: -6.0713)
LM test for residual autocorrelation
test value: 1.4465, p-value: 0.22909
```

- **Computing general statistics and overall test**

Maximum likelihood (ML) estimator of the error variance:

```
> n = length(Verdoorn.Durbin$residuals)
> n
[1] 5
> se2.ML = sum(Verdoorn.Durbin$residuals^2)/n
or
> se2.ML = sum(Verdoorn.Durbin$s2
> se2.ML
[1] 0.0003139959
```

Unbiased estimator of the error variance:

```
> Verdoorn.Durbin.df.resid = n - (Verdoorn.Durbin$parameters - 1)
> Verdoorn.sar.df.resid
[1] 1
> se2 = sum(Verdoorn.Durbin$residuals^2)/Verdoorn.Durbin.df.resid
or
> se2 = Verdoorn.Durbin$SSE/Verdoorn.Durbin.df.resid
> se2
[1] 0.001569979
```

Standard error of regression (SER):

```
> SER = sqrt(se2)
> SER
[1] 0.039622964
```

Coefficient of determination (R^2):

```
# The R function var uses the denominator n-1
> R2 = 1- var(Verdoorn.Durbin$residuals)/var(Verdoorn.Durbin$y)
or
```

```
> R2 = 1- var(Verdoorn.Durbin$residuals)/var(gy5EX)
> R2
[1] 0.9965266
```

Adjusted coefficient of determination:

```
> R2adj = 1 -
(sum(Verdoorn.Durbin$residual^2)/Verdoorn.Durbin.df.resid)/var(Verdoorn.Durbin$y)
or
> R2adj = 1 -
(sum(Verdoorn.Durbin$residual^2)/Verdoorn.Durbin.df.resid)/var(gy5EX)
or
> R2adj = 1 - (1 - R2)*(n-1)/Verdoorn.Durbin.df.resid
> R2adj
[1] 0.9861064
```

- **Computing logLik, AIC and BIC**

```
> logLik.VerdDurbin = Verdoorn.Durbin$LL
> logLik.VerdDurbin
        [,1]
[1,] 11.14436
> AIC.VerdDurbin = AIC(Verdoorn.Durbin)
> AIC.VerdDurbin
[1] -12.28871
> BIC.VerdDurbin = BIC(Verdoorn.Durbin)
> BIC.VerdDurbin
[1] -14.24152
```

- **Impact measures in Spatial Durbin Model (Verdoorn's law)**

```
> Imp1Verdoorn.Durbin = impacts(Verdoorn.Durbin, listw=W5EX.lw)
> Imp1Verdoorn.Durbin
Impact measures (mixed, exact):
          Direct     Indirect    Total
gx5EX 0.3395785 0.04997282 0.3895514
> I5 = diag(5)
> I5
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    1    0    0    0
[3,]    0    0    1    0    0
[4,]    0    0    0    1    0
[5,]    0    0    0    0    1
```

```
> W5EX
          1         2         3         4         5
1 0.0000000 0.5000000 0.5000000 0.0000000 0.0000000
2 0.3333333 0.0000000 0.3333333 0.3333333 0.0000000
3 0.3333333 0.3333333 0.0000000 0.3333333 0.0000000
4 0.0000000 0.3333333 0.3333333 0.0000000 0.3333333
5 0.0000000 0.0000000 0.0000000 1.0000000 0.0000000
> Verdoorn.Durbin$rho
      rho
-1.148049
> I5_rhoDW5EX = I5 - Verdoorn.Durbin$rho*W5EX
> I5_rhoDW5EX
          1         2         3         4         5
1 1.0000000 0.5740244 0.5740244 0.0000000 0.0000000
2 0.3826829 1.0000000 0.3826829 0.3826829 0.0000000
3 0.3826829 0.3826829 1.0000000 0.3826829 0.0000000
4 0.0000000 0.3826829 0.3826829 1.0000000 0.3826829
5 0.0000000 0.0000000 0.0000000 1.1480487 1.0000000
> VW5D = solve(I5_rhoDW5EX)
> VW5D
           1          2          3         4          5
1  2.0437108 -1.3636757 -1.3636757  1.861571 -0.7123913
2 -0.9091171  1.9977769  0.3778639 -1.621508  0.6205236
3 -0.9091171  0.3778639  1.9977769 -1.621508  0.6205236
4  1.2410472 -1.6215084 -1.6215084  3.997149 -1.5296407
5 -1.4247826  1.8615707  1.8615707 -4.588922  2.7561021
> Verdoorn.Durbin$coeff[2]
     gx5EX
0.3767903
> I5EXDcoeff2 = Verdoorn.Durbin$coeff[2]*I5
> I5EXDcoeff2
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.3767903 0.0000000 0.0000000 0.0000000 0.0000000
[2,] 0.0000000 0.3767903 0.0000000 0.0000000 0.0000000
[3,] 0.0000000 0.0000000 0.3767903 0.0000000 0.0000000
[4,] 0.0000000 0.0000000 0.0000000 0.3767903 0.0000000
[5,] 0.0000000 0.0000000 0.0000000 0.0000000 0.3767903
> Verdoorn.Durbin$coeff[3]
 lag.gx5EX
0.4599851
> W5EXDcoeff3 = Verdoorn.Durbin$coeff[3]*W5EX
> W5EXDcoeff3
          1         2         3         4         5
1 0.0000000 0.2299925 0.2299925 0.0000000 0.0000000
2 0.1533284 0.0000000 0.1533284 0.1533284 0.0000000
3 0.1533284 0.1533284 0.0000000 0.1533284 0.0000000
4 0.0000000 0.1533284 0.1533284 0.0000000 0.1533284
5 0.0000000 0.0000000 0.0000000 0.4599851 0.0000000

> SW5EXD = VW5EX%*%(I5EXDcoeff2+W5EXDcoeff3)
```

```
> SW5EXD
          1          2          3          4           5
1 0.44018606 0.33113561 0.33113561 0.06518778 0.006240088
2 0.22075707 0.45417073 0.25023092 0.22699716 0.021729261
3 0.22075707 0.25023092 0.45417073 0.22699716 0.021729261
4 0.04345852 0.22699716 0.22699716 0.47740449 0.199027811
5 0.01248018 0.06518778 0.06518778 0.59708343 0.433945968
> dir5EXD = sum(diag(SW5EXD))/5
> dir5EXD
[1] 0.4519756
> tot5EXD = sum(SW5EXD)/5
> tot5EXD
[1] 1.173885
> ind5EXD = tot5EXD - dir5EXD
> ind5EXD
[1] 0.7219095
```

Exercises

4-2-1 Estimate the spatial Durbin model of the Beveridge curve by the method of maximum likelihood (ML) and interpret the regression coefficients (α=0.01)!

4-2-2 Determine impact measures for the spatial Durbin model of the Beveridge curve and interpret them (α=0.01)!

4-2-3 Compare the standard error of regression (SER) between the spatial error and the spatial Durbin model!

4-2-4 Compare the Akaike information criterion (AIC) between the standard regression and spatial Durbin model and interpret it!

4.5 Model comparison

Estimation of Verdoorn's law has brought out the following summary statistics with four differrent regression models:

Table 4.5.1: Global measures for alternative regression models of Verdoorn's laws

	Standard regression model	SLX model	Spatial lag model (SAR model)	Spatial error model (SEM model)	Spatial Durbin model
$s^2_{e,ML}$	0.0078	0.0035	0.0063	0.0003	0.0003
$s^2_{e,unbiased}$	0.0130	0.0088	0.0158	0.0008	0.0016
SER	0.1139	0.0937	0.1256	0.0276	0.0396
R^2	0.914	0.961	0.930	0.997	0.997
Adj. R^2	0.885	0.922	0.860	0.993	0.986
logLik	5.045	7.036	5.478	10.852	11.144
AIC	-4.090	-6.071	-2.957	-13.703	-12.289
BIC	-5.262	-7.634	-4.519	-15.265	-14.242

Exercises

4-5-1 Explain why some global measures may be misleading for model choice!

4-5-2 Discuss the model choice on the basis of relevant measures provided in table 4.5.1!

5. Panel Data Models
5.1 Panel data structure

• Reading data file VerdoornALQ.CSV into R as an object of type data.frame

```
> VerdoornALQ.df = read.csv(file="VerdoornALQ.CSV", header=TRUE, sep = ",",
dec = ".")
> class(VerdoornALQ.df)
[1] "data.frame"
> str(VerdoornALQ.df)
'data.frame':   15 obs. of  5 variables:
 $ region: int  1 1 1 2 2 2 3 3 3 4 ...
 $ year  : int  2014 2015 2016 2014 2015 2016 2014 2015 2016 2014 ...
 $ u     : num  8 7.5 7 6 5 5 6 6 5.5 3 ...
 $ gx    : num  0.6 0.8 0.85 1 1.1 1.2 1.6 1.6 1.5 2.6 ...
 $ gy    : num  0.4 0.5 0.6 0.6 0.8 0.8 0.9 0.95 0.9 1.1 ...
```

Panel data are generally ordered first by cross-section and then by time period. A cross section of observations (individuals, groups, countries, regions) is repeated over several time periods.

```
> VerdoornALQ.df
   region year   u   gx   gy
1       1 2014 8.0 0.60 0.40
2       1 2015 7.5 0.80 0.50
3       1 2016 7.0 0.85 0.60
4       2 2014 6.0 1.00 0.60
5       2 2015 5.0 1.10 0.80
6       2 2016 5.0 1.20 0.80
7       3 2014 6.0 1.60 0.90
8       3 2015 6.0 1.60 0.95
9       3 2016 5.5 1.50 0.90
10      4 2014 3.0 2.60 1.10
11      4 2015 2.8 2.55 1.10
12      4 2016 2.5 2.60 1.12
13      5 2014 2.0 2.20 1.20
14      5 2015 1.5 2.30 1.50
15      5 2016 1.5 2.40 1.60

> summary(VerdoornALQ.df)
     region       year            u               gx              gy       
 Min.   :1   Min.   :2014   Min.   :1.50   Min.   :0.60   Min.   :0.400  
 1st Qu.:2   1st Qu.:2014   1st Qu.:2.65   1st Qu.:1.05   1st Qu.:0.700  
 Median :3   Median :2015   Median :5.00   Median :1.60   Median :0.900  
 Mean   :3   Mean   :2015   Mean   :4.62   Mean   :1.66   Mean   :0.938  
 3rd Qu.:4   3rd Qu.:2016   3rd Qu.:6.00   3rd Qu.:2.35   3rd Qu.:1.110  
 Max.   :5   Max.   :2016   Max.   :8.00   Max.   :2.60   Max.   :1.600  
```

• Indexed data structure (internal ordering): first index “region”, second index “year”

A conversion of a data.frame object into a pdata.frame object is necessary for applying special functions to panel data. Particularly, time lags for variables of the panel structure can only be formed with the R function lag from a pdata.frame object. In order to use the facilities for pdata.frame objects, the package plm must be loaded:

```
> library(plm)

> VerdoornALQ.pdf = pdata.frame(VerdoornALQ.df, c("region","year"))

> class(VerdoornALQ.pdf)
[1] "pdata.frame" "data.frame"

> str(VerdoornALQ.pdf)
Classes ‘pdata.frame’ and 'data.frame': 15 obs. of  5 variables:
 $ region: Factor w/ 5 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 3 4 ...
  ..- attr(*, "index")=Classes ‘pindex’ and 'data.frame':       15 obs. of  2 variables:
  .. ..$ region: Factor w/ 5 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 3 4 ...
  .. ..$ year  : Factor w/ 3 levels "2014","2015",..: 1 2 3 1 2 3 1 2 3 1 ...
  ..- attr(*, "names")= chr  "1-2014" "1-2015" "1-2016" "2-2014" ...
 $ year  : Factor w/ 3 levels "2014","2015",..: 1 2 3 1 2 3 1 2 3 1 ...
  ..- attr(*, "index")=Classes ‘pindex’ and 'data.frame':       15 obs. of  2 variables:
  .. ..$ region: Factor w/ 5 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 3 4 ...
  .. ..$ year  : Factor w/ 3 levels "2014","2015",..: 1 2 3 1 2 3 1 2 3 1 ...
  ..- attr(*, "names")= chr  "1-2014" "1-2015" "1-2016" "2-2014" ...
 $ u     :Classes 'pseries', 'numeric'  atomic [1:15] 8 7.5 7 6 5 5 6 6 5.5 3 ...
  .. ..- attr(*, "index")=Classes ‘pindex’ and 'data.frame':    15 obs. of  2 variables:
  .. .. ..$ region: Factor w/ 5 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 3 4 ...
  .. .. ..$ year  : Factor w/ 3 levels "2014","2015",..: 1 2 3 1 2 3 1 2 3 1 ...
 $ gx    :Classes 'pseries', 'numeric'  atomic [1:15] 0.6 0.8 0.85 1 1.1 1.2 1.6 1.6 1.5
2.6 ...
  .. ..- attr(*, "index")=Classes ‘pindex’ and 'data.frame':    15 obs. of  2 variables:
  .. .. ..$ region: Factor w/ 5 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 3 4 ...
  .. .. ..$ year  : Factor w/ 3 levels "2014","2015",..: 1 2 3 1 2 3 1 2 3 1 ...
 $ gy    :Classes 'pseries', 'numeric'  atomic [1:15] 0.4 0.5 0.6 0.6 0.8 0.8 0.9 0.95 0.9
1.1 ...
  .. ..- attr(*, "index")=Classes ‘pindex’ and 'data.frame':    15 obs. of  2 variables:
  .. .. ..$ region: Factor w/ 5 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 3 4 ...
  .. .. ..$ year  : Factor w/ 3 levels "2014","2015",..: 1 2 3 1 2 3 1 2 3 1 ...
 - attr(*, "index")=Classes ‘pindex’ and 'data.frame':  15 obs. of  2 variables:
  ..$ region: Factor w/ 5 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 3 4 ...
  ..$ year  : Factor w/ 3 levels "2014","2015",..: 1 2 3 1 2 3 1 2 3 1 ...
```

```
> VerdoornALQ.pdf
       region year   u   gx   gy
1-2014      1 2014 8.0 0.60 0.40
1-2015      1 2015 7.5 0.80 0.50
1-2016      1 2016 7.0 0.85 0.60
2-2014      2 2014 6.0 1.00 0.60
2-2015      2 2015 5.0 1.10 0.80
2-2016      2 2016 5.0 1.20 0.80
3-2014      3 2014 6.0 1.60 0.90
3-2015      3 2015 6.0 1.60 0.95
3-2016      3 2016 5.5 1.50 0.90
4-2014      4 2014 3.0 2.60 1.10
4-2015      4 2015 2.8 2.55 1.10
4-2016      4 2016 2.5 2.60 1.12
5-2014      5 2014 2.0 2.20 1.20
5-2015      5 2015 1.5 2.30 1.50
5-2016      5 2016 1.5 2.40 1.60

> class(VerdoornALQ.pdf$u)
[1] "pseries" "numeric"

> str(VerdoornALQ.pdf$u)
Classes 'pseries', 'numeric'  atomic [1:15] 8 7.5 7 6 5 5 6 6 5.5 3 ...
  ..- attr(*, "index")=Classes ‘pindex’ and 'data.frame':       15 obs. of  2 variables:
  .. ..$ region: Factor w/ 5 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 3 4 ...
  .. ..$ year  : Factor w/ 3 levels "2014","2015",..: 1 2 3 1 2 3 1 2 3 1 ...

> VerdoornALQ.pdf$u
1-2014 1-2015 1-2016 2-2014 2-2015 2-2016 3-2014 3-2015 3-2016 4-2014 4-2015
   8.0    7.5    7.0    6.0    5.0    5.0    6.0    6.0    5.5    3.0    2.8
4-2016 5-2014 5-2015 5-2016
   2.5    2.0    1.5    1.5
```

Time lags of the unemployment rate (u)

```
> lag(VerdoornALQ.pdf$u, 1)
1-2014 1-2015 1-2016 2-2014 2-2015 2-2016 3-2014 3-2015 3-2016 4-2014 4-2015
    NA    8.0    7.5     NA    6.0    5.0     NA    6.0    6.0     NA    3.0
4-2016 5-2014 5-2015 5-2016
   2.8     NA    2.0    1.5
> lag(VerdoornALQ.pdf$u, 2)
1-2014 1-2015 1-2016 2-2014 2-2015 2-2016 3-2014 3-2015 3-2016 4-2014 4-2015
    NA     NA      8     NA     NA      6     NA     NA      6     NA     NA
4-2016 5-2014 5-2015 5-2016
     3     NA     NA      2
```

5.2 Pooling Model

The pooling model is given by

(6) $$y_{rt} = \beta_0 + \sum_{j=1}^{m} \beta_j \cdot x_{jrt} + \varepsilon_{rt},$$

where y is the dependent variable, x_j the *jth* explanatory variable, j=1,2,…,m, and ε the disturbance term. β_0 is the intercept and β_j are the regression coefficients of the explanatory variables x_j. r is the region index, r=1,2,…,n, and t the time index, t=1,2,..,T.

- **OLS estimation of pooling model (Verdoorn's law)**

```
> Verdoorn.pool = plm(gy~gx, data=VerdoornALQ.df, model="pooling")
or
> Verdoorn.pool = plm(gy~gx, data=VerdoornALQ.pdf, model="pooling")

> class(Verdoorn.pool)
[1] "plm"        "panelmodel"

> summary(Verdoorn.pool)
Oneway (individual) effect Pooling Model

Call:
plm(formula = gy ~ gx, data = VerdoornALQ.df, model = "pooling")
Balanced Panel: n=5, T=3, N=15

Residuals :
    Min.   1st Qu.    Median   3rd Qu.      Max.
-0.230000 -0.087900 -0.000458  0.045300  0.354000

Coefficients :
               Estimate   Std. Error  t-value   Pr(>|t|)
(Intercept) 0.246246   0.114982  2.1416  0.05174 .
gx             0.416719   0.063843  6.5273 1.92e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:    1.6592
Residual Sum of Squares: 0.38791
R-Squared:       0.76621
Adj. R-Squared: 0.74823
F-statistic: 42.6055 on 1 and 13 DF, p-value: 1.9202e-05
```

□ Note:
Apart from rounding differences the same results as with the R function lm are obtained.

```
> Verdoorn.lmpool = lm(gy~gx, data=VerdoornALQ.df)
> summary(Verdoorn.lmpool)
```

```
Call:
lm(formula = gy ~ gx, data = VerdoornALQ.df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.22972 -0.08795 -0.00046  0.04535  0.35363


Coefficients:
            Estimate  Std. Error  t value    Pr(>|t|)
(Intercept)  0.24625    0.11498    2.142    0.0517 .
gx           0.41672    0.06384    6.527   1.92e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1727 on 13 degrees of freedom
Multiple R-squared:  0.7662,    Adjusted R-squared:  0.7482
F-statistic: 42.61 on 1 and 13 DF,  p-value: 1.92e-05
```

- **Tests for individual and time effects**

Joint significance test (plmtest) for twoways effects based on the results of the pooling model

```
# default: type="honda"
> plmtest(Verdoorn.pool, effect="twoways", type="honda")

        Lagrange Multiplier Test - two-ways effects (Honda)

data:  gy ~ gx
normal = 1.4607, p-value = 0.07205
alternative hypothesis: significant effects
```

5.3 Fixed effects (FE) model

Fixed effects (FE) model with region fixed effects	Fixed effects (FE) model with region and time fixed effects
(7) $y_{rt} = \alpha_r + \sum_{j=1}^{m} \beta_j \cdot x_{jrt} + \varepsilon_{rt}$	(8) $y_{rt} = \alpha_r + \alpha_t + \sum_{j=1}^{m} \beta_j \cdot x_{jrt} + \varepsilon_{rt}$
α_r: region(individual) fixed effects	α_r: region(individual) fixed effects α_t: time fixed effects

A. Within estimation of individual fixed effects (FE) model (Verdoorn's law)

```
> Verdoorn.FEindiv = plm(gy~gx, data=VerdoornALQ.pdf, model="within",
effect="individual")
> summary(Verdoorn.FEindiv)
Oneway (individual) effect Within Model
Call:
plm(formula = gy ~ gx, data = VerdoornALQ.pdf, effect = "individual",
   model = "within")
Balanced Panel: n=5, T=3, N=15

Residuals :
   Min.  1st Qu.   Median  3rd Qu.     Max.
-0.12900 -0.03330 -0.00407  0.05480  0.06670

Coefficients :
     Estimate  Std. Error   t-value    Pr(>|t|)
gx  1.04400     0.24342   4.2889   0.002023 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:    0.13527
Residual Sum of Squares: 0.044439
R-Squared:      0.67147
Adj. R-Squared: 0.40288
F-statistic: 18.3951 on 1 and 9 DF, p-value: 0.0020232
```

□ <u>Note:</u>

The regression coefficients of explanatory variables are identical to those of least-squares dummy variables (LSDV) estimation of the standard regression model with regional dummies:

```
> Verdoorn.LSDVindiv = lm(gy~0+gx+factor(region), data=VerdoornALQ.df)
> summary(Verdoorn.LSDVindiv)
Call:
lm(formula = gy ~ 0 + gx + factor(region), data = VerdoornALQ.df)

Residuals:
     Min       1Q    Median       3Q       Max
-0.128933 -0.033333 -0.004067  0.054767  0.066667

Coefficients:
                  Estimate  Std. Error  t value   Pr(>|t|)
gx                 1.0440     0.2434    4.289   0.00202 **
factor(region)1  -0.2830     0.1870   -1.513   0.16451
factor(region)2  -0.4151     0.2708   -1.533   0.15972
factor(region)3  -0.7189     0.3835   -1.875   0.09360 .
factor(region)4  -1.5903     0.6301   -2.524   0.03256 *
factor(region)5  -0.9679     0.5613   -1.724   0.11875
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 0.07027 on 9 degrees of freedom
Multiple R-squared: 0.997,   Adjusted R-squared: 0.995
F-statistic: 500 on 6 and 9 DF, p-value: 7.787e-11

- Standard error of regression (SER) and coefficient of determination (Pseudo-R²) of lm function

Degrees of freedom: (Txn=3x5=15) – (k=1) – (n=5) = 9
> Verdoorn.LSDVindiv$df.resid
[1] 9
> s2.resVerdLSDVindiv = sum(Verdoorn.LSDVindiv$resid^2)/9
> s2.resVerdLSDVindiv
[1] 0.00493763
> s.resVerdLSDVindiv = sqrt(s2.resVerdLSDVindiv)
> s.resVerdLSDVindiv
[1] 0.07026827
> sum((Verdoorn.LSDVindiv$resid)^2)
[1] 0.04443867
> sum((VerdoornALQ.df$gy-mean(VerdoornALQ.df$gy))^2)
[1] 1.65924
> R2.VerdLSDVindiv = 1 -
sum((Verdoorn.LSDVindiv$resid)^2)/sum((VerdoornALQ.df$gy-
mean(VerdoornALQ.df$gy))^2)
> R2.VerdLSDVindiv
[1] 0.9732175
or
> R2.VerdLSDVindiv = 1-var(Verdoorn.LSDVindiv$resid)/var(VerdoornALQ.dfo$gy)
> R2.VerdLSDVindiv
[1] 0.9732175
or
> R2.VerdLSDVindiv = var(Verdoorn.LSDVindiv$fitted)/var(VerdoornALQ.dfo$gy)
> R2.VerdLSDVindiv
[1] 0.9732175

- **Standard error of regression (SER) and coefficient of determination (Pseudo-R²)**

Degrees of freedom: (Txn=3x5=15) – (k=1) – (n=5) = 9
> Verdoorn.FEindiv$df.resid
[1] 9
> s2.resVerdFEindiv = sum(Verdoorn.FEindiv$resid^2)/9
> s2.resVerdFEindiv
[1] 0.00493763
> s.resVerdFEindiv = sqrt(s2.resVerdFEindiv)
> s.resVerdFEindiv
[1] 0.07026827

```
> sum((Verdoorn.FEindiv$resid)^2)
[1] 0.04443867
> sum((VerdoornALQ.df$gy-mean(VerdoornALQ.df$gy))^2)
[1] 1.65924
> R2.VerdFEindiv = 1 - sum((Verdoorn.FEindiv$resid)^2)/sum((VerdoornALQ.df$gy-
mean(VerdoornALQ.df$gy))^2)
> R2.VerdFEindiv
[1] 0.9732175
```

or

```
> R2.VerdFEindiv = 1-var(Verdoorn.FEindiv$resid)/var(VerdoornALQ.dfo$gy)
> R2.VerdFEindiv
[1] 0.9732175
```

- Significance tests (t tests) for individual (regional) effects

```
> summary(fixef(Verdoorn.FEindiv, effect="individual"))
```

or

```
> summary(fixef(Verdoorn.FEindiv))

  Estimate   Std. Error t-value   Pr(>|t|)
1 -0.28300    0.18702 -1.5132  0.16451
2 -0.41507    0.27081 -1.5327  0.15972
3 -0.71893    0.38350 -1.8746  0.09360 .
4 -1.59033    0.63013 -2.5238  0.03256 *
5 -0.96787    0.56133 -1.7243  0.11875
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Joint significance test (pFtest) for individual effects

```
> pFtest(Verdoorn.FEindiv, Verdoorn.pool)

        F test for individual effects

data:  gy ~ gx
F = 17.391, df1 = 4, df2 = 9, p-value = 0.0002906
alternative hypothesis: significant effects
```

- Joint significance tests (plmtest) for individual effects

```
> plmtest(Verdoorn.pool, type="bp", effect="individual")

        Lagrange Multiplier Test - (Breusch-Pagan) for balanced panels

data:  gy ~ gx
chisq = 7.3788, df = 1, p-value = 0.0066
alternative hypothesis: significant effects
```

- Wooldridge's test for serial correlation in "short" FE panels

```
> pwartest(gy~gx, data=VerdoornALQ.pdf)

        Wooldridge's test for serial correlation in FE panels

data:  plm.model
chisq = 1.5257, p-value = 0.2168
alternative hypothesis: serial correlation
```

- Locally robust LM tests for spatial lag and error correlation in panel models

```
> slmtest(Verdoorn.FEindiv, listw=W5EX, test="lml")

        LM test for spatial lag dependence

data:  formula (within transformation)
LM = 0.22283, df = 1, p-value = 0.6369
alternative hypothesis: spatial lag dependence

> slmtest(Verdoorn.FEindiv, listw=W5EX, test="rlml")

        Locally robust LM test for spatial lag dependence sub spatial error

data:  formula (within transformation)
LM = 0.33231, df = 1, p-value = 0.5643
alternative hypothesis: spatial lag dependence

> slmtest(Verdoorn.FEindiv, listw=W5EX, test="lme")

        LM test for spatial error dependence

data:  formula (within transformation)
LM = 0.023931, df = 1, p-value = 0.8771
alternative hypothesis: spatial error dependence

> slmtest(Verdoorn.FEindiv, listw=W5EX, test="rlme")

        Locally robust LM test for spatial error dependence sub spatial lag

data:  formula (within transformation)
LM = 0.13341, df = 1, p-value = 0.7149
alternative hypothesis: spatial error dependence
```

- Pesaran local CD test for cross-sectional dependence in panels

Argument w of R function pcdtest: n x n matrix describing proximity between observations

```
> pcdtest(Verdoorn.FEindiv, test="cd", w=W5EX)

        Pesaran CD test for local cross-sectional dependence in panels

data:  gy ~ gx
z = 0.44596, p-value = 0.6556
alternative hypothesis: cross-sectional dependence
```

B. Within estimation of twoways fixed effects (FE) model (Verdoorn's law)

```
> Verdoorn.FE = plm(gy~gx, data=VerdoornALQ.pdf, model="within",
effect="twoways")

> class(Verdoorn.FE)
[1] "plm"        "panelmodel"

> summary(Verdoorn.FE)
Twoways effects Within Model

Call:
plm(formula = gy ~ gx, data = VerdoornALQ.df, effect = "twoways",
    model = "within")
Balanced Panel: n=5, T=3, N=15

Residuals :
    Min.  1st Qu.   Median  3rd Qu.     Max.
-0.10500 -0.02190 -0.00374  0.03780  0.06590

Coefficients :
   Estimate Std. Error t-value Pr(>|t|)
gx  0.75478    0.28870  2.6144  0.03469 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:    0.060347
Residual Sum of Squares: 0.030533
R-Squared:      0.49404
Adj. R-Squared: 0.23055
F-statistic: 6.8351 on 1 and 7 DF, p-value: 0.03469
```

The regression coefficients of explanatory variables are identical to those of least-squares dummy variables (LSDV) estimation of the standard regression model with regional and time dummies:

```
> Verdoorn.LSDV = lm(gy~0+gx+factor(region)+factor(year), data=VerdoornALQ.df)
```

- **Standard error of regression (SER) and coefficient of determination (Pseudo-R²)**

Degrees of freedom: (Txn=3x5=15) – (k=1) – (n=5) – (T=3) + 1 = 7
To avoid exact multicollinearity, not all n+T=5+3=8 region and time effects can be estimated. One region or period must be used as a reference which explains the term "+1" in the computation of the degrees of freedom.

```
> Verdoorn.FE$df.resid
[1] 7

> s2.resVerdFE = sum(Verdoorn.FE$resid^2)/7
> s2.resVerdFE
[1] 0.004361853
> s.resVerdFE = sqrt(s2.resVerdFE)
> s.resVerdFE
[1] 0.06604433
> R2.VerdFE = 1-var(Verdoorn.FE$resid)/var(VerdoornALQ.dfo$gy)
> R2.VerdFE
[1] 0.9815982
```

- Significance tests (t tests) for individual (regional) effects

```
> summary(fixef(Verdoorn.FE, effect="individual"))
or
> summary(fixef(Verdoorn.FE))
   Estimate    Std. Error t-value    Pr(>|t|)
1 -0.066083   0.219857 -0.3006   0.7725
2 -0.096921   0.319851 -0.3030   0.7707
3 -0.265817   0.453901 -0.5856   0.5765
4 -0.843174   0.746782 -1.1291   0.2961
5 -0.302654   0.665103 -0.4550   0.6629
```

- Significance tests (t tests) for time effects

```
> summary(fixef(Verdoorn.FE, effect="time"))
        Estimate  Std. Error t-value   Pr(>|t|)
2014 -0.36764    0.46286 -0.7943   0.4531
2015 -0.29048    0.48303 -0.6014   0.5666
2016 -0.28667    0.49456 -0.5796   0.5803
```

- Joint significance test (pFtest) for twoways effects

```
> pFtest(Verdoorn.FE, Verdoorn.pool)

        F test for twoways effects

data:  gy ~ gx
F = 13.656, df1 = 6, df2 = 7, p-value = 0.001492
alternative hypothesis: significant effects
```

- Joint significance tests (plmtest) for individual effects

```
> plmtest(Verdoorn.pool, type="bp", effect="twoways")

        Lagrange Multiplier Test - two-ways effects (Breusch-Pagan) for
        balanced panels

data:  gy ~ gx
chisq = 7.8022, df = 2, p-value = 0.02022
alternative hypothesis: significant effects
```

- Locally robust LM tests for spatial lag and error correlation in panel models

```
> slmtest(Verdoorn.FE, listw=W5EX, test="lml")

        LM test for spatial lag dependence

data:  formula (within transformation)
LM = 3.3558, df = 1, p-value = 0.06697
alternative hypothesis: spatial lag dependence

> slmtest(Verdoorn.FE, listw=W5EX, test="rlml")

        Locally robust LM test for spatial lag dependence sub spatial error

data:  formula (within transformation)
LM = 4.1299, df = 1, p-value = 0.04213
alternative hypothesis: spatial lag dependence

> slmtest(Verdoorn.FE, listw=W5EX, test="lme")

        LM test for spatial error dependence

data:  formula (within transformation)
LM = 1.2279, df = 1, p-value = 0.2678
alternative hypothesis: spatial error dependence

> slmtest(Verdoorn.FE, listw=W5EX, test="rlme")

        Locally robust LM test for spatial error dependence sub spatial lag

data:  formula (within transformation)
LM = 2.002, df = 1, p-value = 0.1571
alternative hypothesis: spatial error dependence
```

- Pesaran local CD test for cross-sectional dependence in panels

Argument w of R function pcdtest: n x n matrix describing proximity between observations

```
> pcdtest(Verdoorn.FE, test="cd", w=WS5EX)

      Pesaran CD test for local cross-sectional dependence in panels

data:  gy ~ gx
z = -0.58236, p-value = 0.5603
alternative hypothesis: cross-sectional dependence
```

5.4 Random effects (RE) model

The random effects (RE) model with individual and idiocratic random errors are given by

(9) $y_{rt} = \sum_{j=1}^{m} \beta_j \cdot x_{jrt} + \upsilon_{rt}$,

$\upsilon_{rt} = \alpha_r + \varepsilon_{rt}$.

The disturbance term υ_{rt} is composed of the individual random error α_r and the idiocratic random error ε_{rt}.

- **Least-squares estimation of random effects (RE) model (Verdoorn's law)**

```
> Verdoorn.REindiv = plm(gy~gx, data=VerdoornALQ.pdf, model="random",
effect="individual")
> summary(Verdoorn.REindiv)
Oneway (individual) effect Random Effect Model
   (Swamy-Arora's transformation)

Call:
plm(formula = gy ~ gx, data = VerdoornALQ.pdf, effect = "individual",
    model = "random")
Balanced Panel: n=5, T=3, N=15

Effects:
                   var    std.dev   share
idiosyncratic 0.004938  0.070268  0.131
individual     0.032833  0.181198  0.869
theta:  0.7815

Residuals :
    Min.     1st Qu.    Median   3rd Qu.     Max.
-0.14600 -0.05960 -0.00985  0.04540  0.14700
```

```
Coefficients :
              Estimate   Std. Error  t-value   Pr(>|t|)
(Intercept) 0.053556   0.227854  0.2350 0.8178340
gx          0.532797   0.124407  4.2827 0.0008914 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:    0.20802
Residual Sum of Squares: 0.086282
R-Squared:      0.58522
Adj. R-Squared: 0.55331
F-statistic: 18.3416 on 1 and 13 DF, p-value: 0.0008914
```

- Hausman test (FE vs. RE Model)

```
> phtest(Verdoorn.FEindiv, Verdoorn.REindiv)

        Hausman Test

data:  gy ~ gx
chisq = 5.9699, df = 1, p-value = 0.01455
alternative hypothesis: one model is inconsistent
```

6. Spatial Panel Data Models
6.1 Spatial Panel Data Structure

• Reordering of data.frame object (first time, second region)

While panel data are generally ordered first by cross-section and then by time period, operations with a spatial weights matrix demand a reverse internal ordering. Spatial panel data are stacked first by time period and then by region. A time period (year, quarter, month) is repeated over all spatial units (regions).

```
> VerdoornALQ.dfo = VerdoornALQ.df[order(VerdoornALQ.df$year) , ]

> VerdoornALQ.dfo
   region year  u    gx    gy
1       1  2014 8.0 0.60 0.40
4       2  2014 6.0 1.00 0.60
7       3  2014 6.0 1.60 0.90
10      4  2014 3.0 2.60 1.10
13      5  2014 2.0 2.20 1.20
2       1  2015 7.5 0.80 0.50
5       2  2015 5.0 1.10 0.80
8       3  2015 6.0 1.60 0.95
11      4  2015 2.8 2.55 1.10
14      5  2015 1.5 2.30 1.50
3       1  2016 7.0 0.85 0.60
6       2  2016 5.0 1.20 0.80
9       3  2016 5.5 1.50 0.90
12      4  2016 2.5 2.60 1.12
15      5  2016 1.5 2.40 1.60
```

• Indexed data structure (internal ordering): first index “year”, second index “region”

A reordering of the pdata.frame object VerdoornALQ.pdf is necessary for applying special functions to spatial panel data. Particularly, spatial lags for variables of the panel structure can only be formed with the R function slag from a reordered pdata.frame object.

In order to use facilities for analyzing spatial panel data, the R package splm must be loaded:

```
> library(splm)
```

Data must be stacked first by time period and then by region.

```
> VerdoornALQ.pdfo = VerdoornALQ.pdf[order(VerdoornALQ.pdf$year) , ]

> class(VerdoornALQ.pdfo)
[1] "pdata.frame" "data.frame"
```

```
> VerdoornALQ.pdfo
       region year   u   gx   gy
1-2014      1 2014 8.0 0.60 0.40
2-2014      2 2014 6.0 1.00 0.60
3-2014      3 2014 6.0 1.60 0.90
4-2014      4 2014 3.0 2.60 1.10
5-2014      5 2014 2.0 2.20 1.20
1-2015      1 2015 7.5 0.80 0.50
2-2015      2 2015 5.0 1.10 0.80
3-2015      3 2015 6.0 1.60 0.95
4-2015      4 2015 2.8 2.55 1.10
5-2015      5 2015 1.5 2.30 1.50
1-2016      1 2016 7.0 0.85 0.60
2-2016      2 2016 5.0 1.20 0.80
3-2016      3 2016 5.5 1.50 0.90
4-2016      4 2016 2.5 2.60 1.12
5-2016      5 2016 1.5 2.40 1.60

> class(VerdoornALQ.pdfo$u)
[1] "pseries" "numeric"

> str(VerdoornALQ.pdfo$u)
Classes 'pseries', 'numeric'  atomic [1:15] 8 6 6 3 2 7.5 5 6 2.8 1.5 ...
  ..- attr(*, "index")='data.frame':    15 obs. of  2 variables:
  .. ..$ region: Factor w/ 5 levels "1","2","3","4",..: 1 2 3 4 5 1 2 3 4 5 ...
  .. ..$ year  : Factor w/ 3 levels "2014","2015",..: 1 1 1 1 1 2 2 2 2 2 ...

> VerdoornALQ.pdfo$u
1-2014 2-2014 3-2014 4-2014 5-2014 1-2015 2-2015 3-2015 4-2015 5-2015 1-2016
   8.0    6.0    6.0    3.0    2.0    7.5    5.0    6.0    2.8    1.5    7.0
2-2016 3-2016 4-2016 5-2016
   5.0    5.5    2.5    1.5
```

Spatial lag (first-order contiguity) of the unemployment rate (u):

```
> slag(VerdoornALQ.pdfo$u, W5EX.lw)
or
> slag(VerdoornALQ.pdfo$u, W5EX.lw, 1)
 [1] 6.000000 5.666667 5.666667 4.666667 3.000000 5.500000 5.433333 5.100000
 [9] 4.166667 2.800000 5.250000 5.000000 4.833333 4.000000 2.500000
```

6.2 Spatial panel fixed effects (FE) SLX model

Spatial cross-regressive (SLX) model with region fixed effects	Spatial cross-regressive (SLX) model with region and time fixed effects
(10) $y_{rt} = \alpha_r + \sum_{j=1}^{m} \beta_j \cdot x_{jrt} + \sum_{j=1}^{m} \theta_j \cdot SL(x_{jrt}) + \varepsilon_{rt}$	(11) $y_{rt} = \alpha_r + \alpha_t + \sum_{j=1}^{m} \beta_j \cdot x_{jrt} + \sum_{j=1}^{m} \theta_j \cdot SL(x_{jrt}) + \varepsilon_{rt}$
$SL(x_{jrt})$: spatial lag of the explanatory variable x_j of region r and period t θ_j: regression coefficient of the explanatory variable x_j	

Within estimation of spatial panel FE SLX models can be accomplished for variables from data.frame objects (VerdoornALQ.df, VerdoornALQ,dfo) or pdata.frame objects (VerdoornALQ.pdf, VerdoornALQ,pdfo).

To that end, the spatial lag of explanatory variable output growth (gx),

```
> VerdoornALQ.pdfo$gx
1-2014 2-2014 3-2014 4-2014 5-2014 1-2015 2-2015 3-2015 4-2015 5-2015 1-2016
  0.60   1.00   1.60   2.60   2.20   0.80   1.10   1.60   2.55   2.30   0.85
2-2016 3-2016 4-2016 5-2016
  1.20   1.50   2.60   2.40,
```

must be formed:

```
> VerdoornALQ.pdfo$SLgx = slag(VerdoornALQ.pdfo$gx, W5EX)
or
> VerdoornALQ.pdfo$SLgx = slag(VerdoornALQ.pdfo$gx, W5EX, 1)

> class(VerdoornALQ.pdfo$SLgx)
[1] "pseries" "numeric"

> VerdoornALQ.pdfo$SLgx
  1-2014   2-2014   3-2014   4-2014   5-2014   1-2015   2-2015   3-2015
1.300000 1.600000 1.400000 1.600000 2.600000 1.350000 1.650000 1.483333
  4-2015   5-2015   1-2016   2-2016   3-2016   4-2016   5-2016
1.666667 2.550000 1.350000 1.650000 1.550000 1.700000 2.600000

> class(VerdoornALQ.pdfo)
[1] "pdata.frame" "data.frame"

> VerdoornALQ.pdfo
       region year   u   gx   gy     SLgx
1-2014      1 2014 8.0 0.60 0.40 1.300000
2-2014      2 2014 6.0 1.00 0.60 1.600000
3-2014      3 2014 6.0 1.60 0.90 1.400000
```

```
4-2014      4 2014 3.0 2.60 1.10 1.600000
5-2014      5 2014 2.0 2.20 1.20 2.600000
1-2015      1 2015 7.5 0.80 0.50 1.350000
2-2015      2 2015 5.0 1.10 0.80 1.650000
3-2015      3 2015 6.0 1.60 0.95 1.483333
4-2015      4 2015 2.8 2.55 1.10 1.666667
5-2015      5 2015 1.5 2.30 1.50 2.550000
1-2016      1 2016 7.0 0.85 0.60 1.350000
2-2016      2 2016 5.0 1.20 0.80 1.650000
3-2016      3 2016 5.5 1.50 0.90 1.550000
4-2016      4 2016 2.5 2.60 1.12 1.700000
5-2016      5 2016 1.5 2.40 1.60 2.600000

> VerdoornALQ.dfo = as.data.frame(VerdoornALQ.pdfo)
> class(VerdoornALQ.dfo)
[1] "data.frame"
> VerdoornALQ.dfo
   region year  u    gx    gy      SLgx
1       1 2014 8.0 0.60 0.40 1.300000
2       2 2014 6.0 1.00 0.60 1.600000
3       3 2014 6.0 1.60 0.90 1.400000
4       4 2014 3.0 2.60 1.10 1.600000
5       5 2014 2.0 2.20 1.20 2.600000
6       1 2015 7.5 0.80 0.50 1.350000
7       2 2015 5.0 1.10 0.80 1.650000
8       3 2015 6.0 1.60 0.95 1.483333
9       4 2015 2.8 2.55 1.10 1.666667
10      5 2015 1.5 2.30 1.50 2.550000
11      1 2016 7.0 0.85 0.60 1.350000
12      2 2016 5.0 1.20 0.80 1.650000
13      3 2016 5.5 1.50 0.90 1.550000
14      4 2016 2.5 2.60 1.12 1.700000
15      5 2016 1.5 2.40 1.60 2.600000
```

A. SLX model with region (individual) fixed effect

```
> Verdoorn.FEindivSLX = plm(gy~gx+SLgx, data=VerdoornALQ.dfo, model="within",
effect="individual")

> summary(Verdoorn.FEindivSLX)
Oneway (individual) effect Within Model

Call:
plm(formula = gy ~ gx + SLgx, data = VerdoornALQ.dfo, effect = "individual",
    model = "within")
Balanced Panel: n=5, T=3, N=15

Residuals :
     Min.      1st Qu.    Median    3rd Qu.      Max.
-0.13400 -0.02860 -0.00677  0.04820  0.07380
```

```
Coefficients :
        Estimate  Std. Erro r t-value    Pr(>|t|)
gx      1.03179    0.25691  4.0161  0.003862 **
SLgx  0.21542    0.50601  0.4257  0.681526
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total Sum of Squares:    0.13527
Residual Sum of Squares: 0.043454
R-Squared:      0.67875
Adj. R-Squared: 0.43782
F-statistic: 8.45143 on 2 and 8 DF, p-value: 0.01065
```

- **Standard error of regression (SER) and coefficient of determination (Pseudo-R²)**

Degrees of freedom: (Txn=3x5=15) – (k=2) – (n=5) = 8

```
> Verdoorn.FEindivSLX$df.resid
[1] 8
> s2.resVerdFEindivSLX = sum(Verdoorn.FEindivSLX$resid^2)/8
> s2.resVerdFEindivSLX
[1] 0.005431772
> s.resVerdFEindivSLX = sqrt(s2.resVerdFEindivSLX)
> s.resVerdFEindivSLX
[1] 0.07370056
> R2.VerdFEindivSLX = 1-
var(Verdoorn.FEindivSLX$resid)/var(VerdoornALQ.dfo$gy)
> R2.VerdFEindivSLX
[1] 0.9738108
```

- **Extracting and testing fixed effects: fixedef.plm**

```
> summary(fixef(Verdoorn.FEindivSLX))
or
> summary(fixef(Verdoorn.FEindivSLX, effect= "individual"))
  Estimate  Std. Error t-value  Pr(>|t|)
1 -0.56108   0.68199 -0.8227  0.43451
2 -0.75350   0.84417 -0.8926  0.39813
3 -1.01816   0.80981 -1.2573  0.24411
4 -1.91545   1.00994 -1.8966  0.09446 .
5 -1.49630   1.37380 -1.0892  0.30780
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Locally robust LM tests for spatial lag and error correlation in panel models

```
> slmtest(Verdoorn.FEindivSLX, listw=W5EX, test="lml")

        LM test for spatial lag dependence

data:  formula (within transformation)
LM = 0.0041427, df = 1, p-value = 0.9487
alternative hypothesis: spatial lag dependence

> slmtest(Verdoorn.FEindivSLX, listw=W5EX, test="rlml")

        Locally robust LM test for spatial lag dependence sub spatial error

data:  formula (within transformation)
LM = 10.942, df = 1, p-value = 0.0009402
alternative hypothesis: spatial lag dependence

> slmtest(Verdoorn.FEindivSLX, listw=W5EX, test="lme")

        LM test for spatial error dependence

data:  formula (within transformation)
LM = 0.013248, df = 1, p-value = 0.9084
alternative hypothesis: spatial error dependence

> slmtest(Verdoorn.FEindivSLX, listw=W5EX, test="rlme")

        Locally robust LM test for spatial error dependence sub spatial lag

data:  formula (within transformation)
LM = 10.951, df = 1, p-value = 0.0009356
alternative hypothesis: spatial error dependence
```

- Pesaran local CD test for cross-sectional dependence in panels

Argument w of R function pcdtest: n x n matrix describing proximity between observations

```
> pcdtest(Verdoorn.FEindivSLX, test="cd", w=W5EX)

        Pesaran CD test for local cross-sectional dependence in panels

data:  gy ~ gx + SLgx
z = -0.0246, p-value = 0.9804
alternative hypothesis: cross-sectional dependence
```

B. SLX with region and time (twoways) fixed effect

```
> Verdoorn.FESLX = plm(gy~gx+SLgx, data=VerdoornALQ.dfo, model="within",
effect="twoways")

> summary(Verdoorn.FESLX)
Twoways effects Within Model
Call:
plm(formula = gy ~ gx + SLgx, data = VerdoornALQ.dfo, effect = "twoways",
    model = "within")
Balanced Panel: n=5, T=3, N=15

Residuals :
   Min.    1st Qu.  Median 3rd Qu.    Max.
-0.0416 -0.0219 -0.0139  0.0213   0.0609

Coefficients :
         Estimate  Std. Error  t-value    Pr(>|t|)
gx       0.25270    0.29444   0.8583   0.4237
SLgx -1.77802    0.70423  -2.5248   0.0450 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Total Sum of Squares:    0.060347
Residual Sum of Squares: 0.014804
R-Squared:      0.75468
Adj. R-Squared: 0.42758
F-statistic: 9.22875 on 2 and 6 DF, p-value: 0.014764
```

- **Standard error of regression and coefficient of determination (Pseudo-R^2)**

Degrees of freedom: (Txn=3x5=15) – (k=2) – (n=5) – (T=3) + 1 = 6
To avoid exact multicollinearity, not all n+T=5+3=8 region and time effects can be estimated. One region or period must be used as a reference which explains the term “+1” in the computation of the degrees of freedom.

```
> Verdoorn.FESLX$df.resid
[1] 6
> s2.resVerdFESLX = sum(Verdoorn.FESLX$resid^2)/6
> s2.resVerdFESLX
[1] 0.002467409
> s.resVerdFESLX = sqrt(s2.resVerdFESLX)
> s.resVerdFESLX
[1] 0.04967302
> R2.VerdFESLX = 1-var(Verdoorn.FESLX$resid)/var(VerdoornALQ.dfo$gy)
> R2.VerdFESLX
[1] 0.9910776
```

- **Extracting and testing fixed effects: fixedef.plm**

```
> summary(fixef(Verdoorn.FESLX))
or
> summary(fixef(Verdoorn.FESLX, effect= "individual"))
  Estimate Std. Error t-value   Pr(>|t|)
1   2.6812     1.1006  2.4361  0.05074 .
2   3.3595     1.3900  2.4169  0.05208 .
3   3.1483     1.3947  2.2574  0.06478 .
4   3.3975     1.7710  1.9184  0.10350
5   5.4453     2.3309  2.3361  0.05815 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> summary(fixef(Verdoorn.FESLX, effect="time"))
       Estimate Std. Error t-value   Pr(>|t|)
2014   3.4583     1.5548  2.2242  0.06780 .
2015   3.6417     1.5993  2.2771  0.06305 .
2016   3.7190     1.6296  2.2822  0.06261 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Locally robust LM tests for spatial lag and error correlation in panel models

```
> slmtest(Verdoorn.FESLX, listw=W5EX, test="lml")

        LM test for spatial lag dependence

data:  formula (within transformation)
LM = 2.8126, df = 1, p-value = 0.09352
alternative hypothesis: spatial lag dependence

> slmtest(Verdoorn.FESLX, listw=W5EX, test="rlml")

        Locally robust LM test for spatial lag dependence sub spatial error

data:  formula (within transformation)
LM = 0.39987, df = 1, p-value = 0.5272
alternative hypothesis: spatial lag dependence

> slmtest(Verdoorn.FESLX, listw=W5EX, test="lme")

        LM test for spatial error dependence

data:  formula (within transformation)
LM = 2.5206, df = 1, p-value = 0.1124
alternative hypothesis: spatial error dependence
```

> slmtest(Verdoorn.FESLX, listw=W5EX, test="rlme")

Locally robust LM test for spatial error dependence sub spatial lag

data: formula (within transformation)
LM = 0.10787, df = 1, p-value = 0.7426
alternative hypothesis: spatial error dependence

- Pesaran local CD test for cross-sectional dependence in panels

Argument w of R function pcdtest: n x n matrix describing proximity between observations

> pcdtest(Verdoorn.FESLX, test="cd", w=W5EX)

Pesaran CD test for local cross-sectional dependence in panels

data: gy ~ gx + SLgx
z = -2.2543, p-value = 0.02418
alternative hypothesis: cross-sectional dependence

6.3 Spatial panel fixed effect (FE) lag model

Spatial panel fixed effect (FE) lag model with region fixed effects	Spatial panel fixed effect (FE) lag model with region and time fixed effects
(12) $y_{rt} = \alpha_r + \lambda \cdot SL(y_{rt}) + \sum_{j=1}^{m} \beta_j \cdot x_{jrt} + \sum_{j=1}^{m} \theta_j \cdot SL(x_{jrt}) + \varepsilon_{rt}$	(13) $y_{rt} = \alpha_r + \alpha_t + \lambda \cdot SL(y_{rt}) + \sum_{j=1}^{m} \beta_j \cdot x_{jrt} + \sum_{j=1}^{m} \theta_j \cdot SL(x_{jrt}) + \varepsilon_{rt}$
$SL(y_{rt})$: spatial lag of the dependent variable y of region r and period t λ: regression coefficient of the explanatory variable x_j	

Maximum likelihood (ML) estimation of spatial panel FE lag models can be accomplished for variables from data.frame objects (VerdoornALQ.df, VerdoornALQ,dfo) or pdata.frame objects (VerdoornALQ.pdf, VerdoornALQ,pdfo).

A. Spatial lag model with region (individual) fixed effect

> Verdoorn.FEindivslag = spml(gy~gx, data=VerdoornALQ.pdfo, listw=W5EX.lw, model="within", effect="individual", lag=TRUE, spatial.error="none")

> summary(Verdoorn.FEindivslag)
Spatial panel fixed effects lag model

```
Call:
spml(formula = gy ~ gx, data = VerdoornALQ.pdfo, listw = W5EX.lw,
   model = "within", effect = "individual", lag = TRUE, spatial.error = "none")

Residuals:
    Min.    1st Qu.    Median    3rd Qu.     Max.
-0.13000 -0.03290 -0.00491  0.05330  0.06740

Spatial autoregressive coefficient:
          Estimate  Std. Error t-value   Pr(>|t|)
lambda  0.10489    0.20771   0.505   0.6136

Coefficients:
    Estimate  Std. Error t-value    Pr(>|t|)
gx  1.02285    0.18786  5.4448 5.186e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- **Standard error of regression (SER) and coefficient of determination (Pseudo-R^2)**

Degrees of freedom: (Txn=3x5=15) – (k+1=2) – (n=5) = 8

When the spatial panel FE lag model is estimated, R reports the `intercept' with the function effects.splm. However, fixed effect does not have the intercept. What R is reporting is not an estimated parameter but the average value of the fixed effed coefficients.

```
> s2.resVerdFEindivslag = sum(Verdoorn.FEindivslag$resid^2)/8
> s2.resVerdFEindivslag
[1] 0.005445342
> s.resVerdFEindivslag = sqrt(s2.resVerdFEindivslag)
> s.resVerdFEindivslag
[1] 0.07379256
> R2.VerdFEindivslag = 1-var(Verdoorn.FEindivslag$resid)/var(VerdoornALQ.dfo$gy)
> R2.VerdFEindivslag
[1] 0.9737454
or
> R2.VerdFEindivslag = summary(Verdoorn.FEindivslag)$rsqr
> R2.VerdFEindivslag
[1] 0.9737454
```

- **Extracting and testing fixed effects: effects.splm**

```
> effects.splm(Verdoorn.FEindivslag)
or
> effects(Verdoorn.FEindivslag)

Intercept:
           Estimate Std. Error t-value Pr(>|t|)
(Intercept) -0.85602    0.31020 -2.7595 0.005789 **
```

```
Spatial fixed effects:
   Estimate Std. Error t-value Pr(>|t|)
1  0.502348   0.143427  3.5025 0.000461 ***
2  0.375995   0.207694  1.8103 0.070245 .
3  0.088409   0.294119  0.3006 0.763728
4 -0.787475   0.483265 -1.6295 0.103209
5 -0.179276   0.430495 -0.4164 0.677086
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- **Computing logLik, AIC and BIC**

The log likelihood (logLik) is not available from splm object after running the R function spml:

```
> class(Verdoorn.FEindivslag)
[1] "splm"
> Verdoorn.FEindivslag$logLik
[1] NULL
```

The value of logLik is computed with the spml command by specifying the argument quite = FALSE:

```
> Verdoorn.FEindivslag = spml(gy~gx, data=VerdoornALQ.pdfo, listw=W5EX.lw,
model="within", effect="individual", lag=TRUE, spatial.error="none", quiet=FALSE)

 Spatial Lag Fixed Effects Model

Spatial fixed effects model
 Jacobian calculated using neighbourhood matrix eigenvalues
Computing eigenvalues ...

lambda:  -0.4231936     function value:  20.86957
lambda:  0.120418       function value:  23.46528
lambda:  0.4563884     function value:  22.09641
lambda:  0.0859158     function value:  23.46403
lambda:  0.1048048     function value:  23.46787
lambda:  0.1048656     function value:  23.46787
lambda:  0.1048891     function value:  23.46787
lambda:  0.1048892     function value:  23.46787
lambda:  0.1048892     function value:  23.46787
lambda:  0.1048892     function value:  23.46787
lambda:  0.1048892     function value:  23.46787
```

After assigning the last function value of 23.46787 to the slot logLik,

```
> Verdoorn.FEindivslag$logLik = 23.46787
> Verdoorn.FEindivslag$logLik
[1] 23.46787,
```

information criteria like AIC and BIC can be computed.

For spatial panel data no R functions of AIC and BIC are available. For computing AIC and BIC for a splm object, a user-defined function has to be used:

User-defined function AICsplm

```
# AIC and BIC function for splm object #

AICsplm = function(object, k=2, criterion=c("AIC", "BIC")){
  sp = summary(object)
  l = sp$logLik
  np = length(coef(sp))
  N = nrow(sp$model)
  if (sp$effects=="sptpfe") {
    n = length(sp$res.eff[[1]]$res.sfe)
    T = length(sp$res.eff[[1]]$res.tfe)
    np = np+n+T
  }
  if (sp$effects=="spfe") {
      n = length(sp$res.eff[[1]]$res.sfe)
      np = np+n+1
  }
  if (sp$effects=="tpfe") {
        T = length(sp$res.eff[[1]]$res.tfe)
        np = np+T+1
  }
  if(criterion=="AIC"){
    aic = -2*l+k*np
    names(aic) = "AIC"
    return(aic)
  }
  if(criterion=="BIC"){
    bic = -2*l+log(N)*np
    names(bic) = "BIC"
    return(bic)
  }
}
```

Loading user-defined function AICsplm

```
> source("AICsplm.R")
```

and computing AIC and BIC:

```
>  AICsplm (Verdoorn.FEindivslag, criterion="AIC")
      AIC
-30.93574
> AICsplm (Verdoorn.FEindivslag, criterion="BIC")
      BIC
-25.27134
```

B. Spatial lag model with region and time (twoways) fixed effect

```
> Verdoorn.FEslag = spml(gy~gx, data=VerdoornALQ.pdfo, listw=W5EX.lw,
model="within", effect="twoways", lag=TRUE, spatial.error="none")

> summary(Verdoorn.FEslag)
Spatial panel fixed effects lag model
```

```
Call:
spml(formula = gy ~ gx, data = VerdoornALQ.pdfo, listw = W5EX.lw,
   model = "within", effect = "twoways", lag = TRUE, spatial.error = "none")

Residuals:
   Min.      1st Qu.   Median    3rd Qu.     Max.
-0.07750 -0.01520  0.00283  0.02000  0.05530

Spatial autoregressive coefficient:
          Estimate  Std. Error t-value   Pr(>|t|)
lambda -0.57083    0.22598 -2.5261  0.01153 *

Coefficients:
    Estimate  Std. Error t-value   Pr(>|t|)
gx  0.59722    0.16459  3.6285 0.000285 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- **Standard error of regression (SER) and coefficient of determination (Pseudo-R^2)**

Degrees of freedom: (Txn=3x5=15) – (k+1=2) – (n=5) – (T=3) + 1 = 6
To avoid exact multicollinearity, not all n+T=5+3=8 region and time effects can be estimated. One region or period must be used as a reference which explains the term "+1" in the computation of the degrees of freedom.

```
> s2.resVerdFEslag = sum(Verdoorn.FEslag$resid^2)/6
> s2.resVerdFEslag
[1] 0.002758696

> s.resVerdFEslag = sqrt(s2.resVerdFEslag)
> s.resVerdFEslag
[1] 0.05252329

> R2.VerdFEslag = 1-var(Verdoorn.FEslag$resid)/var(VerdoornALQ.dfo$gy)
> R2.VerdFEslag
[1] 0.9900242
or
> R2.VerdFEslag = summary(Verdoorn.FEslag)$rsqr
> R2.VerdFEslag
[1] 0.9900242
```

- **Extracting and testing fixed effects: effects.splm**

```
> effects.splm(Verdoorn.FEslag)
or
> effects(Verdoorn.FEslag)
```

```
Intercept:
             Estimate  Std. Error  t-value  Pr(>|t|)
(Intercept)  0.46956    0.26066  1.8014  0.07164 .

Spatial fixed effects:
   Estimate   Std. Error   t-value   Pr(>|t|)
1  0.053463   0.119506  0.4474   0.6546
2  0.086965   0.173859  0.5002   0.6169
3 -0.043290   0.246724 -0.1755   0.8607
4 -0.319025   0.405924 -0.7859   0.4319
5  0.221887   0.361527  0.6138   0.5394

Time period fixed effects:
   Estimate    Std. Error  t-value   Pr(>|t|)
1 -0.099905   0.251596 -0.3971   0.6913
2  0.040616   0.262559  0.1547   0.8771
3  0.059289   0.268825  0.2206   0.8254
```

- **Computing logLik, AIC and BIC**

The log likelihood (logLik) is not available from splm object after running the R function spml:

```
> class(Verdoorn.FEslag)
[1] "splm"

> Verdoorn.FEslag$logLik
[1] NULL
```

The value of logLik is computed with the spml command by specifying the argument quite = FALSE:

```
> Verdoorn.FEslag = spml(gy~gx, data=VerdoornALQ.pdfo, listw=W5EX.lw,
model="within", effect="twoways", lag=TRUE, spatial.error="none", quiet=FALSE)

 Spatial Lag Fixed Effects Model

Spatial fixed effects model
 Jacobian calculated using neighbourhood matrix eigenvalues
Computing eigenvalues ...

lambda:  -0.4231936    function value:  28.41245
lambda:  0.120418      function value:  25.22832
lambda:  -0.7591641    function value:  28.21717
lambda:  -0.551478     function value:  28.61451
lambda:  -0.5631885    function value:  28.61771
lambda:  -0.5720145    function value:  28.61829
lambda:  -0.5708659    function value:  28.6183
lambda:  -0.570832     function value:  28.6183
lambda:  -0.5708345    function value:  28.6183
```

```
lambda:  -0.5708345     function value:  28.6183
lambda:  -0.5708345     function value:  28.6183
lambda:  -0.5708345     function value:  28.6183
```

After assigning the last function value of 23.46787 to the slot logLik,

```
> Verdoorn.FEslag$logLik = 28.6183
> Verdoorn.FEslag$logLik
[1] 28.6183,
```

information criteria like AIC and BIC can be computed.

Loading user-defined function AICsplm

```
> source("AICsplm.R")
```

and computing AIC and BIC:

```
> AICsplm (Verdoorn.FEslag, criterion="AIC")
    AIC
-37.2366
> AICsplm (Verdoorn.FEslag, criterion="BIC")
    BIC
-30.1561
```

6.4 Spatial panel fixed effect (FE) error model

Spatial error model (SEM) with region fixed effects	Spatial error model (SEM) with region and time fixed effects
(14) $y_{rt} = \alpha_r + \sum_{j=1}^{m} \beta_j \cdot x_{jrt} + \varepsilon_{rt}.$ $\varepsilon_{rt} = \rho \cdot SL(\varepsilon_{rt}) + v_{rt}$	(15) $y_{rt} = \alpha_r + \alpha_t + \sum_{j=1}^{m} \beta_j \cdot x_{jrt} + \varepsilon_{rt}$ $\varepsilon_{rt} = \rho \cdot SL(\varepsilon_{rt}) + v_{rt}$
$SL(\varepsilon_{rt})$: spatial lag of the the disturbance ε of region r and period t ρ: spatial autoregressive coefficient in the disturbance	

Maximum likelihood (ML) estimation of spatial panel FE error models can be accomplished for variables from data.frame objects (VerdoornALQ.df, VerdoornALQ,dfo) or pdata.frame objects (VerdoornALQ.pdf, VerdoornALQ,pdfo).

A. Spatial error model with region (individual) fixed effect

```
> Verdoorn.FEindivserr = spml(gy~gx, data=VerdoornALQ.pdfo, listw=W5EX.lw,
model="within", effect="individual", lag=FALSE, spatial.error="kkp")
> summary(Verdoorn.FEindivserr)
Spatial panel fixed effects error model

Call:
spml(formula = gy ~ gx, data = VerdoornALQ.pdfo, listw = W5EX.lw,
    model = "within", effect = "individual", lag = FALSE, spatial.error = "kkp")

Residuals:
   Min.    1st Qu.  Median  3rd Qu.    Max.
-0.1300 -0.0333  -0.0032  0.0535   0.0667

Spatial error parameter:
       Estimate   Std. Error t-value    Pr(>|t|)
rho 0.061589   0.267192  0.2305   0.8177

Coefficients:
     Estimate   Std. Error t-value    Pr(>|t|)
gx  1.03202    0.18875  5.4676 4.561e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- **Standard error of regression (SER) and coefficient of determination (Pseudo-R^2)**

Degrees of freedom: (Txn=3x5=15) – (k+1=2) – (n=5) = 8
When the spatial panel FE error model is estimated, R reports the `intercept’ with the function effects.splm. However, fixed effect does not have the intercept. What R is reporting is not an estimated parameter but the average value of the fixed effed coefficients.

```
> s2.resVerdFEindivserr = sum(Verdoorn.FEindivserr$resid^2)/8
> s2.resVerdFEindivserr
[1] 0.00555633
> s.resVerdFEindivserr = sqrt(s2.resVerdFEindivserr)
> s.resVerdFEindivserr
[1] 0.07454079

> R2.VerdFEindivserr = 1-var(Verdoorn.FEindivserr$resid)/var(VerdoornALQ.dfo$gy)
> R2.VerdFEindivserr
[1] 0.9732102
or
> R2.VerdFEindivserr = summary(Verdoorn.FEindivserr)$rsqr
> R2.VerdFEindivserr
[1] 0.9732102
```

- **Extracting and testing fixed effects: fixedef.plm**

```
> effects.splm(Verdoorn.FEindivserr)
or
> effects(Verdoorn.FEindivserr)

Intercept:
            Estimate Std. Error t-value Pr(>|t|)
(Intercept) -0.77515    0.31270 -2.4789  0.01318 *

Spatial fixed effects:
   Estimate Std. Error t-value Pr(>|t|)
1  0.501134   0.144580  3.4661 0.000528 ***
2  0.373262   0.209364  1.7828 0.074613 .
3  0.074988   0.296484  0.2529 0.800327
4 -0.784227   0.487151 -1.6098 0.107436
5 -0.165156   0.433957 -0.3806 0.703513
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

B. Spatial error model with region and time (twoways) fixed effect

```
> Verdoorn.FEserr = spml(gy~gx, data=VerdoornALQ.pdfo, listw=W5EX.lw,
model="within", effect="twoways", lag=FALSE, spatial.error="kkp")

> summary(Verdoorn.FEserr)
Spatial panel fixed effects error model

Call:
spml(formula = gy ~ gx, data = VerdoornALQ.pdfo, listw = W5EX.lw,
    model = "within", effect = "twoways", lag = FALSE, spatial.error = "kkp")

Residuals:
    Min.  1st Qu.   Median  3rd Qu.     Max.
-0.11000 -0.02190 -0.00987  0.04090  0.06950

Spatial error parameter:
      Estimate   Std. Error t-value   Pr(>|t|)
rho -0.53709    0.24881 -2.1586  0.03088 *

Coefficients:
     Estimate  Std. Error t-value    Pr(>|t|)
gx  0.62401    0.19582  3.1866  0.00144 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- **Standard error of regression (SER) and coefficient of determination (Pseudo-R²)**

Degrees of freedom: (Txn=3x5=15) – (k+1=2) – (n=5) – (T=3) + 1 = 6

To avoid exact multicollinearity, not all n+T=5+3=8 region and time effects can be estimated. One region or period must be used as a reference which explains the term “+1” in the computation of the degrees of freedom.

```
> s2.resVerdFEserr = sum(Verdoorn.FEserr$resid^2)/6
> s2.resVerdFEserr
[1] 0.005237985

> s.resVerdFEserr = sqrt(s2.resVerdFEserr)
> s.resVerdFEserr
[1] 0.07237393

> R2.VerdFEserr = 1-var(Verdoorn.FEserr$resid)/var(VerdoornALQ.dfo$gy)
> R2.VerdFEserr
[1] 0.9810589
or
> R2.VerdFEserr = summary(Verdoorn.FEserr)$rsqr
> R2.VerdFEserr
[1] 0.9810589
```

- **Extracting and testing fixed effects: effects.splm**

```
> effects.splm(Verdoorn.FEserr)
or
> effects(Verdoorn.FEserr)

Intercept:
               Estimate   Std. Error  t-value    Pr(>|t|)
(Intercept) -0.097852   0.285606 -0.3426   0.7319

Spatial fixed effects:
   Estimate   Std. Error  t-value    Pr(>|t|)
1  0.129847   0.130942  0.9916   0.3214
2  0.144777   0.190496  0.7600   0.4473
3  0.036907   0.270333  0.1365   0.8914
4 -0.407500   0.444766 -0.9162   0.3596
5  0.095969   0.396120  0.2423   0.8086

Time period fixed effects:
  Estimate   Std. Error  t-value   Pr(>|t|)
1 -0.06056    0.27567  -0.2197   0.8261
2  0.02576    0.28768  0.0895    0.9287
3  0.03480    0.29455   0.1181   0.9060
```

6.4 Spatial panel fixed effect (FE) Durbin model

Spatial cross-regressive (SLX) model with region fixed effects	Spatial cross-regressive (SLX) model with region and time fixed effects
(16) $y_{rt} = \alpha_r + \lambda \cdot SL(y_{rt}) + \sum_{j=1}^{m} \beta_j \cdot x_{jrt} + \sum_{j=1}^{m} \theta_j \cdot SL(x_{jrt}) + \varepsilon_{rt}$	(17) $y_{rt} = \alpha_r + \alpha_t + \lambda \cdot SL(y_{rt}) + \sum_{j=1}^{m} \beta_j \cdot x_{jrt} + \sum_{j=1}^{m} \theta_j \cdot SL(x_{jrt}) + \varepsilon_{rt}$

Maximum likelihood (ML) estimation of spatial panel FE error models can be accomplished for variables from data.frame objects (VerdoornALQ.df, VerdoornALQ,dfo) or pdata.frame objects (VerdoornALQ.pdf, VerdoornALQ,pdfo).

A. Spatial Durbin model with region (individual) fixed effect

```
> Verdoorn.FEindivDurbin = spml(gy~gx+SLgx, data=VerdoornALQ.pdfo,
listw=W5EX.lw, model="within", effect="individual", lag=TRUE, spatial.error="none")

> summary(Verdoorn.FEindivDurbin)
Spatial panel fixed effects lag model

Call:
spml(formula = gy ~ gx + SLgx, data = VerdoornALQ.pdfo, listw = W5EX.lw,
   model = "within", effect = "individual", lag = TRUE, spatial.error = "none")

Residuals:
   Min.  1st Qu.   Median  3rd Qu.     Max.
-0.13300 -0.02900 -0.00654  0.04880  0.07300

Spatial autoregressive coefficient:
          Estimate   Std. Error  t-value    Pr(>|t|)
lambda 0.025261   0.269927  0.0936   0.9254

Coefficients:
        Estimate  Std. Error t-value      Pr(>|t|)
gx      1.02851    0.18858  5.4541    4.923e-08 ***
SLgx  0.18354    0.48205  0.3807    0.7034
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- **Standard error of regression (SER) and coefficient of determination (Pseudo-R²)**

Degrees of freedom: (Txn=3x5=15) – (k+1=3) – (n=5) = 7

When the spatial panel FE Durbin model is estimated, R reports the `intercept' with the function effects.splm. However, fixed effect does not have the intercept. What R is reporting is not an estimated parameter but the average value of the fixed effed coefficients.

```
> s2.resVerdFEindivDurbin = sum(Verdoorn.FEindivDurbin$resid^2)/7
> s2.resVerdFEindivDurbin
[1] 0.006203668

> s.resVerdFEindivDurbin = sqrt(s2.resVerdFEindivDurbin)
> s.resVerdFEindivDurbin
[1] 0.07876337

> R2.VerdFEindivDurbin = 1-
var(Verdoorn.FEindivDurbin$resid)/var(VerdoornALQ.dfo$gy)
> R2.VerdFEindivDurbin
[1] 0.973828
or
> R2.VerdFEindivDurbin = summary(Verdoorn.FEindivDurbin)$rsqr
> R2.VerdFEindivDurbin
[1] 0.973828
```

- **Extracting and testing fixed effects: effects.splm**

```
> effects.splm(Verdoorn.FEindivDurbin)
or
> effects(Verdoorn.FEindivDurbin)

Intercept:
            Estimate Std. Error t-value Pr(>|t|)
(Intercept) -1.11120    0.68128 -1.6311   0.1029

Spatial fixed effects:
  Estimate Std. Error t-value Pr(>|t|)
1  0.57427    0.49789  1.1534   0.2487
2  0.39216    0.61629  0.6363   0.5246
3  0.12561    0.59121  0.2125   0.8317
4 -0.76892    0.73731 -1.0429   0.2970
5 -0.32312    1.00295 -0.3222   0.7473
```

- **Computing logLik, AIC and BIC**

```
> Verdoorn.FEindivDurbin = spml(gy~gx+SLgx, data=VerdoornALQ.pdfo, listw = W5EX.lw, model="within", effect="individual", lag=TRUE, spatial.error="none", quiet = FALSE)


 Spatial Lag Fixed Effects Model
```

```
Spatial fixed effects model
 Jacobian calculated using neighbourhood matrix eigenvalues
Computing eigenvalues ...

lambda:  -0.4231936     function value:  22.60583
lambda:  0.120418       function value:  23.48
lambda:  0.4563884      function value:  22.54439
lambda:  0.009604227    function value:  23.52221
lambda:  0.0235714      function value:  23.52335
lambda:  0.02512894     function value:  23.52336
lambda:  0.02526333     function value:  23.52336
lambda:  0.02526126     function value:  23.52336
lambda:  0.02526125     function value:  23.52336
lambda:  0.02526124     function value:  23.52336
lambda:  0.02521071     function value:  23.52336
lambda:  0.02524194     function value:  23.52336
lambda:  0.02525387     function value:  23.52336
lambda:  0.02525842     function value:  23.52336
lambda:  0.02526016     function value:  23.52336
lambda:  0.02526083     function value:  23.52336
lambda:  0.02526108     function value:  23.52336
lambda:  0.02526118     function value:  23.52336
lambda:  0.02526122     function value:  23.52336
lambda:  0.02526123     function value:  23.52336
lambda:  0.02526123     function value:  23.52336
lambda:  0.02526123     function value:  23.52336
```

After assigning the last function value of 23.52336 to the slot logLik,

```
> Verdoorn.FEindivDurbin$logLik = 23.52336
> Verdoorn.FEindivDurbin$logLik
[1] 23.52336,
```

information criteria like AIC and BIC can be computed.

Loading user-defined function AICsplm

```
> source("AICsplm.R")
```

and computing AIC and BIC:

```
> AICsplm (Verdoorn.FEindivDurbin, criterion="AIC")
     AIC
-29.04672
> AICsplm (Verdoorn.FEindivDurbin, criterion="BIC")
     BIC
-22.67427
```

B. Spatial Durbin model with region and time (twoways) fixed effect

```
> Verdoorn.FEDurbin = spml(gy~gx+SLgx, data=VerdoornALQ.pdfo, listw=W5EX.lw,
model="within", effect="twoways", lag=TRUE, spatial.error="none")

> summary(Verdoorn.FEDurbin)
Spatial panel fixed effects lag model

Call:
spml(formula = gy ~ gx + SLgx, data = VerdoornALQ.pdfo, listw = W5EX.lw,
   model = "within", effect = "twoways", lag = TRUE, spatial.error = "none",
   quiet = FALSE)

Residuals:
   Min.  1st Qu.   Median  3rd Qu.     Max.
-0.03860 -0.01470 -0.00579  0.01510  0.05410

Spatial autoregressive coefficient:
          Estimate  Std. Error  t-value   Pr(>|t|)
lambda -0.38615    0.23296 -1.6576   0.0974 .

Coefficients:
        Estimate  Std. Error  t-value   Pr(>|t|)
gx       0.24140    0.16667  1.4484 0.1475179
SLgx -1.44062    0.40475 -3.5593 0.0003718 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- **Standard error of regression (SER) and coefficient of determination (Pseudo-R²)**

Degrees of freedom: (Txn=3x5=15) – (k+1=3) – (n=5) – (T=3) + 1 = 5

To avoid exact multicollinearity, not all n+T=5+3=8 region and time effects can be estimated. One region or period must be used as a reference which explains the term “+1” in the computation of the degrees of freedom.

```
> s2.resVerdFEDurbin = sum(Verdoorn.FEDurbin$resid^2)/5
> s2.resVerdFEDurbin
[1] 0.002014134
> s.resVerdFEDurbin = sqrt(s2.resVerdFEDurbin)
> s.resVerdFEDurbin
[1] 0.04487911
> R2.VerdFEDurbin = 1-var(Verdoorn.FEDurbin$resid)/var(VerdoornALQ.dfo$gy)
> R2.VerdFEDurbin
[1] 0.9939306
or
> R2.VerdFEDurbin = summary(Verdoorn.FEDurbin)$rsqr
> R2.VerdFEDurbin
[1] 0.9939306
```

- **Extracting and testing fixed effects: effects.splm**

```
> effects.splm(Verdoorn.FEDurbin)
or
> effects(Verdoorn.FEDurbin)

Intercept:
              Estimate  Std. Error  t-value    Pr(>|t|)
(Intercept)  3.39292    0.88267   3.8439  0.0001211 ***

Spatial fixed effects:
   Estimate  Std. Error t-value    Pr(>|t|)
1 -0.83456    0.60929 -1.3697   0.1708
2 -0.24731    0.76947 -0.3214   0.7479
3 -0.42432    0.77207 -0.5496   0.5826
4 -0.12795    0.98042 -0.1305   0.8962
5  1.63415    1.29039  1.2664   0.2054

Time period fixed effects:
   Estimate    Std. Error t-value     Pr(>|t|)
1 -0.161868   0.860741 -0.1881   0.8508
2  0.044256   0.885335  0.0500   0.9601
3  0.117611   0.902104  0.1304   0.8963
```

- **Computing logLik, AIC and BIC**

```
> Verdoorn.FEDurbin = spml(gy~gx+SLgx, data=VerdoornALQ.pdfo, listw=W5EX.lw,
model="within", effect="twoways", lag=TRUE, spatial.error="none", quiet=FALSE)

Spatial Lag Fixed Effects Model

Spatial fixed effects model
 Jacobian calculated using neighbourhood matrix eigenvalues
Computing eigenvalues ...

lambda:  -0.4231936    function value:  33.14024
lambda:  0.120418      function value:  30.62385
lambda:  -0.7591641    function value:  31.32618
lambda:  -0.3543899    function value:  33.14502
lambda:  -0.3861533    function value:  33.15729
lambda:  -0.3859683    function value:  33.15729
lambda:  -0.3861493    function value:  33.15729
lambda:  -0.3861484    function value:  33.15729
lambda:  -0.3861484    function value:  33.15729
lambda:  -0.3861483    function value:  33.15729
lambda:  -0.3861483    function value:  33.15729
lambda:  -0.3861484    function value:  33.15729
```

After assigning the last function value of 33.15729 to the slot logLik,

```
> Verdoorn.FEDurbin$logLik = 33.15729
> Verdoorn.FEDurbin$logLik
[1] 33.15729,
```

information criteria like AIC and BIC can be computed.

Loading user-defined function AICsplm

```
> source("AICsplm.R")
```

and computing AIC and BIC:

```
> AICsplm (Verdoorn.FEDurbin, criterion="AIC")
      AIC
-44.31458
> AICsplm (Verdoorn.FEDurbin, criterion="BIC")
     BIC
-36.52603
```

7. Applications of spatial data analysis

In the preceding chapters, spatial data analysis is illustrated with an example of five artificial regions. For graphical purposes, a SpatialPolygons object was formed from a textfile with coordinates of the boundaries of the example regions. Maps of the regions can be drawn with the plot command. When a SpatialPolygons file is used as an argument of the plot command, choropleth maps for discrete or grouped variables can be created.

Here, data material and geographical information are provided for accomplishing spatial data analysis of two applications:

- Effects of industrial clusters on regional performance in Germany,
- Existence of a wage curve in East Germany.

Maps for countries need not be created from scratch. In fact, geographical information on boundaries of administrative regions are available at different spatial scales in shape files. They are provided by national and supranational statistical offices. For drawing maps in R, a shape file has first to be transformed into a SpatialPolygonsDataFrame object. After merging the SpatialPolygonsDataFrame file with a data.frame object, discrete or grouped georeferenced variables can be portrayed in form of choropleth maps.

In principle, it is possible to create spatial weights matrices from geographical information stored in shape files. However, at the present stage this cannot be accomplished inside R but with the aid of the program GEODA.[1] For this reason, spatial weights matrices are provided for Germany as a whole and East Germany for a regional disaggregation at the level of NUTS-3 regions (urban and rural districts).

Area data for both applications are provided in CSV files. While cross-sectional data are available for evaluating cluster effects, spatial panel data is supplied to test the existence of a wage curve in East Germany. Both applications are examples from current empirical research projects. The presentations on the results of spatial data analyses by seminar participants will take place in the second part of the seminar "Spatial Econometrics".

7.1 Application 1: Effects of regional industrial clusters in Germany

For evaluating economic effects of regional industry cluster in Germany, knowledge on their location and scope must be available. While many case studies are available, quantitative investigations for whole countries using advanced methods are rather scarce. Here we draw on the top-down study of Kosfeld and Titze (2017) who identified clusters of German R&D-intensive industries at the level of NUT-3 regions for the year 2006.[2] In a first step, cluster templates in the form of value-added chains of R&D-intensive industries are formed by using all interindustry linkages included in the 2006 German input-output table. The significant flows are determined by

---

[1] See Anselin (2007), Spatial Regression Analysis in R – A Workbook, Center of Spatially Integrated Social Sciences (CSISS).

[2] Kosfeld and Titze (2017), Benchmark Value-Added Chains and Regional Clusters in R&D-Intensive Industries, International Regional Science Review 40, 530-558.

qualitative input-output analysis (QIOA). In a second step, regional industry clusters are identified with the aid of Kulldorff's spatial scan method.

The spatial scan for delineating industrial clusters was carried out over German 439 NUTS-3 regions existing in 2006. After the local government reorganization in Saxony-Anhalt (2007), Saxony (2008) and Mecklenburg-Western Pomerania (2011) the number of NUT-3 regions is reduced to 402. As area data for control variables are often only available for the new district bondaries, this regional breakdown should be used for evaluating effects of clusters. Thus, information on industrial clusters is adjusted for 402 NUT-3 regions after the district reforms.

- **Reading shape file NUTS3GER402 in a SpatialPolygonsDataFrame**

Loading package sp:
```
> library(sp)
```
Loading package spdep:
```
> library(spdep)
```

Install package rgeos used by maptools for polygon geometry computation.
Loading package maptools:
```
> library(maptools)
```

Reading shape file NUTS3GER402 (projected):
```
> NUTS3GER402.spdf = readShapePoly("NUTS3GER402",proj4string =
CRS("+proj=utm +zone=32 +datumETRS89"))
> class(NUTS3GER402.spdf)
[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"
```

The file NUTS3GER402.spdf created by applying the readShapePoly command does not belong to the class SpatialPolygons but already to the class SpatialPolygonsDataFrame. The file contains information on the boundary coordinates and area of the districts as well as several identification and geographical variables like SP_ID (consecutive numbers of the districts), RS and KREISNR (official numbers of the districts), KREIS (district names) and KREISTYP (district types).

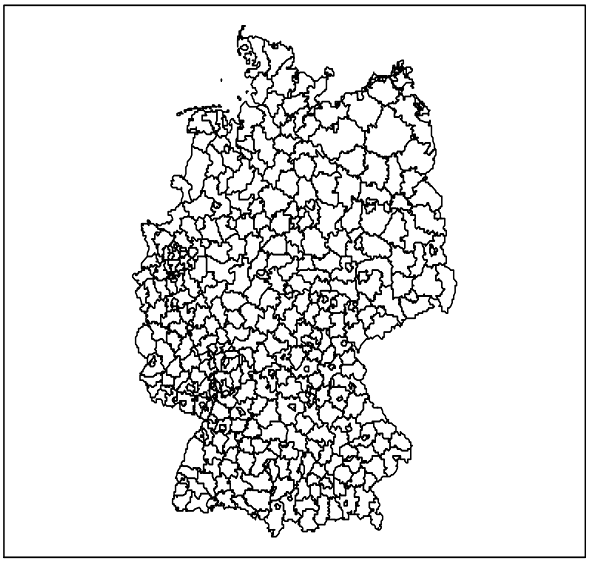
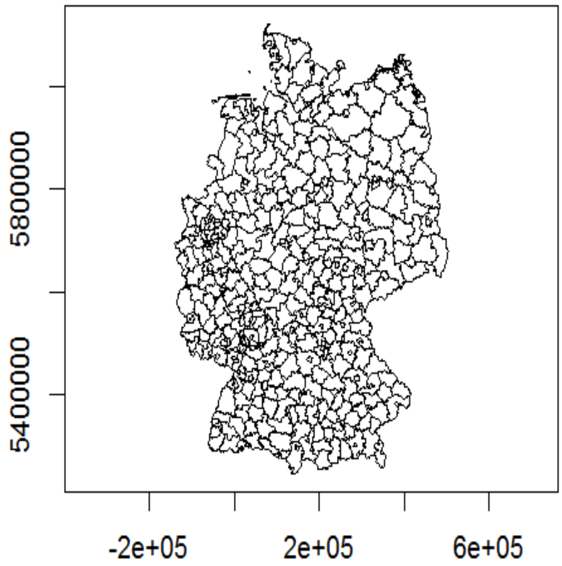
```
> summary(NUTS3GER402.spdf)
Object of class SpatialPolygonsDataFrame
Coordinates:
        min     max
x -139647.8  502633
y 5246514.0 6121038
Is projected: TRUE
proj4string : [+proj=utm +zone=32 +datumETRS89]
Data attributes:
    SP_ID         RS              KREIS               KREISTYP
 0     :  1   01001  :  1   Ansbach     :  2   Kreis         : 42
```

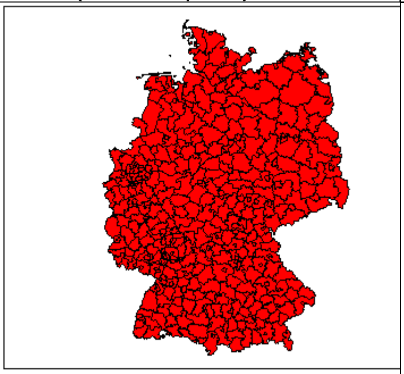
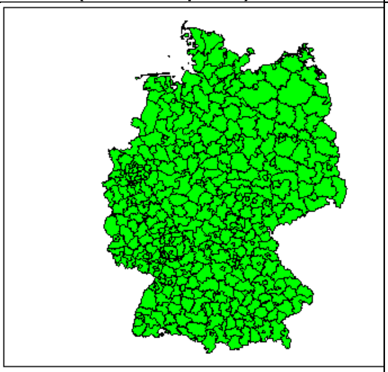
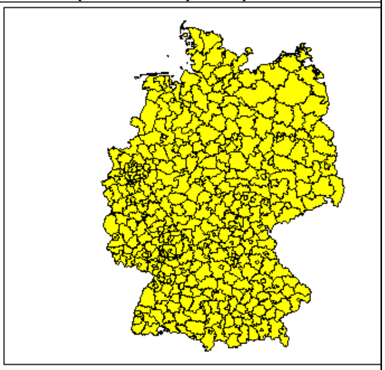
```
1      :  1   01002  :  1   Aschaffenburg:  2   Kreisfreie Stadt: 98
10     :  1   01003  :  1   Augsburg     :  2   Landkreis        :252
100    :  1   01004  :  1   Bamberg      :  2   Regionalverband :  1
101    :  1   01051  :  1   Bayreuth     :  2   Stadtkreis       :  9
102    :  1   01053  :  1   Coburg       :  2
(Other):396   (Other):396   (Other)      :390
  PROZ_NEHE          X1             KREISNR
Min.   :16.72   Min.   :-25.94   Min.   : 1001
1st Qu.:25.47   1st Qu.:  2.49   1st Qu.: 5759
Median :29.65   Median : 19.66   Median : 8233
Mean   :35.15   Mean   : 67.62   Mean   : 8293
3rd Qu.:37.14   3rd Qu.: 57.52   3rd Qu.: 9676
Max.   :70.87   Max.   :392.81   Max.   :16077
```

- **Elementary Maps from SpatialPolygonsDataFrame NUTS3GER402.spdf**

The maps can be copied from R as metafiles:
- right mouse click
- select “Copy as bitmap” or “Copy as metafile”.

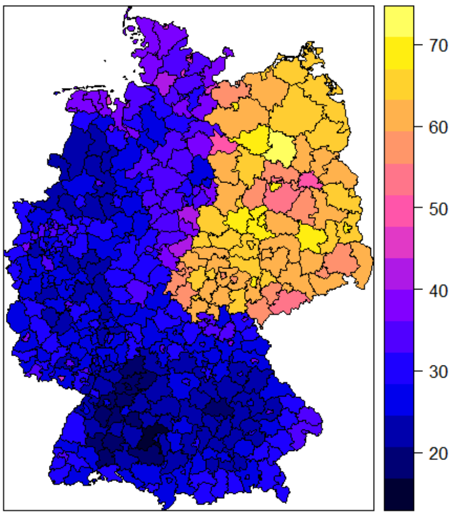
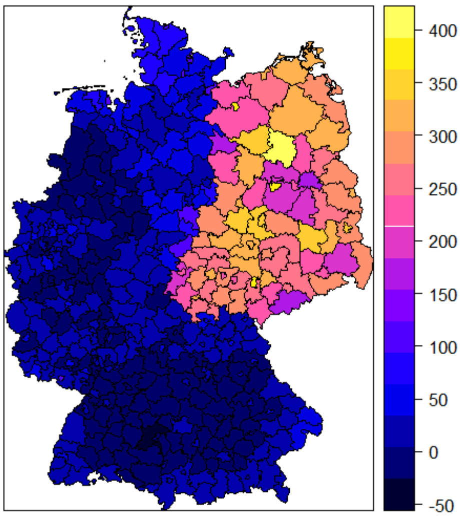
After including them in Word, the fringes may be cutted:
- Format
- Zuschneiden (cut).

Map of 402 German NUTS-3 regions	
without axis	with axis
> plot(NUTS3GER402.spdf) > box(which="plot")	> plot(NUTS3GER402.spdf, axes=TRUE)
	

Plot of maps with all polygons filled with		
red colour	green colur	yellow colour
> plot(NUTS3GER402.spdf, col="red") > box(which="plot")	> plot(NUTS3GER402.spdf, col="green") > box(which="plot")	> plot(NUTS3GER402.spdf, col="yellow") > box(which="plot")
		

- **Lattice (trellis) plot of attributes of SpatialPolygonsDataFrame**

Maps of quantitative geographical variables PROZ_NEHE and X1 from SpatialPolygonsDataFrame NUTS3GER402.spdf:

Maps quantitative geographical variables from SpatialPolygonsDataFrame NUTS3GER402.spdf	
> spplot(NUTS3GER402.spdf, c("PROZ_NEHE"))	> spplot(NUTS3GER402.spdf, c("X1"))
	

- **Reading CSV data file in a data.frame object**

Reading CSV file Clusterdata402Kr.CSV with information on regional R&D clusters, target and control variables at the district level in a data.frame object:

```
> Clusterdata402Kr.df = read.csv("Clusterdata402Kr.CSV", header = TRUE)

> class(Clusterdata402Kr.df)
[1] "data.frame"

> dim(Clusterdata402Kr.df)
[1] 402  65

> str(Clusterdata402Kr.df)
'data.frame':   402 obs. of  65 variables:
 $ KREISNR       : int  1001 1002 1003 1004 1051 1053 1054 1055 1056 1057 ...
 $ KREISNAME     : Factor w/ 383 levels "Aachen, Städteregion",..: 94 158 196 231
67 139 239 263 270 272 ...
 $ Cln           : int  1 1 1 1 1 1 0 1 1 1 ...
 $ Clkombi       : Factor w/ 92 levels
"0Cl","Auto","Auto.Chemie.CommE.IT.Pharma",..: 65 44 91 65 69 89 1 91 39 50 ...
 $ AutoCl        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ ChemieCl      : int  0 1 0 0 0 0 0 0 1 1 ...
 $ CommEquipCl   : int  1 0 0 1 1 0 0 0 1 0 ...
 $ ElecMachCl    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ ITCl          : int  0 1 0 0 0 0 0 0 0 0 ...
 $ MachCl        : int  1 1 0 1 0 1 0 0 0 0 ...
 $ MedInstCl     : int  0 0 1 0 0 0 0 1 1 1 ...
 $ PharmaCl      : int  0 0 1 0 1 1 0 1 0 0 ...
 $ nClNom        : Factor w/ 6 levels "0 Cl","1 Cl",..: 3 4 3 3 3 3 1 3 4 3 ...
 $ nCl           : int  2 3 2 2 2 2 0 2 3 2 ...
 $ Cl0           : int  0 0 0 0 0 0 1 0 0 0 ...
 $ Cl1           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Cl2           : int  1 0 1 1 1 1 0 1 0 1 ...
 $ Cl3           : int  0 1 0 0 0 0 0 0 1 0 ...
 $ Cl4           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Clmore4       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ BWS2001       : num  2346 7291 5291 1955 2911 ...
 $ BWS2006       : num  2856 7723 5507 2102 2985 ...
 $ BIP2001       : num  2596 8070 5856 2164 3222 ...
 $ BIP2006       : num  3157 8538 6088 2324 3301 ...
 $ ET2001        : num  56.9 154.4 117.9 46.6 59.5 ...
 $ ET2006        : num  56.5 154.5 115.5 43.9 55.3 ...
 $ ProdETBWS01   : num  41.3 47.2 44.9 41.9 48.9 ...
 $ ProdETBWS06   : num  50.5 50 47.7 47.9 54 ...
 $ gPrETBWS01_06 : num  0.225 0.058 0.063 0.141 0.104 0.354 0.066 -0.002 0.092
0.146 ...
 $ ProdETBIP01   : num  45.7 52.3 49.7 46.4 54.2 ...
 $ ProdETBIP06   : num  55.9 55.3 52.7 52.9 59.7 ...
 $ gPrETBIP01_06 : num  0.224 0.057 0.062 0.14 0.103 0.353 0.065 -0.003 0.091
0.145 ...
```

```
 $ Workh2001     : num  79.7 219.6 167.3 65.7 86.4 ...
 $ Workh2006     : num  78.4 217.8 162.3 61.2 78.5 ...
 $ ProdhBWS01    : num  29.4 33.2 31.6 29.8 33.7 ...
 $ ProdhBWS06    : num  36.4 35.5 33.9 34.3 38 ...
 $ gPrhBWS01_06  : num  0.237 0.068 0.072 0.153 0.129 0.374 0.083 0.014 0.1
0.168 ...
 $ ProdhBIP01    : num  32.6 36.8 35 33 37.3 ...
 $ ProdhBIP06    : num  40.3 39.2 37.5 37.9 42.1 ...
 $ gPrhBIP01_06  : num  0.236 0.066 0.071 0.151 0.128 0.373 0.082 0.013 0.099
0.167 ...
 $ Pop2001       : int  84480 232242 213496 79646 137447 181661 165026 203386
293914 133624 ...
 $ Pop2006       : int  86630 235366 211213 77936 136829 186911 166783 205952
300402 135562 ...
 $ Area          : num  56.7 118.7 214.2 71.6 1428.1 ...
 $ Dichte        : num  1526.8 1983.7 986 1088 95.8 ...
 $ SVB2006       : int  36517 100359 76968 28862 32982 38041 45860 49160 74600
21833 ...
 $ SVBfem2006    : int  17160 47892 37694 12547 14613 18654 21954 25404 34254
10748 ...
 $ qSVBfem06     : num  0.47 0.477 0.49 0.435 0.443 0.49 0.479 0.517 0.459 0.492
...
 $ SVBpart2006   : int  7651 22510 15893 5112 5946 7637 8455 9269 13226 4846 ...
 $ qSVBpart06    : num  0.21 0.224 0.206 0.177 0.18 0.201 0.184 0.189 0.177 0.222
...
 $ SVBuni2006    : int  2185 10392 5400 1586 1589 2209 1688 2106 5373 927 ...
 $ qSVBuni06     : num  0.06 0.104 0.07 0.055 0.048 0.058 0.037 0.043 0.072 0.042
...
 $ SVBnoqual2006 : int  5062 12465 10001 3838 4658 4802 5377 6424 10035 2972
...
 $ qSVBnoqual06  : num  0.139 0.124 0.13 0.133 0.141 0.126 0.117 0.131 0.135
0.136 ...
 $ nPlants2006   : int  2230 5585 5146 2011 3599 4338 6131 5999 7489 3044 ...
 $ sizePlants2006: num  16.4 18 15 14.4 9.2 8.8 7.5 8.2 10 7.2 ...
 $ SVBserv2006   : int  27424 82890 58884 21244 22001 26047 36153 36924 48179
15841 ...
 $ qSVBserv06    : num  0.751 0.826 0.765 0.736 0.667 0.685 0.788 0.751 0.646
0.726 ...
 $ SVBagrar2006  : int  86 252 374 281 989 894 1070 1035 2429 768 ...
 $ qSVBagrar06   : num  0.0024 0.0025 0.0049 0.0097 0.03 0.0235 0.0233 0.0211
0.0326 0.0352 ...
 $ SVBind2006    : int  7420 12836 13339 5055 6560 8128 3251 6943 17498 2676 ...
 $ qSVBind06     : num  0.203 0.128 0.173 0.175 0.199 ...
 $ SVByoung2006  : int  7946 22001 17197 6360 7736 8455 12121 11890 16536
4936 ...
 $ qSVByoung06   : num  0.218 0.219 0.223 0.22 0.235 ...
 $ SVBold2006    : int  8848 24196 17608 6739 7656 8726 10187 11466 16766 5088
...
 $ qSVBold06     : num  0.242 0.241 0.229 0.234 0.232 ...
```

Target variables:

Economic variable	Indicator	R-Variable
productivity (2006)	GRP per employee	ArbeitsProdETBIP06
	GVA per employee	ProdETBWS06
	GRP per man hour	ProdhBIP06
	GVA per man hour	ProdhBWS06
Productivity growth (2001 – 2006)	Growht rate of GRP per employee	gPrETBIP01_06
	Growth rate of GVA per employee	gPrETBWS01_06
	Growth rate of GRP per ma hour	gPrhBIP01_06
	Growth rate of GVA per man hour	gPrhBWS01_06

GRP: gross regional product, GVA: gross value added

```
> Clusterdata402Kr.df$KREISNAME =
as.character(Clusterdata402Kr.df$KREISNAME)

> class(Clusterdata402Kr.df$KREISNAME)
[1] "character"

> Clusterdata402Kr.df[1:3, 1:7]
  KREISNR KREISNAME Cln       Clkombi      AutoCl   ChemieCl   CommEquipCl
1    1001      Flensburg    1     CommE.Mach       0            0              1
2    1002          Kiel          1    Chemie.IT.Mach    0            1              0
3    1003        Lübeck       1      MedI.Pharma       0            0              0

> Clusterdata402Kr.df[1:3, 60:65]
  SVBind2006 qSVBind06 SVByoung2006 qSVByoung06 SVBold2006 qSVBold06
1       7420         0.2032            7946              0.2176          8848          0.2423
2      12836        0.1279          22001              0.2192         24196         0.2411
3      13339        0.1733          17197              0.2234         17608         0.2288
```

- **Ordering of SpatialPolygonsDataFrame NUTS3GER402.spdf by Column KREISNR**

```
> NUTS3GER402.spdf$KREISNR == Clusterdata402Kr.df$KREISNR
  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
FALSE FALSE
 [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE FALSE
 [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
TRUE  TRUE
…
[337]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
FALSE FALSE
```

```
[349] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE
FALSE FALSE
[361]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
FALSE FALSE
[373] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
TRUE  TRUE
[385]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
TRUE  TRUE
[397]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE

> NUTS3GER402.spdf$KREISNR
  [1]  1001  1002  1003  1004  1051  1053  1054  1055  1057  1058  1059  1060
 [13]  1061  1062  2000  3101  3102  3103  3151  3152  3153  3154  3155  3156
 [25]  3157  3158  3241  3251  3252  3254  3255  1056  3256  3257  3351  3352
…
[337] 12066 12067 12068 12069 12070 12071 12072 12073 13003 13004 13072
13076
[349] 13071 13073 13074 13075 14511 14522 14523 14521 14524 14612 14626
14625
[361] 14627 14628 14713 14729 14730 15001 15082 15091 15002 15084 15088
15087
[373] 15003 15085 15086 15083 15090 15089 15081 16051 16052 16053 16054
16055
[385] 16056 16061 16062 16063 16064 16065 16066 16067 16068 16069 16070
16071
[397] 16072 16073 16074 16075 16076 16077

> NUTS3GER402.spdf =
NUTS3GER402.spdf[order(NUTS3GER402.spdf$KREISNR),]

> NUTS3GER402.spdf$KREISNR == Clusterdata402Kr.df$KREISNR
  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE TRUE
 [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE TRUE TRUE
 [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE TRUE TRUE
…
[286] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE TRUE TRUE
[301] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE TRUE TRUE
[316] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE TRUE TRUE
[331] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE TRUE TRUE

> NUTS3GER402.spdf$KREISNR
  [1]  1001  1002  1003  1004  1051  1053  1054  1055  1056  1057  1058  1059
 [13]  1060  1061  1062  2000  3101  3102  3103  3151  3152  3153  3154  3155
 [25]  3156  3157  3158  3241  3251  3252  3254  3255  3256  3257  3351  3352
```

```
…
[337] 12066 12067 12068 12069 12070 12071 12072 12073 13003 13004 13071
13072
[349] 13073 13074 13075 13076 14511 14521 14522 14523 14524 14612 14625
14626
[361] 14627 14628 14713 14729 14730 15001 15002 15003 15081 15082 15083
15084
[373] 15085 15086 15087 15088 15089 15090 15091 16051 16052 16053 16054
16055
[385] 16056 16061 16062 16063 16064 16065 16066 16067 16068 16069 16070
16071
[397] 16072 16073 16074 16075 16076 16077
```

- **Merging a SpatialPolygonsDataFrame and a data.frame object**

Merging the SpatialPolygonsDataFrame NUTS3GER402.spdf with the data.frame Clusterdata402Kr.df:

```
> Clusterdata402Kr.spdf = merge(NUTS3GER402.spdf, Clusterdata402Kr.df, by.x =
"KREISNR", by.y = "KREISNR")

> class(Clusterdata402Kr.spdf)
[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"

> summary(Clusterdata402Kr.spdf)
Object of class SpatialPolygonsDataFrame
Coordinates:
       min     max
x -139647.8  502633
y 5246514.0 6121038
Is projected: TRUE
proj4string : [+proj=utm +zone=32 +datumETRS89]
Data attributes:
    KREISNR          SP_ID          RS                KREIS
 Min.    : 1001      0     :  1   01001  :  1   Ansbach       :  2
 1st Qu. : 5759      1     :  1   01002  :  1   Aschaffenburg :  2
 Median : 8233      10     :  1   01003  :  1   Augsburg      :  2
 Mean    : 8293    100     :  1   01004  :  1   Bamberg       :  2
 3rd Qu. : 9676    101     :  1   01051  :  1   Bayreuth      :  2
 Max.    :16077    102     :  1   01053  :  1   Coburg        :  2
                   (Other) :396  (Other):396 (Other)         :390
            KREISTYP     PROZ_NEHE        X1              KREISNAME
 Kreis             :  42   Min.    : 16.72   Min.    : -25.94   Length : 402
 Kreisfreie Stadt  :  98   1st Qu. : 25.47   1st Qu. :   2.49   Class  : character
 Landkreis         : 252   Median : 29.65   Median :  19.66   Mode   : character
 Regionalverband :   1   Mean    : 35.15   Mean    :  67.62
 Stadtkreis        :   9   3rd Qu. : 37.14   3rd Qu. :  57.52
                           Max.    : 70.87   Max.    : 392.81
```

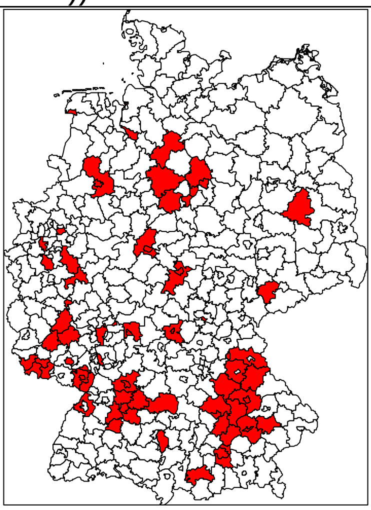
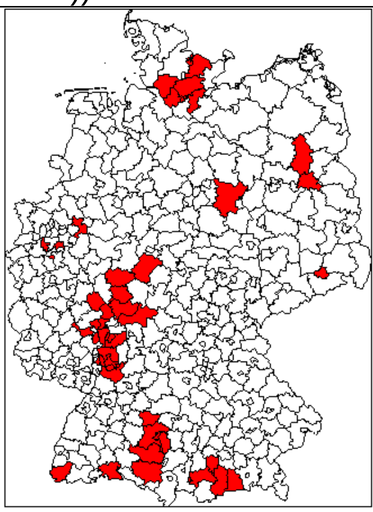
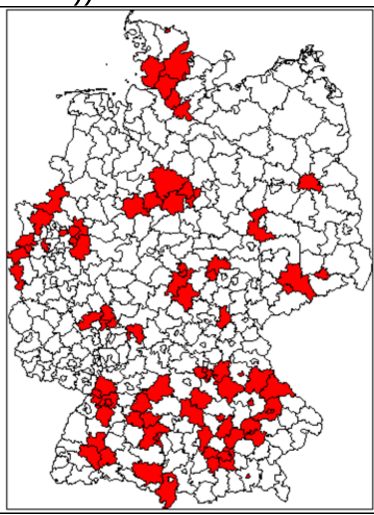
```
       Cln             Clkombi          AutoCl          ChemieCl
 Min.    : 0.000  0Cl          : 84  Min.    : 0.0000  Min.    : 0.0000
 1st Qu. : 1.000  Mach         : 44  1st Qu. : 0.0000  1st Qu. : 0.0000
 Median : 1.000   Pharma       : 20  Median : 0.0000   Median : 0.0000
 Mean   : 0.791   Auto         : 13  Mean   : 0.1766   Mean   : 0.1095
 3rd Qu.: 1.000   MedI         : 13  3rd Qu. : 0.0000  3rd Qu.: 0.0000
 Max.   : 1.000   ElecM.Mach : 12    Max.    : 1.0000  Max.    : 1.0000
                  (Other)      : 216
…
…
  SVBold2006      qSVBold06
 Min.    :  2647   Min.    : 0.1783
 1st Qu. :  6479   1st Qu. : 0.2174
 Median : 10232    Median : 0.2286
 Mean    : 15049   Mean    : 0.2309
 3rd Qu. : 17444   3rd Qu. : 0.2433
 Max.    :241229   Max.    : 0.2916
```

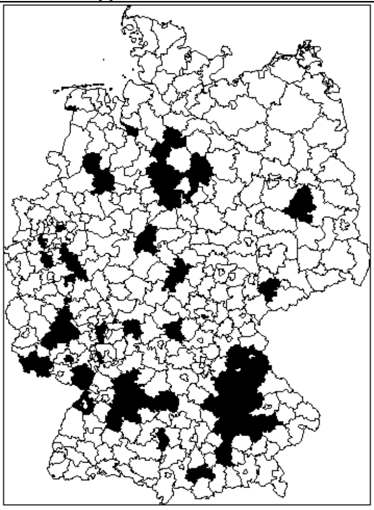
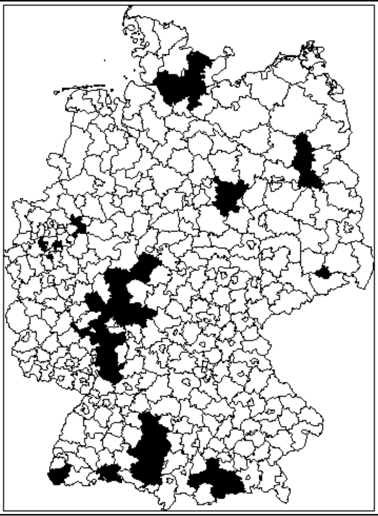
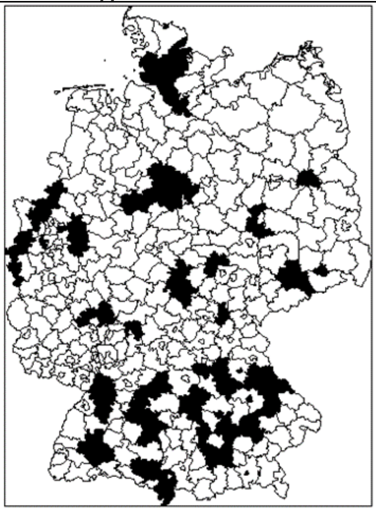
- **Plotting the spatial distributions of R&D clusters**

The maps maps can copied from R as metafiles:
- right mouse click
- select “Copy as bitmap” or “Copy as metafile”.

After including them in Word, the fringes can be cutted:
- Format
- Zuschneiden (cut).

By selecting the menu item “Zuschneiden” (cut) also potential legends can be removed if not needed.

spplot(Clusterdata402Kr.spdf, c("AutoCl"), col.regions=c("white", "red"))	spplot(Clusterdata402Kr.spdf, c("ChemieCl"), col.regions=c("white", "red"))	spplot(Clusterdata402Kr.spdf, c("CommEquipCl"), col.regions=c("white", "red"))
		

spplot(Clusterdata402Kr.spdf, c("AutoCl"), col.regions=c("white", "black"))	spplot(Clusterdata402Kr.spdf, c("ChemieCl"), col.regions=c("white", "black"))	spplot(Clusterdata402Kr.spdf, c("CommEquipCl"), col.regions=c("white", "black"))
		

- **Contiguity (neighbourhood) matrix and weights list object (row-standardized weights)**

```
> W01Kr402.df = read.csv("W01Kr402R.CSV", header=TRUE)
or
> W01Kr402.df = read.csv("W01Kr402R.CSV")

> class(W01Kr402.df)
[1] "data.frame"
> str(W01Kr402.df)
'data.frame':   402 obs. of  402 variables:
 $ X1001 : int  0 0 0 0 0 0 0 0 0 0 ...
 $ X1002 : int  0 0 0 0 0 0 0 0 0 1 ...
 $ X1003 : int  0 0 0 0 0 1 0 1 0 0 ...
 $ X1004 : int  0 0 0 0 0 0 0 0 0 1 ...
 $ X1051 : int  0 0 0 0 0 0 1 0 0 0 ...
 $ X1053 : int  0 0 1 0 0 0 0 0 0 0 ...

> dim(W01Kr402.df)
[1] 402 402

> W01Kr402.df[1:6,1:6]
  X1001 X1002 X1003 X1004 X1051 X1053
1     0     0     0     0     0     0
2     0     0     0     0     0     0
3     0     0     0     0     0     1
4     0     0     0     0     0     0
5     0     0     0     0     0     0
6     0     0     1     0     0     0
```

```
> W01Kr402.mat = as.matrix(W01Kr402.df, nrow=402, ncol=402)
or
> W01Kr402.mat = as.matrix(W01Kr402.df)

> class(W01Kr402.mat)
[1] "matrix"

> str(W01Kr402.mat)
 int [1:402, 1:402] 0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:402] "X1001" "X1002" "X1003" "X1004" ...

> dim(W01Kr402.mat)
[1] 402 402
> W01Kr402.mat[1:6,1:6]
     X1001 X1002 X1003 X1004 X1051 X1053
[1,]     0     0     0     0     0     0
[2,]     0     0     0     0     0     0
[3,]     0     0     0     0     0     1
[4,]     0     0     0     0     0     0
[5,]     0     0     0     0     0     0
[6,]     0     0     1     0     0     0

> rownames(W01Kr402.mat)
NULL
> colnames(W01Kr402.mat)
  [1] "X1001"  "X1002"  "X1003"  "X1004"  "X1051"  "X1053"  "X1054"  "X1055"
...
  [401] "X16076" "X16077"

> KREISID = Clusterdata402Kr.spdf$KREISNR
> class(KREISID)
[1] "integer"
```

Check whether KREISID is not sorted or sorted (in increasing order):

```
> is.unsorted(KREISID)
[1] FALSE
or
> is.sorted = Negate(is.unsorted)
> is.sorted(KREISID)
[1] TRUE

> KREISID = as.character(KREISID)
> class(KREISID)
[1] "character"

> KREISID
  [1] "01001" "01002" "01003" "01004" "01051" "01053" "01054" "01055" "01056"
…
  [397] "16072" "16073" "16074" "16075" "16076" "16077"
```

```
> rownames(W01Kr402.mat) = KREISNR
> rownames(W01Kr402.mat)
  [1] "01001" "01002" "01003" "01004" "01051" "01053" "01054" "01055" "01056"
...
  [397] "16072" "16073" "16074" "16075" "16076" "16077"

> colnames(W01Kr402.mat) = KREISID
> colnames(W01Kr402.mat)
  [1] "01001" "01002" "01003" "01004" "01051" "01053" "01054" "01055" "01056"
…
  [397] "16072" "16073" "16074" "16075" "16076" "16077"

> W01Kr402.mat[1:6,1:6]
        01001 01002 01003 01004 01051 01053
01001     0     0     0     0     0     0
01002     0     0     0     0     0     0
01003     0     0     0     0     0     1
01004     0     0     0     0     0     0
01051     0     0     0     0     0     0
01053     0     0     1     0     0     0

> isSymmetric(W01Kr402.mat)
[1] TRUE
```

Convert a square spatial weights matrix to a weights list object:

```
> library(spdep)

> W402Kr.lw = mat2listw(W01Kr402.mat, row.names = KREISID, style="W")

> class(W402Kr.lw)
[1] "listw" "nb"

> summary(W402Kr.lw)
Characteristics of weights list object:
Neighbour list object:
Number of regions: 402
Number of nonzero links: 2090
Percentage nonzero weights: 1.293285
Average number of links: 5.199005
Link number distribution:
 1  2  3  4  5  6  7  8  9 10 11
27 28 44 45 64 73 67 30 15  6  3
27 least connected regions:
01001 03405 07211 07317 08121 08211 09163 09172 09261 09262 09263 09361
09362 09363 09461 09462 09463 09464 09561 09662 09663 09762 09763 12052
13003 15002 16055 with 1 link
3 most connected regions:
05158 07338 09472 with 11 links
Weights style: W
Weights constants summary:
```

```
       n         nn   S0            S1           S2
W 402 161604 402 182.8518 1728.306

> str(W402Kr.lw)
List of 3
 $ style     : chr "W"
 $ neighbours:List of 402
  ..$ : Named int 12
  .. ..- attr(*, "names")= chr "01059"
  ..$ : Named int [1:2] 10 11
  .. ..- attr(*, "names")= chr [1:2] "01057" "01058"
..
..$ : Named int [1:3] 100 101 103
  .. ..- attr(*, "names")= chr [1:3] "05754" "05758" "05766"
  .. [list output truncated]
  ..- attr(*, "class")= chr "nb"
  ..- attr(*, "region.id")= chr [1:402] "01001" "01002" "01003" "01004" ...
  ..- attr(*, "call")= logi NA
  ..- attr(*, "sym")= logi TRUE
 $ weights   :List of 402
  ..$ : num 1
  ..$ : num [1:2] 0.5 0.5
  ..$ : num [1:4] 0.25 0.25 0.25 0.25
…
..$ : num [1:3] 0.333 0.333 0.333
  .. [list output truncated]
  ..- attr(*, "mode")= chr "general"
  ..- attr(*, "glist")= chr [1:127] "list(1, c(1, 1), c(1, 1, 1, 1), c(1, 1, 1), c(1, 1, 1, 1), c(1, "
"1, 1, 1, 1, 1, 1), c(1, 1), c(1, 1, 1, 1), c(1, 1, 1, 1), c(1, " "1, 1, 1, 1), c(1, 1, 1, 1, 1, 1,
1), c(1, 1, 1, 1), c(1, 1, 1, " "1, 1, 1, 1, 1), c(1, 1, 1, 1, 1), c(1, 1, 1, 1, 1), c(1, 1, 1, " ...
  ..- attr(*, "glistsym")= atomic [1:1] TRUE
  .. ..- attr(*, "d")= num 0
  ..- attr(*, "W")= logi TRUE
  ..- attr(*, "comp")=List of 1
  .. ..$ d: num [1:402] 1 2 4 3 4 7 2 4 4 5 ...
 - attr(*, "class")= chr [1:2] "listw" "nb"
 - attr(*, "region.id")= chr [1:402] "01001" "01002" "01003" "01004" ...
 - attr(*, "call")= language nb2listw(neighbours = res$neighbours, glist = res$weights,
style = style,      zero.policy = TRUE)
```

- **Supplementary exercise: Creating a row-standardized weights matrix object from a contiguity (neighbourhood) matrix**

```
> W01Kr402.mat[1:6,1:6]
      01001 01002 01003 01004 01051 01053
01001     0     0     0     0     0     0
01002     0     0     0     0     0     0
01003     0     0     0     0     0     1
01004     0     0     0     0     0     0
01051     0     0     0     0     0     0
01053     0     0     1     0     0     0
```

```
> class(W01Kr402.mat)
[1] "matrix"

> dim(W01Kr402.mat)
[1] 402 402

> W01Kr402.sums.rows = apply(W01Kr402.mat, 1, sum)
> W01Kr402.sums.rows
01001 01002 01003 01004 01051 01053 01054 01055 01056 01057 01058 01059
    1     2     4     3     4     7     2     4     4     5     7     4
…
16066 16067 16068 16069 16070 16071 16072 16073 16074 16075 16076 16077
    7     6     6     8     7     8     4     7     7     6     7     5

> length(W01Kr402.sums.rows)
[1] 402
> W01Kr402.sums.cols = apply(W01Kr402.mat, 2, sum)
> W01Kr402.sums.cols
01001 01002 01003 01004 01051 01053 01054 01055 01056 01057 01058 01059
    1     2     4     3     4     7     2     4     4     5     7     4
…
16066 16067 16068 16069 16070 16071 16072 16073 16074 16075 16076 16077
    7     6     6     8     7     8     4     7     7     6

> length(W01Kr402.sums.cols)
[1] 402
> W01Kr402.sums.rows == W01Kr402.sums.cols
01001 01002 01003 01004 01051 01053 01054 01055 01056 01057 01058
 TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
…
16066 16067 16068 16069 16070 16071 16072 16073 16074 16075 16076
 TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE

> WKr402.mat = W01Kr402.mat/W01Kr402.sums.rows

> class(WKr402.mat)
[1] "matrix"

> dim(WKr402.mat)
[1] 402 402

> WKr402.mat[1:8,1:8]
      01001 01002  01003     01004 01051 01053 01054 01055
01001     0     0  0.0000000     0   0.0   0.00  0.00  0.00
01002     0     0  0.0000000     0   0.0   0.0   0.00  0.00
01003     0     0  0.0000000     0   0.0   0.25  0.00  0.25
01004     0     0  0.0000000     0   0.0   0.00  0.00  0.00
01051     0     0  0.0000000     0   0.0   0.00  0.25  0.00
01053     0     0  0.1428571     0   0.0   0.00  0.00  0.00
01054     0     0  0.0000000     0   0.5   0.00  0.00  0.00
01055     0     0  0.2500000     0   0.0   0.00  0.00  0.00
```

All row sums are equal to 1:

```
> WKr402.sums.rows = apply(WKr402.mat, 1, sum)
> WKr402.sums.rows
01001 01002 01003 01004 01051 01053 01054 01055 01056 01057 01058 01059
    1     1     1     1     1     1     1     1     1     1     1
```

7.2 Application 2: Existence of a wage curve in East Germany

The wage curve introduced by Blanchflower and Oswald (1990, 1994) postulates a negative correlation between wages and unemployment. Empirical studies focus on particular theoretical channels establishing the relationship. Econometric panel models mostly draw on unionized bargaining or the efficiency wage hypothesis.

The efficiency wage hypothesis states that employers offer wage premiums to promote workers' efforts and avoid shirking. In view of high costs of monitoring workers' productivity, this strategy does not run contrary to profit maximization. The higher the unemployment rate, the more difficult workers will find a new job in case of a dismissal. Against this backdrop, firms can afford to offer lower wage premiums in slack labour markets. Therefore, a negative link between regional unemployment and wage should be expected.

Mostly wage curve analysis treats local labour markets as isolated economies in spite of the presence of commuter flows between the place of residence and job location. If working conditions in a neighbouring region are favourable relative to costs of commuting, workplaces outside the home region are attractive for employees. This demands a spatial econometric approach that can be rationalized by monopsonistic competition.

Here data is provided for examining whether a long-run East German wage curve exists and which role spatial effects play in this context. The wage curve is typically estimated as a Mincer-type earnings function with an enriched set of control variables for individual or regional characteristics. The inclusion of spatial spillovers in the wage curve is grounded in the theory of monopsonistic competition.

Formal representation of the wage curve:

• FIxed effects (FE) panel model :

$$(18)\quad \log(w_{rt}) = \alpha_{r\bullet} + \alpha_{\bullet t} + \beta \cdot \log(u_{rt}) + \sum_{j=1}^{m} \delta_j \cdot X_{jrt} + \varepsilon_{rt}$$

• Spatial cross-regressive panel model (SLX) with fixed effects (FE):

$$(19)\quad \log(w_{rt}) = \alpha_{r\bullet} + \alpha_{\bullet t} + \beta \cdot \log(u_{rt}) + \beta^* \cdot SL(\log(u_{st}))$$

$$+ \sum_{j=1}^{m} \delta_j \cdot X_{jrt} + \sum_{j=1}^{m} \delta_j^* \cdot SL(X_{jrt}) + \varepsilon_{rt}\,.$$

• Spatial Durbin panel model (SDPM) with fixed effects (FE):

$$(20)\quad \log(w_{rt}) = \alpha_{r\bullet} + \alpha_{\bullet t} + \lambda \cdot SL(\log(w_{rt})) + \beta \cdot \log(u_{rt}) + \beta^{*} \cdot SL(\log(u_{st})) + \sum_{j=1}^{m} \delta_j \cdot X_{jrt} + \sum_{j=1}^{m} \delta_j^{*} \cdot SL(X_{jrt}) + \varepsilon_{rt}.$$

The estimation of cross-sectional regression models of the wage curve for each year of the period of investigation may provide information on the stability of the wage curve elasticity. This approach neglects, however, unobservable regional characteristics as well as national trends.

Table 1: Regional characteristics (control variables)

Type of control	Variable	Indicator	Relevance
• Experience	Share of younger workers (agejung)	Share of employees younger than 30 years	Less work experience; cognitive abilities (speed and memory)
	Share of elder workers (ageold)	Share of employees of 50 years or older	More work experience; less physical strength; cognitive abilities (vocabulary size, verbal abilities)
• Vocational education	Share of high qualified workers (svb_hq)	Share of emploees with an acad. degree	Benefits of higher education („education pays“)
	Share of low qualified workers (svb_oq)	Share of employees without vocational training qualification	Low-productivity workers
• Gender	Share of female workers (svb_fq)	Share of female workers	Higher labour supply elasticity; less employed in sectors with high entry/exit costs
• Working time	Share of part-time workers (svb_tq)	Share of part-time workers	Less bargaining power (high share in unfavourable business conditions)
• Firm size	Firm size (dbetr) Squared firm size (dbetr2)	Average number of employees per firm in full-time equivalents	Small firms have less monitoring costs and offer different employment opportunities than large firms
• Sectoral composition	Service sector (svb_dq)	Share of employees in service sector	Sectoral wage differentials

	Industrial sector (WZCq)	Share of employees in industrial sector	
	Agricultural sector (WZAq)	Share of em- ployees in agri- culture sector	Low-wage sector
• Centres and accessibiilty	Urbanity (Agg- lomeration) (dichte)	Share of agricul- ture in own re- gion and sur- roundings	Higher wages in urba- nized regions due to agglomeration advan- tages

einw: Population, qkm: Area, erw: Employed persons,

svb: Employees subject to social security contributions

A. Regional disaggregation: 77 NUTS-3 regions (urban and rural districts)

- **Reading CSV data file in a data.frame object**

The CSV file WageCurveDataO.CSV contains panel data for estimating an East German wage curve for 77 East German NUTS-3 regions for the period 1995-2010 Hint: A delineation of 33 regional labour markets (RAM) is also available.[3]

Ordinary (traditional) panel data structure:
Panel data are first ordered by cross-section and then by time period
→ Time series for each region are stacked

Reading CSV file WageCurveDataO.CSV with information on hourly wages (w), unemployment rates (alq) and control variables in a data.frame object:

```
> WageCurveDataO.df = read.csv("WageCurveDataO.CSV ", header = TRUE)

> class(WageCurveDataO.df)
[1] "data.frame"

> dim(WageCurveDataO.df)
[1] 1232   21

> str(WageCurveDataO.df)
'data.frame':   1232 obs. of  21 variables:
 $ krnr   : int  11000 11000 11000 11000 11000 11000 11000 11000 11000 11000 ...
 $ jahr   : int  1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 ...
 $ name   : Factor w/ 76 levels "Altenburger Land",..: 6 6 6 6 6 6 6 6 6 6 ...
 $ w      : num  74.4 67.7 76.2 77.2 79 ...
 $ einw   : num  3471418 3458763 3425759 3398822 3386667 ...
 $ qkm    : num  892 892 892 892 892 ...
```

[3] Kosfeld and Werner (2012), Deutsche Arbeitsmarktregionen – Neuabgrenzung nach den Kreisgebietsreformen 2007–2011, Raumforschung und Raumordnung 70, 49-64.

```
 $ dichte : num  3893 3879 3842 3812 3798 ...
 $ alq    : num  0.128 0.138 0.157 0.16 0.159 0.157 0.161 0.169 0.181 0.176 ...
 $ erw    : num  1612719 1581935 1551262 1541142 1541170 ...
 $ svb    : int  1254004 1210386 1158925 1132570 1131645 1139096 1125714
1103776 1065424 1042262 ...
 $ svb_fq : num  0.459 0.459 0.459 0.465 0.494 0.498 0.503 0.509 0.513 0.514 ...
 $ svb_tq : num  0.131 0.138 0.147 0.159 0.153 0.163 0.169 0.176 0.182 0.186 ...
 $ svb_oq : num  0.145 0.147 0.149 0.152 0.151 0.15 0.148 0.143 0.139 0.134 ...
 $ svb_hq : num  0.125 0.122 0.12 0.118 0.12 0.122 0.123 0.127 0.129 0.131 ...
 $ svb_dq : num  0.719 0.727 0.734 0.748 0.763 0.778 0.791 0.801 0.81 0.816 ...
 $ dbetr  : num  15.6 15 14.5 14.1 13.5 ...
 $ dbetr2 : num  242 225 209 199 182 ...
 $ WZAq   : num  0.0062 0.0062 0.006 0.0059 0.0054 0.0053 0.0052 0.0049 0.0047
0.0045 ...
 $ WZCq   : num  0.1091 0.1072 0.1043 0.0955 0.1035 ...
 $ agejung: num  0.231 0.219 0.215 0.213 0.215 ...
 $ ageold : num  0.242 0.235 0.231 0.223 0.213 ...

> WageCurveDataO.df[1:3,]
   krnr jahr   name      w    einw    qkm   dichte   alq     erw     svb svb_fq
1 11000 1995 Berlin 74.421 3471418 891.68 3893.121 0.128 1612719 1254004
0.459
2 11000 1996 Berlin 67.690 3458763 891.68 3878.929 0.138 1581935 1210386
0.459
3 11000 1997 Berlin 76.210 3425759 891.68 3841.915 0.157 1551262 1158925
0.459
  svb_tq svb_oq svb_hq svb_dq  dbetr  dbetr2   WZAq   WZCq agejung ageold
1  0.131  0.145  0.125  0.719 15.556 241.989 0.0062 0.1091  0.2311 0.2418
2  0.138  0.147  0.122  0.727 14.984 224.520 0.0062 0.1072  0.2186 0.2355
3  0.147  0.149  0.120  0.734 14.456 208.976 0.0060 0.1043  0.2148 0.2305

> WageCurveDataO.df[1230:1232,]
      krnr jahr             name      w   einw    qkm  dichte   alq   erw   svb
1230 16077 2008 Altenburger Land 60.546 101705 569.09 178.715 0.160 37300
26809
1231 16077 2009 Altenburger Land 61.778 100215 569.09 176.097 0.158 37000
26352
1232 16077 2010 Altenburger Land 62.308  98810 569.09 173.628 0.136 36900
26479
     svb_fq svb_tq svb_oq svb_hq svb_dq  dbetr  dbetr2   WZAq   WZCq agejung
1230  0.472  0.193  0.072  0.065  0.596 10.694 114.362 0.0266 0.1104  0.2109
1231  0.481  0.202  0.074  0.066  0.606 10.743 115.412 0.0273 0.1101  0.2080
1232  0.483  0.198  0.071  0.065  0.607 10.707 114.640 0.0265 0.1086  0.2049
     ageold
1230 0.2928
1231 0.3093
1232 0.3229
```

Storing the share of industrial workers WZCq to a CSV file:

```
> write.csv(WageCurveDataO.df$WZCq, file="WZCq.CSV", row.names = FALSE)
```

Spatial panel data structure:
Panel data are first ordered by time period and then by region
→ Cross-sectional data for each time period are stacked

Transforming the traditional panel data structure to a spatial panel data structure:

```
> WageCurveDataO.dfo = WageCurveDataO.df[order(WageCurveDataO.df$jahr) , ]

> class(WageCurveDataO.dfo)
[1] "data.frame"

> dim(WageCurveDataO.dfo)
[1] 1232   21

> str(WageCurveDataO.dfo)
'data.frame':   1232 obs. of  21 variables:
 $ krnr   : int  11000 12051 12052 12053 12054 12060 12061 12062 12063 12064 ...
 $ jahr   : int  1995 1995 1995 1995 1995 1995 1995 1995 1995 1995 ...
 $ name   : Factor w/ 76 levels "Altenburger Land",..: 6 8 11 20 49 4 12 17 27 38 ...
 $ w      : num  74.4 56 60.2 61 66 ...
 $ einw   : num  3471418 85994 123214 80807 136619 ...
 $ qkm    : num  892 230 165 148 188 ...
 $ dichte : num  3893 374 746 547 726 ...
 $ alq    : num  0.128 0.094 0.067 0.131 0.038 0.131 0.102 0.151 0.138 0.138 ...
 $ erw    : num  1612719 39921 72307 47196 87920 ...
 $ svb    : int  1254004 33276 61941 39576 80054 51027 48813 44114 39109 55294
...
 $ svb_fq : num  0.459 0.466 0.528 0.508 0.53 0.457 0.461 0.421 0.461 0.434 ...
 $ svb_tq : num  0.131 0.124 0.133 0.124 0.104 0.114 0.124 0.123 0.122 0.127 ...
 $ svb_oq : num  0.145 0.088 0.103 0.122 0.103 0.103 0.109 0.09 0.115 0.101 ...
 $ svb_hq : num  0.125 0.094 0.153 0.127 0.174 0.086 0.087 0.071 0.079 0.084 ...
 $ svb_dq : num  0.719 0.658 0.756 0.787 0.806 0.634 0.596 0.505 0.554 0.583 ...
 $ dbetr  : num  15.6 18.1 20.9 20.4 22 ...
 $ dbetr2 : num  242 328 438 418 483 ...
 $ WZAq   : num  0.0062 0.0077 0.009 0.0111 0.0046 0.0357 0.053 0.0663 0.0557
0.0927 ...
 $ WZCq   : num  0.109 0.149 0.15 0.125 0.133 ...
 $ agejung: num  0.231 0.23 0.259 0.253 0.22 ...
 $ ageold : num  0.242 0.24 0.202 0.191 0.235 ...

> WageCurveDataO.dfo[1:3,]
    krnr jahr                    name      w    einw    qkm   dichte   alq
1  11000 1995                  Berlin 74.421 3471418 891.68 3893.121 0.128
17 12051 1995 Brandenburg an der Havel 56.029   85994 229.71  374.359 0.094
33 12052 1995                 Cottbus 60.179  123214 165.15  746.073 0.067
       erw     svb svb_fq svb_tq svb_oq svb_hq svb_dq  dbetr  dbetr2   WZAq
1  1612719 1254004  0.459  0.131  0.145  0.125  0.719 15.556 241.989 0.0062
17   39921   33276  0.466  0.124  0.088  0.094  0.658 18.104 327.755 0.0077
33   72307   61941  0.528  0.133  0.103  0.153  0.756 20.926 437.897 0.0090
     WZCq agejung ageold
1  0.1091  0.2311 0.2418
```

```
17 0.1491  0.2299 0.2398
33 0.1503  0.2589 0.2021

> WageCurveDataO.dfo[1230:1232,]
     krnr jahr            name      w  einw     qkm  dichte   alq   erw
1200 16075 2010 Saale-Orla-Kreis 62.845  87799 1148.39  76.454 0.086 40300
1216 16076 2010            Greiz 63.459 107555  843.54 127.504 0.100 38700
1232 16077 2010 Altenburger Land 62.308  98810  569.09 173.628 0.136 36900
       svb svb_fq svb_tq svb_oq svb_hq svb_dq  dbetr  dbetr2   WZAq   WZCq
1200 29449  0.450  0.181  0.083  0.066  0.472 11.481 131.813 0.0509 0.1024
1216 28005  0.463  0.177  0.059  0.072  0.544  9.110  82.992 0.0351 0.1457
1232 26479  0.483  0.198  0.071  0.065  0.607 10.707 114.640 0.0265 0.1086
     agejung ageold
1200  0.2015 0.3413
1216  0.1911 0.3413
1232  0.2049 0.3229
```

Data.frame objects can be used for estimating panel models. Spatial panel models are only correctly estimated when cross-sectional data is stacked for each time period. This is why the new ordered data.frame object WageCurveDataO.dfo is formed.

Lags and differences need special objects called pdata.frames. Time lags can be created within the general infrastructure of the R package plm for traditional panel data structures that are first ordered by cross-section and then by time period:

```
> WageCurveDataO.pdf = pdata.frame(WageCurveDataO.df, c("krnr", "jahr"))

> class(WageCurveDataO.pdf)
[1] "pdata.frame" "data.frame"
> class(WageCurveDataO.pdf$krnr)
> class(WageCurveDataO.pdf$krnr)
[1] "pseries" "factor"
> class(WageCurveDataO.pdf$jahr)
[1] "pseries" "factor"
> class(WageCurveDataO.pdf$w)
[1] "pseries" "numeric"
> class(WageCurveDataO.pdf$name)
[1] "pseries" "factor"
> WageCurveDataO.pdf$name = as.character(WageCurveDataO.pdf$name)
> class(WageCurveDataO.pdf$name)
[1] "pseries"   "character"

> dim(WageCurveDataO.pdf)
[1] 1232   21

> WageCurveDataO.pdf[1:3,]
            krnr jahr   name     w    einw    qkm   dichte   alq     erw
11000-1995 11000 1995 Berlin 74.421 3471418 891.68 3893.121 0.128 1612719
11000-1996 11000 1996 Berlin 67.690 3458763 891.68 3878.929 0.138 1581935
11000-1997 11000 1997 Berlin 76.210 3425759 891.68 3841.915 0.157 1551262
```

```
           svb svb_fq svb_tq svb_oq svb_hq svb_dq  dbetr  dbetr2   WZAq
11000-1995 1254004  0.459  0.131  0.145  0.125  0.719 15.556 241.989 0.0062
11000-1996 1210386  0.459  0.138  0.147  0.122  0.727 14.984 224.520 0.0062
11000-1997 1158925  0.459  0.147  0.149  0.120  0.734 14.456 208.976 0.0060
          WZCq agejung ageold
11000-1995 0.1091  0.2311 0.2418
11000-1996 0.1072  0.2186 0.2355
11000-1997 0.1043  0.2148 0.2305

> WageCurveDataO.pdf[1230:1232,]
         krnr jahr           name      w   einw    qkm  dichte   alq   erw
16077-2008  16077  2008  Altenburger  Land  60.546  101705  569.09  178.715  0.160
37300
16077-2009  16077  2009  Altenburger  Land  61.778  100215  569.09  176.097  0.158
37000
16077-2010  16077  2010  Altenburger  Land  62.308   98810  569.09  173.628  0.136
36900
          svb svb_fq svb_tq svb_oq svb_hq svb_dq  dbetr  dbetr2   WZAq
16077-2008 26809  0.472  0.193  0.072  0.065  0.596 10.694 114.362 0.0266
16077-2009 26352  0.481  0.202  0.074  0.066  0.606 10.743 115.412 0.0273
16077-2010 26479  0.483  0.198  0.071  0.065  0.607 10.707 114.640 0.0265
          WZCq agejung ageold
16077-2008 0.1104  0.2109 0.2928
16077-2009 0.1101  0.2080 0.3093
16077-2010 0.1086  0.2049 0.3229
```

Spatial lags of variables belonging to a panel data set can be formed using the slag function of the R package splm. For this, the pdata.frame has to be first ordered by time period and then by region.

```
> WageCurveDataO1.pdfo =
WageCurveDataO.pdf[order(WageCurveDataO.pdf$jahr) , ]

> class(WageCurveDataO.pdfo)
[1] "pdata.frame" "data.frame"
> class(WageCurveDataO.pdfo$krnr)
[1] "pseries" "pseries" "factor"
> class(WageCurveDataO.pdfo$jahr)
[1] "pseries" "pseries" "factor"
> class(WageCurveDataO.pdfo$w)
[1] "pseries" "numeric"
> class(WageCurveDataO.pdfo$name)
[1] "pseries"   "character"

> dim(WageCurveDataO.pdfo)
[1] 1232   21

> WageCurveDataO.pdfo[1:3,]
         krnr jahr                   name      w    einw    qkm   dichte
1995-11000 11000 1995                 Berlin 74.421 3471418 891.68 3893.121
1995-12051 12051 1995 Brandenburg an der Havel 56.029   85994 229.71  374.359
```

```
1995-12052 12052 1995          Cottbus 60.179  123214 165.15  746.073
              alq     erw     svb svb_fq svb_tq svb_oq svb_hq svb_dq  dbetr
1995-11000 0.128 1612719 1254004  0.459  0.131  0.145  0.125  0.719 15.556
1995-12051 0.094   39921   33276  0.466  0.124  0.088  0.094  0.658 18.104
1995-12052 0.067   72307   61941  0.528  0.133  0.103  0.153  0.756 20.926
            dbetr2   WZAq   WZCq agejung ageold
1995-11000 241.989 0.0062 0.1091  0.2311 0.2418
1995-12051 327.755 0.0077 0.1491  0.2299 0.2398
1995-12052 437.897 0.0090 0.1503  0.2589 0.2021

> WageCurveDataO.pdfo[1230:1232,]
            krnr jahr            name      w   einw     qkm  dichte   alq
2010-16075 16075 2010 Saale-Orla-Kreis 62.845  87799 1148.39  76.454 0.086
2010-16076 16076 2010            Greiz 63.459 107555  843.54 127.504 0.100
2010-16077 16077 2010 Altenburger Land 62.308  98810  569.09 173.628 0.136
              erw   svb svb_fq svb_tq svb_oq svb_hq svb_dq  dbetr  dbetr2   WZAq
2010-16075 40300 29449  0.450  0.181  0.083  0.066  0.472 11.481 131.813 0.0509
2010-16076 38700 28005  0.463  0.177  0.059  0.072  0.544  9.110  82.992 0.0351
2010-16077 36900 26479  0.483  0.198  0.071  0.065  0.607 10.707 114.640 0.0265
             WZCq agejung ageold
2010-16075 0.1024  0.2015 0.3413
2010-16076 0.1457  0.1911 0.3413
2010-16077 0.1086  0.2049 0.3229
```

- **Compiling a matrix of commuting times between East German regional labour markets (RAM)**

The use of a contiguity spatial weights matrix may only insufficiently explain spatial spillovers in the labour markets. For example natural barriers and the lack of commuting infrastructures may weaken the strength spillovers. Here we use commuting times between district towns for establishing a distance-based weights matrix for East German NUTS-3 regions.

Autoroute distances and travel times between 77 East German district towns are computed by means of the distance calculator of www.entfernung.org.

Reading the upper triangular matrix of travel times between districts in a data.frame object:

```
> DistanceMinROst.df = read.csv(file= "DistanceMinROst.CSV", header=TRUE)
> class(DistanceMinROst.df)
[1] "data.frame"
> dim(DistanceMinROst.df)
[1] 77 77
> DistanceMinROst.df[1:5,1:5]
  X11000 X12051 X12052 X12053 X12054
1      0     74     91     77     41
2     NA      0    110     96     49
3     NA     NA      0     76     82
4     NA     NA     NA      0     66
5     NA     NA     NA     NA      0
```

Conversion of an upper triangular matrix into a symmetric matrix:

```
> DistanceMinROst = as.matrix(DistanceMinROst.df)
> class(DistanceMinROst)
[1] "matrix"
> dim(DistanceMinROst)
[1] 77 77

> DistanceMinROst[1:5,1:5]
     X11000 X12051 X12052 X12053 X12054
[1,]      0     74     91     77     41
[2,]     NA      0    110     96     49
[3,]     NA     NA      0     76     82
[4,]     NA     NA     NA      0     66
[5,]     NA     NA     NA     NA      0
> y = which(is.na(DistanceMinROst)==TRUE)
> y[1:10]
 [1]  2  3  4  5  6  7  8  9 10 11
> DistanceMinROst[y] = 0
> DistanceMinROst[1:5,1:5]
     X11000 X12051 X12052 X12053 X12054
[1,]      0     74     91     77     41
[2,]      0      0    110     96     49
[3,]      0      0      0     76     82
[4,]      0      0      0      0     66
[5,]      0      0      0      0      0
> DistanceMinROst = DistanceMinROst + t(DistanceMinROst)
> DistanceMinROst[1:5,1:5]
     X11000 X12051 X12052 X12053 X12054
[1,]      0     74     91     77     41
[2,]     74      0    110     96     49
[3,]     91    110      0     76     82
[4,]     77     96     76      0     66
[5,]     41     49     82     66      0
> rownames(DistanceMinROst)
NULL
> colnames(DistanceMinROst)
 [1] "X11000" "X12051" "X12052" "X12053" "X12054" "X12060" "X12061" "X12062"
...
[73] "X16073" "X16074" "X16075" "X16076" "X16077"
> WageCurveDataO.dfo$krnr[1:77]
 [1] 11000 12051 12052 12053 12054 12060 12061 12062 12063 12064 12065
…
[73] 16073 16074 16075 16076 16077
> class(WageCurveDataO.dfo$krnr[1:77])
[1] "integer"
> KREISNRO = as.character(WageCurveDataO.dfo$krnr[1:77])
> class(KREISNRO)
[1] "character"
> KREISNRO
 [1] "11000" "12051" "12052" "12053" "12054" "12060" "12061" "12062" "12063"
```

```
…
 [73] "16073" "16074" "16075" "16076" "16077"
> rownames(DistanceMinROst) = KREISNRO
> colnames(DistanceMinROst) = KREISNRO

> DistanceMinROst[1:5,1:5]
      11000 12051 12052 12053 12054
11000     0    74    91    77    41
12051    74     0   110    96    49
12052    91   110     0    76    82
12053    77    96    76     0    66
12054    41    49    82    66     0
> isSymmetric(DistanceMinROst)
[1] TRUE
```

- **Distance-based weights matrix (minutes by car with threshold of 90 min.) and weights list object (row-standardized distance-based weights for NUTS-3 regions))**

```
> WdistMinO = 1/DistanceMinROst
> class(WdistMinO)
[1] "matrix"
> dim(WdistMinO)
[1] 77 77
> WdistMinO[1:5,1:5]
           11000       12051       12052      12053      12054
11000        Inf 0.013513514 0.010989011 0.01298701 0.02439024
12051 0.01351351         Inf 0.009090909 0.01041667 0.02040816
12052 0.01098901 0.009090909         Inf 0.01315789 0.01219512
12053 0.01298701 0.010416667 0.013157895        Inf 0.01515152
12054 0.02439024 0.020408163 0.012195122 0.01515152        Inf
> diag(WdistMinO) = 0
> WdistMinO[1:5,1:5]
        11000      12051      12052      12053      12054
11000 0.00000000 0.013513514 0.010989011 0.01298701 0.02439024
12051 0.01351351 0.000000000 0.009090909 0.01041667 0.02040816
12052 0.01098901 0.009090909 0.000000000 0.01315789 0.01219512
12053 0.01298701 0.010416667 0.013157895 0.00000000 0.01515152
12054 0.02439024 0.020408163 0.012195122 0.01515152 0.00000000
> which(is.infinite(WdistMinO))
integer(0)
> 1/90
[1] 0.01111111
> Wdist90MinO = WdistMinO
> Wdist90MinO[WdistMinO<1/90] = 0
> class(Wdist90MinO)
[1] "matrix"
> dim(Wdist90MinO)
[1] 77 77
```

```
> WdistMinO[1:5,1:5]
          11000       12051       12052       12053      12054
11000 0.00000000 0.013513514 0.010989011 0.01298701 0.02439024
12051 0.01351351 0.000000000 0.009090909 0.01041667 0.02040816
12052 0.01098901 0.009090909 0.000000000 0.01315789 0.01219512
12053 0.01298701 0.010416667 0.013157895 0.00000000 0.01515152
12054 0.02439024 0.020408163 0.012195122 0.01515152 0.00000000
> Wdist90MinO[1:5,1:5]
          11000      12051      12052      12053      12054
11000 0.00000000 0.01351351 0.00000000 0.01298701 0.02439024
12051 0.01351351 0.00000000 0.00000000 0.00000000 0.02040816
12052 0.00000000 0.00000000 0.00000000 0.01315789 0.01219512
12053 0.01298701 0.00000000 0.01315789 0.00000000 0.01515152
12054 0.02439024 0.02040816 0.01219512 0.01515152 0.00000000

> Wdist90MinO.lw = mat2listw(Wdist90MinO, style="W")
> class(Wdist90MinO.lw)
[1] "listw" "nb"
> summary(Wdist90MinO.lw)
Characteristics of weights list object:
Neighbour list object:
Number of regions: 77
Number of nonzero links: 1268
Percentage nonzero weights: 21.38641
Average number of links: 16.46753
Link number distribution:
 4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
 1  1  5  4  3  4  4  3  2  5  3  3  3  4  3  4  3  1  2  2  2  2  2  1  2  3
30 32 33
 3  1  1
1 least connected region:
15081 with 4 links
1 most connected region:
15002 with 33 links
Weights style: W
Weights constants summary:
   n    nn  S0       S1       S2
W 77 5929  77 12.91318 314.3047
```

- **Spatial lags of explanatory variables**

If the spatial lag of the endogenous variable wage is needed in estimating a panel model, it is automaticall generated by the R function spml.

Spatial lag of the share of young workers:

```
> lageoldWO90 = slag(WageCurveDataO.pdfo$ageold, listw=Wdist90MinO.lw)
> lageoldWO90[1:7]
[1] 0.2109039 0.2104530 0.2069495 0.2134811 0.2117159 0.2067422 0.2108972
…
```

```
> lageoldWO90[1226:1232]
[1] 0.3109909 0.3249197 0.3164554 0.3123707 0.3134627 0.3162814 0.3111580
```
or
```
> WageCurveDataO.pdfo$lageoldWO90 = slag(WageCurveDataO.pdfo$ageold,
Wdist90MinO.lw)
> WageCurveDataO.pdfo[1:3,]
                krnr  jahr         name            w      einw       qkm     dichte
11000-1995 11000 1995                 Berlin 74.421 3471418   891.68 3893.121
12051-1995 12051 1995 Brandenburg an der Havel 56.029   85994 229.71  374.359
12052-1995 12052 1995                Cottbus 60.179  123214 165.15  746.073
                alq      erw       svb       svb_fq svb_tq svb_oq svb_hq svb_dq  dbetr
11000-1995 0.128 1612719 1254004  0.459   0.131   0.145   0.125    0.719  15.556
12051-1995 0.094   39921       33276  0.466   0.124   0.088   0.094    0.658  18.104
12052-1995 0.067   72307       61941  0.528   0.133   0.103   0.153    0.756  20.926

                dbetr2  WZAq  WZCq  agejung ageold lageoldWO90
11000-1995 241.989 0.0062 0.1091  0.2311  0.2418   0.2109039
12051-1995 327.755 0.0077 0.1491  0.2299  0.2398   0.2104530
12052-1995 437.897 0.0090 0.1503  0.2589  0.2021   0.206949
> dim(WageCurveDataO.pdfo)
[1] 1232   22
```
If desired the column can be dropped from the data.frame by indexing it with a negative sign,
```
> WageCurveDataO.pdfo = WageCurveDataO.pdfo[,-22]
```
or
by defining it as a null object, i.e. ignoring column:
```
> WageCurveDataO.pdfo$lageoldWO90 = NULL
> WageCurveDataO.pdfo[1:3,]
         krnr jahr                    name      w    einw    qkm   dichte
11000-1995 11000 1995                 Berlin 74.421 3471418 891.68 3893.121
12051-1995 12051 1995 Brandenburg an der Havel 56.029   85994 229.71  374.359
12052-1995 12052 1995                Cottbus 60.179  123214 165.15  746.073
          alq     erw     svb svb_fq svb_tq svb_oq svb_hq svb_dq  dbetr
11000-1995 0.128 1612719 1254004  0.459  0.131  0.145  0.125  0.719 15.556
12051-1995 0.094   39921   33276  0.466  0.124  0.088  0.094  0.658 18.104
12052-1995 0.067   72307   61941  0.528  0.133  0.103  0.153  0.756 20.926
         dbetr2   WZAq   WZCq agejung ageold
11000-1995 241.989 0.0062 0.1091  0.2311 0.2418
12051-1995 327.755 0.0077 0.1491  0.2299 0.2398
12052-1995 437.897 0.0090 0.1503  0.2589 0.2021
```

Spatial lag of the log unemployment rate:
```
> llogalqWO90 = slag(log(WageCurveDataO.pdfo$alq), listw = Wdist90MinO.lw)
> llogalqWO90[1:7]
[1] -2.238989 -2.120154 -2.197226 -2.249748 -2.084136 -2.056369 -2.232920
…
> llogalqWO90[1226:1232]
[1] -2.236402 -2.420795 -2.334431 -2.209242 -2.226084 -2.231881 -2.200075
```

- **for sensivity analysis: Contiguity matrix and weights list object (row-standardized weights for NUTS-3 regions)**

```
> WstarO.df = read.csv("WstarRO.csv")
> class(WstarO.df)
[1] "data.frame"
> dim(WstarO.df)
[1] 77 77
> WstarO.df[1:5,1:5]
  X11000 X12051 X12052 X12053 X12054
1      0      0      0      0      1
2      0      0      0      0      0
3      0      0      0      0      0
4      0      0      0      0      0
5      1      0      0      0      0
> WstarO = as.matrix(WstarO.df, nrow=77, ncol=77)
or
> WstarO = as.matrix(WstarO.df)

> WstarO[1:5,1:5]
     X11000 X12051 X12052 X12053 X12054
[1,]      0      0      0      0      1
[2,]      0      0      0      0      0
[3,]      0      0      0      0      0
[4,]      0      0      0      0      0
[5,]      1      0      0      0      0
> KREISNRO
 [1] "11000" "12051" "12052" "12053" "12054" "12060" "12061" "12062"
…
[73] "16073" "16074" "16075" "16076" "16077"
> colnames(WstarO) = KREISNRO
> rownames(WstarO) = KREISNRO
> WstarO[1:5,1:5]
      11000 12051 12052 12053 12054
11000     0     0     0     0     1
12051     0     0     0     0     0
12052     0     0     0     0     0
12053     0     0     0     0     0
12054     1     0     0     0     0
> isSymmetric(WstarO.mat)
[1] TRUE

> WmatO.lw = mat2listw(WstarO, row.names = KreisnrO, style="W")
> class(WmatO.lw)
[1] "listw" "nb"
> summary(WmatO.lw)
Characteristics of weights list object:
Neighbour list object:
Number of regions: 77
Number of nonzero links: 364
Percentage nonzero weights: 6.139315
```

```
Average number of links: 4.727273
Link number distribution:
 1  2  3  4  5  6  7  8  9
 5  9 11 11  9 15  9  6  2
5 least connected regions:
12052 13003 15002 16055 16056 with 1 link
2 most connected regions:
11000 15084 with 9 links
Weights style: W
Weights constants summary:
   n   nn S0       S1       S2
W 77 5929 77 38.28549 330.6279
```

B. Regional disaggregation: 33 regional labour markets (RAM)

- **Reading CSV assignment file in a data.frame object**

```
> ZuordnungRAM141Kreise2012Ost.df =
read.csv("ZuordnungRAM141Kreise2012Ost.CSV", header=TRUE)
> class(ZuordnungRAM141Kreise2012Ost.df)
[1] "data.frame"
> dim(ZuordnungRAM141Kreise2012Ost.df)
[1] 77  4
> str(ZuordnungRAM141Kreise2012Ost.df)
'data.frame':   77 obs. of  4 variables:
 $ KREISNR  : int  11000 12051 12052 12053 12054 12060 12061 12062 12063
12064 ...
 $ name  : Factor w/ 76 levels "Altenburger Land",..: 6 8 11 20 49 4 12 17 27 38 ...
 $ RAM   : int  109 116 118 110 109 109 109 111 112 113 ...
 $ Centre: int  11000 12051 12052 12053 11000 11000 11000 12062 12063 12064 ...
```

The variable CENTRE contains the district towns (= district capitals).

```
> ZuordnungRAM141Kreise2012Ost.df$name =
as.character(ZuordnungRAM141Kreise2012Ost.df$name)
> class(ZuordnungRAM141Kreise2012Ost.df$name)
[1] "character"
> ZuordnungRAM141Kreise2012Ost.df[1:3,]
  KREISNR                    name RAM Centre
1 11000                  Berlin 109  11000
2 12051 Brandenburg an der Havel 116  12051
3 12052                 Cottbus 118  12052
…
> ZuordnungRAM141Kreise2012Ost.df[73:75,]
   KREISNR                name RAM Centre
73 16073  Saalfeld-Rudolstadt 141  16073
74 16074 Saale-Holzland-Kreis 136  16053
75 16075     Saale-Orla-Kreis 141  16073
```

- **Merging NUTS-3 data frame with assignment object**

```
> WageCurveDataO.dfo[1:3,1:3]
    krnr jahr                   name
1  11000 1995                 Berlin
17 12051 1995 Brandenburg an der Havel
33 12052 1995                Cottbus
> class(WageCurveDataO.dfo)
[1] "data.frame"
> dim(WageCurveDataO.dfo)
[1] 1232   21
> class(WageCurveDataO.dfo$krnr)
[1] "integer"
> class(WageCurveDataO.dfo$jahr)
[1] "integer"
> class(WageCurveDataO.dfo$name)
[1] "factor"
> WageCurveDataO.dfo$name = as.character(WageCurveDataO.dfo$name)
> class(WageCurveDataO.dfo$name)
[1] "character"

> KREISNRT16 = rep(ZuordnungRAM141Kreise2012Ost.df$KREISNR, 16)
> class(KREISNRT16)
[1] "integer"
> length(KREISNRT16)
[1] 1155
> KREISNRT16[1:11]
 [1] 11000 12051 12052 12053 12054 12060 12061 12062 12063 12064 12065
> KREISNRT16[70:80]
 [1] 16070 16071 16072 16073 16074 16075 16076 16077 11000 12051 12052
> KREISNRT16[1222:1232]
 [1] 16067 16068 16069 16070 16071 16072 16073 16074 16075 16076 16077
> RAMT16 = rep(ZuordnungRAM141Kreise2012Ost.df$RAM, 16)
> CentreT16 = rep(ZuordnungRAM141Kreise2012Ost.df$Centre, 16)

> WageCurveDataOA.dfo = cbind(WageCurveDataO.dfo, KREISNRT16, RAMT16,
CentreT16)
> dim(WageCurveDataOA.dfo)
[1] 1232   24
> head(WageCurveDataOA.dfo)
    krnr jahr                   name      w    einw     qkm   dichte   alq
1  11000 1995                 Berlin 74.421 3471418  891.68 3893.121 0.128
17 12051 1995 Brandenburg an der Havel 56.029   85994  229.71  374.359 0.094
33 12052 1995                Cottbus 60.179  123214  165.15  746.073 0.067
49 12053 1995        Frankfurt (Oder) 60.965   80807  147.85  546.547 0.131
65 12054 1995                Potsdam 66.017  136619  188.25  725.732 0.038
81 12060 1995                 Barnim 56.746  151783 1479.69  102.578 0.131
       erw     svb svb_fq svb_tq svb_oq svb_hq svb_dq  dbetr  dbetr2   WZAq
1  1612719 1254004  0.459  0.131  0.145  0.125  0.719 15.556 241.989 0.0062
17   39921   33276  0.466  0.124  0.088  0.094  0.658 18.104 327.755 0.0077
```

```
33   72307   61941  0.528  0.133  0.103  0.153  0.756 20.926 437.897 0.0090
49   47196   39576  0.508  0.124  0.122  0.127  0.787 20.442 417.875 0.0111
65   87920   80054  0.530  0.104  0.103  0.174  0.806 21.975 482.901 0.0046
81   62397   51027  0.457  0.114  0.103  0.086  0.634 12.738 162.257 0.0357
     WZCq agejung ageold KREISNRT16 RAMT16 CentreT16
1  0.1091  0.2311 0.2418      11000    109     11000
17 0.1491  0.2299 0.2398      12051    116     12051
33 0.1503  0.2589 0.2021      12052    118     12052
49 0.1253  0.2532 0.1906      12053    110     12053
65 0.1330  0.2202 0.2351      12054    109     11000
81 0.1640  0.2477 0.1951      12060    109     11000
```

```
> WageCurveDataOA.dfo$KREISNRT16 == WageCurveDataOA.dfo$krnr
   [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE TRUE
…
[1219] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE TRUE
```

- **Aggregate NUTS-3 data to RAM level**

The R package plyr contains tools for splitting, combining and aggregating data:

```
> library(plyr)

> wT16O = ddply(WageCurveDataOA.dfo, .(WageCurveDataOA.dfo$jahr,
WageCurveDataOA.dfo$RAM), function(WageCurveDataOA.dfo)
data.frame(wT16O=weighted.mean(WageCurveDataOA.dfo$w,
WageCurveDataOA.dfo$svb)))
> class(wT16O)
[1] "data.frame"
> dim(wT16O)
[1] 528   3
> wT16O[1:4,]
  WageCurveDataOA.dfo$jahr WageCurveDataOA.dfo$RAM    wT16O
1                     1995                     109 72.77998
2                     1995                     110 57.31709
3                     1995                     111 55.44064
4                     1995                     112 55.27200
```

Computation of number of unemployed persons (AL) from unemployment rate (ALQ) and employed persons (ERW):

ALQ = AL/(AL+ERW) ⇔ AL = ALQ·AL + ALQ·ERW ⇔ (1 – ALQ)·AL = ALQ·ERW
⇔ AL = (ALQ·ERW)/(1 – ALQ)

```
> WageCurveDataOA.dfo$al = (WageCurveDataOA.dfo$alq*
WageCurveDataOA.dfo$erw)/(1- WageCurveDataOA.dfo$alq)
> length(WageCurveDataOA.dfo$al)
[1] 1232
```

```
> WageCurveDataOA.dfo$al[1:6]
[1] 236729.394   4141.914   5192.464   7114.702   3472.931   9406.222
> WageCurveDataOA.dfo$al[1227:1232]
[1] 1986.251 5466.667 3282.969 3791.904 4300.000 5808.333

> alqT16O = ddply(WageCurveDataOA.dfo, .(WageCurveDataOA.dfo$jahr,
WageCurveDataOA.dfo$RAM), function(WageCurveDataOA.dfo)
data.frame(alqT16O=weighted.mean(WageCurveDataOA.dfo$alq,
WageCurveDataOA.dfo$erw+ WageCurveDataOA.dfo$al)))
> dim(alqT16O)
[1] 528   3

> alqT16O[1:4,]
  WageCurveDataOA.dfo$jahr WageCurveDataOA.dfo$RAM   alqT16O
1                     1995                     109 0.1233279
2                     1995                     110 0.1370124
3                     1995                     111 0.1650540
4                     1995                     112 0.1380000

> agejungT16O = ddply(WageCurveDataOA.dfo, .( WageCurveDataOA.dfo$jahr,
WageCurveDataOA.dfo$RAM), function(WageCurveDataOA.dfo)
data.frame(agejungT16O=weighted.mean(WageCurveDataOA.dfo$agejung,
WageCurveDataOA.dfo$einw)))
> dim(agejungT16O)
[1] 528   3
> agejungT16O[1:4,]
  WageCurveDataOA.dfo$jahr WageCurveDataOA.dfo$RAM agejungT16O
1                     1995                     109   0.2319171
2                     1995                     110   0.2403437
3                     1995                     111   0.2394377
4                     1995                     112   0.2369000

> ageoldT16O = ddply(WageCurveDataOA.dfo, .( WageCurveDataOA.dfo$jahr,
WageCurveDataOA.dfo$RAM), function(WageCurveDataOA.dfo)
data.frame(ageoldT16O=weighted.mean(WageCurveDataOA.dfo$ageold,
WageCurveDataOA.dfo$einw)))
> dim(ageoldT16O)
[1] 528   3
> ageoldT16O[1:4,]
  WageCurveDataOA.dfo$jahr WageCurveDataOA.dfo$RAM ageoldT16O
1                     1995                     109  0.2384173
2                     1995                     110  0.2002949
3                     1995                     111  0.2207983
4                     1995                     112  0.2199000

> svb_hqT16O = ddply(WageCurveDataOA.dfo, .( WageCurveDataOA.dfo$jahr,
WageCurveDataOA.dfo$RAM), function(WageCurveDataOA.dfo)
data.frame(svb_hqT16O=weighted.mean(WageCurveDataOA.dfo$svb_hq,
WageCurveDataOA.dfo$svb)))
> dim(svb_hqT16O)
[1] 528   3
```

```
> svb_hqT16O[1:4,]
  WageCurveDataOA.dfo$jahr WageCurveDataOA.dfo$RAM svb_hqT16O
1                     1995                     109 0.12505419
2                     1995                     110 0.10840351
3                     1995                     111 0.08312003
4                     1995                     112 0.07900000

> svb_oqT16O = ddply(WageCurveDataOA.dfo, .( WageCurveDataOA.dfo$jahr,
WageCurveDataOA.dfo$RAM), function(WageCurveDataOA.dfo)
data.frame(svb_oqT16O=weighted.mean(WageCurveDataOA.dfo$svb_oq,
WageCurveDataOA.dfo$svb)))

> dim(svb_oqT16O)
[1] 528   3
> svb_oqT16O[1:4,]
  WageCurveDataOA.dfo$jahr WageCurveDataOA.dfo$RAM svb_oqT16O
1                     1995                     109 0.13993502
2                     1995                     110 0.10400340
3                     1995                     111 0.08711428
4                     1995                     112 0.11500000

> svb_fqT16O = ddply(WageCurveDataOA.dfo, .( WageCurveDataOA.dfo$jahr,
WageCurveDataOA.dfo$RAM), function(WageCurveDataOA.dfo)
data.frame(svb_fqT16O=weighted.mean(WageCurveDataOA.dfo$svb_fq,
WageCurveDataOA.dfo$svb)))
> dim(svb_fqT16O)
[1] 528   3
> svb_fqT16O[1:4,]
  WageCurveDataOA.dfo$jahr WageCurveDataOA.dfo$RAM svb_fqT16O
1                     1995                     109  0.4629608
2                     1995                     110  0.4672077
3                     1995                     111  0.4273486
4                     1995                     112  0.4610000

> svb_tqT16O = ddply(WageCurveDataOA.dfo, .( WageCurveDataOA.dfo$jahr,
WageCurveDataOA.dfo$RAM), function(WageCurveDataOA.dfo)
data.frame(svb_tqT16O=weighted.mean(WageCurveDataOA.dfo$svb_tq,
WageCurveDataOA.dfo$svb)))
> dim(svb_tqT16O)
[1] 528   3
> svb_tqT16O[1:4,]
  WageCurveDataOA.dfo$jahr WageCurveDataOA.dfo$RAM svb_tqT16O
1                     1995                     109  0.1286493
2                     1995                     110  0.1090028
3                     1995                     111  0.1258857
4                     1995                     112  0.1220000

> svb_dqT16O = ddply(WageCurveDataOA.dfo, .( WageCurveDataOA.dfo$jahr,
WageCurveDataOA.dfo$RAM), function(WageCurveDataOA.dfo)
data.frame(svb_dqT16O=weighted.mean(WageCurveDataOA.dfo$svb_dq,
WageCurveDataOA.dfo$svb)))
```

```
> dim(svb_dqT16O)
[1] 528   3
> svb_dqT16O[1:4,]
  WageCurveDataOA.dfo$jahr WageCurveDataOA.dfo$RAM svb_dqT16O
1                     1995                     109  0.7166452
2                     1995                     110  0.6400277
3                     1995                     111  0.4386284
4                     1995                     112  0.5540000

> WZAqT16O = ddply(WageCurveDataOA.dfo, .( WageCurveDataOA.dfo$jahr,
WageCurveDataOA.dfo$RAM), function(WageCurveDataOA.dfo)
data.frame(WZAqT16O=weighted.mean(WageCurveDataOA.dfo$WZAq,
WageCurveDataOA.dfo$svb)))
> dim(WZAqT16O)
[1] 528   3
> WZAqT16O[1:4,]
  WageCurveDataOA.dfo$jahr WageCurveDataOA.dfo$RAM    WZAqT15O
1                     1995                     109 0.008753639
2                     1995                     110 0.035395414
3                     1995                     111 0.039751363
4                     1995                     112 0.055700000

> WZCqT16O = ddply(WageCurveDataOA.dfo, .( WageCurveDataOA.dfo$jahr,
WageCurveDataOA.dfo$RAM), function(WageCurveDataOA.dfo)
data.frame(WZCqT16O=weighted.mean(WageCurveDataOA.dfo$WZCq,
WageCurveDataOA.dfo$svb)))
> dim(WZCqT16O)
[1] 528   3
> WZCqT16O[1:4,]
  WageCurveDataOA.dfo$jahr WageCurveDataOA.dfo$RAM  WZCqT15O
1                     1995                     109 0.1142297
2                     1995                     110 0.1652525
3                     1995                     111 0.3288471
4                     1995                     112 0.1715000

> dichteT16O = ddply(WageCurveDataOA.dfo, .(WageCurveDataOA.dfo$jahr,
WageCurveDataOA.dfo$RAM), function(WageCurveDataOA.dfo)
data.frame(dichteT16O=weighted.mean(WageCurveDataOA.dfo$dichte,
WageCurveDataOA.dfo$qkm)))
> dim(dichteT16O)
[1] 528   3
> dichteT16O[1:4,]
  WageCurveDataOA.dfo$jahr WageCurveDataOA.dfo$RAM dichteT16O
1                     1995                     109  807.79717
2                     1995                     110  112.96963
3                     1995                     111   94.03782
4                     1995                     112   76.06100
```

Computation of number of establishments (betr) from employees reliable to the sociclunemployment rate (ALQ) and employed persons (ERW):
betr = svb/dbetr

```
> WageCurveDataOA.dfo$betr =
WageCurveDataOA.dfo$svb/WageCurveDataOA.dfo$dbetr
> class(WageCurveDataOA.dfo$betr)
[1] "numeric"
> length(WageCurveDataOA.dfo$betr)
[1] 1232


> WageCurveDataOA.dfo$betr[1:6]
[1] 80612.240  1838.047  2960.002  1936.014  3642.958  4005.888
> WageCurveDataOA.dfo$betr[1227:1232]
[1] 1712.037 3135.989 2347.072 2565.020 3074.094 2473.055
> WageCurveDataOA.dfo$betr = as.integer(WageCurveDataOA.dfo$betr)
> class(WageCurveDataOA.dfo$betr)
[1] "integer"
> WageCurveDataOA.dfo$betr[1:6]
[1] 80612  1838  2960  1936  3642  4005
> WageCurveDataOA.dfo$betr[1227:1232]
[1] 1712 3135 2347 2565 3074 2473

> dbetrT16O = ddply(WageCurveDataOA.dfo, .(WageCurveDataOA.dfo$jahr,
WageCurveDataOA.dfo$RAM), function(WageCurveDataOA.dfo)
data.frame(dbetrT16O=weighted.mean(WageCurveDataOA.dfo$dbetr,
WageCurveDataOA.dfo$betr)))
> dim(dbetrT16O)
[1] 528   3
> dbetrT16O[1:4,]
  WageCurveDataOA.dfo$jahr WageCurveDataOA.dfo$RAM dbetrT16O
1                     1995                     109  15.53368
2                     1995                     110  14.94571
3                     1995                     111  15.45027
4                     1995                     112  11.42200
```

Merging RAM data.frames:

```
> RAMOjahr.dfo = wT16O
> RAMOjahr.dfo$RAM = wT16O[,2]
> RAMOjahr.dfo$jahr = wT16O[,1]
> RAMOjahr.dfo[1:4,]
  WageCurveDataOA.dfo$jahr WageCurveDataOA.dfo$RAM    wT16O RAM jahr
1                     1995                     109 72.77998 109 1995
2                     1995                     110 57.31709 110 1995
3                     1995                     111 55.44064 111 1995
4                     1995                     112 55.27200 112 1995
> RAMOjahr.dfo = RAMOjahr.dfo[,-c(1:2)]
```

```
> RAMOjahr.dfo[1:4,]
     wT16O RAM jahr
1 72.77998 109 1995
2 57.31709 110 1995
3 55.44064 111 1995
4 55.27200 112 1995
> RAMOjahr.dfo = RAMOjahr.dfo[,-1]
> RAMOjahr.dfo[1:4,]
  RAM jahr
1 109 1995
2 110 1995
3 111 1995
4 112 1995
> WageCurveT15RAM.dfo = RAMOjahr.dfo
> WageCurveT15RAM.dfo$w = wT16O[,3]
> WageCurveT15RAM.dfo$lnw = log(wT16O[,3])
> WageCurveT15RAM.dfo$alq = alqT16O[,3]
> WageCurveT15RAM.dfo$lnalq = log(alqT16O[,3])
> WageCurveT15RAM.dfo$young = agejungT16O[,3]
> WageCurveT15RAM.dfo$old = ageoldT16O[,3]
> WageCurveT15RAM.dfo$svb_hq = svb_hqT16O[,3]
> WageCurveT15RAM.dfo$svb_oq = svb_oqT16O[,3]
> WageCurveT15RAM.dfo$svb_fq = svb_fqT16O[,3]
> WageCurveT15RAM.dfo$svb_tq = svb_tqT16O[,3]
> WageCurveT15RAM.dfo$svb_dq = svb_dqT16O[,3]
> WageCurveT15RAM.dfo$WZAq = WZAqT16O[,3]
> WageCurveT15RAM.dfo$WZCq = WZCqT16O[,3]
> WageCurveT15RAM.dfo$dichte = dichteT16O[,3]
> WageCurveT15RAM.dfo$dbetr = dbetrT16O[,3]
> WageCurveT15RAM.dfo$dbetr2 = dbetrT16O[,3]^2
> class(WageCurveT15RAM.dfo)
[1] "data.frame"
> dim(WageCurveT15RAM.dfo)
[1] 495  18
> str(WageCurveT15RAM.dfo)
'data.frame':   528 obs. of  18 variables:
 $ RAM   : int  109 110 111 112 113 114 115 116 117 118 ...
 $ jahr  : int  1995 1995 1995 1995 1995 1995 1995 1995 1995 1995 ...
 $ w     : num  72.8 57.3 55.4 55.3 56.4 ...
 $ lnw   : num  4.29 4.05 4.02 4.01 4.03 ...
 $ alq   : num  0.123 0.137 0.165 0.138 0.138 ...
 $ lnalq : num  -2.09 -1.99 -1.8 -1.98 -1.98 ...
 $ young : num  0.232 0.24 0.239 0.237 0.226 ...
 $ old   : num  0.238 0.2 0.221 0.22 0.221 ...
 $ svb_hq: num  0.1251 0.1084 0.0831 0.079 0.084 ...
 $ svb_oq: num  0.1399 0.104 0.0871 0.115 0.101 ...
 $ svb_fq: num  0.463 0.467 0.427 0.461 0.434 ...
 $ svb_tq: num  0.129 0.109 0.126 0.122 0.127 ...
 $ svb_dq: num  0.717 0.64 0.439 0.554 0.583 ...
 $ WZAq  : num  0.00875 0.0354 0.03975 0.0557 0.0927 ...
 $ WZCq  : num  0.114 0.165 0.329 0.172 0.186 ...
```

```
 $ dichte: num  807.8 113 94 76.1 79.9 ...
 $ dbetr : num  15.5 14.9 15.5 11.4 12 ...
 $ dbetr2: num  241 223 239 130 144 ...
> WageCurveT15RAM.dfo[1:4,]
  RAM jahr        w      lnw       alq     lnalq     young       old     svb_hq
1 109 1995 72.77998 4.287441 0.1233279 -2.092909 0.2319171 0.2384173
0.12505419
2 110 1995 57.31709 4.048599 0.1370124 -1.987684 0.2403437 0.2002949
0.10840351
3 111 1995 55.44064 4.015313 0.1650540 -1.801482 0.2394377 0.2207983
0.08312003
4 112 1995 55.27200 4.012266 0.1380000 -1.980502 0.2369000 0.2199000
0.07900000
      svb_oq    svb_fq    svb_tq    svb_dq        WZAq       WZCq    dichte
1 0.13993502 0.4629608 0.1286493 0.7166452 0.008753639 0.1142297 807.79717
2 0.10400340 0.4672077 0.1090028 0.6400277 0.035395414 0.1652525 112.96963
3 0.08711428 0.4273486 0.1258857 0.4386284 0.039751363 0.3288471  94.03782
4 0.11500000 0.4610000 0.1220000 0.5540000 0.055700000 0.1715000  76.06100
     dbetr   dbetr2
1 15.53368 241.2952
2 14.94571 223.3742
3 15.45027 238.7110
4 11.42200 130.4621
```

- **Compiling a matrix of commuting times between East German labour market regions**

We use commuting times between RAM centres for establishing a distance-based weights matrix for East German regional labour markets. Specifically, travel times between 33 East German RAM centres are computed by means of the distance calculator of www.entfernung.org.

Reading the upper triangular matrix of travel times between RAMs in a data.frame object:

```
> DistanceRAMMinOst.df = read.csv(file= "RAMOstTimeR.CSV", header=TRUE)
> class(DistanceRAMMinOst.df)
[1] "data.frame"
> dim(DistanceRAMMinOst.df)
[1] 33 33
> DistanceRAMMinOst.df[1:5,1:5]
  X109 X110 X111 X112 X113
1    0   77  100   94   79
2   NA    0  108  124   28
3   NA   NA    0  129  126
4   NA   NA   NA    0  131
5   NA   NA   NA   NA    0
```

Conversion of an upper triangular matrix into a symmetric matrix:

```
> DistanceRAMMinOst = as.matrix(DistanceRAMMinOst.df)
> class(DistanceRAMMinOst)
[1] "matrix"
> dim(DistanceRAMMinOst)
[1] 33 33
> DistanceRAMMinOst[1:5,1:5]
     X109 X110 X111 X112 X113
[1,]    0   77  100   94   79
[2,]   NA    0  108  124   28
[3,]   NA   NA    0  129  126
[4,]   NA   NA   NA    0  131
[5,]   NA   NA   NA   NA    0

> yRAM = which(is.na(DistanceRAMMinOst)==TRUE)
> yRAM[1:10]
 [1]  2  3  4  5  6  7  8  9 10 11
> DistanceRAMMinOst[yRAM] = 0
> DistanceRAMMinOst[1:5,1:5]
     X109 X110 X111 X112 X113
[1,]    0   77  100   94   79
[2,]    0    0  108  124   28
[3,]    0    0    0  129  126
[4,]    0    0    0    0  131
[5,]    0    0    0    0    0
> DistanceRAMMinOst = DistanceRAMMinOst + t(DistanceRAMMinOst)
> DistanceRAMMinOst[1:5,1:5]
     X109 X110 X111 X112 X113
[1,]    0   77  100   94   79
[2,]   77    0  108  124   28
[3,]  100  108    0  129  126
[4,]   94  124  129    0  131
[5,]   79   28  126  131    0
> rownames(DistanceRAMMinOst)
NULL
> colnames(DistanceRAMMinOst)
 [1] "X109" "X110" "X111" "X112" "X113" "X114" "X115" "X116" "X117" "X118"
[11] "X119" "X120" "X121" "X122" "X123" "X124" "X125" "X126" "X127" "X128"
[21] "X129" "X130" "X131" "X132" "X133" "X134" "X135" "X136" "X137" "X138"
[31] "X139" "X140" "X141"
> WageCurveT15RAM.dfo$RAM[1:33]
 [1] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
127
[20] 128 129 130 131 132 133 134 135 136 137 138 139 140 141
> class(WageCurveT15RAM.dfo$RAM[1:33])
[1] "integer"
> RAMO = as.character(WageCurveT15RAM.dfo$RAM[1:33])
> class(RAMO)
[1] "character"
```

```
> RAMO
 [1] "109" "110" "111" "112" "113" "114" "115" "116" "117" "118" "119" "120"
[13] "121" "122" "123" "124" "125" "126" "127" "128" "129" "130" "131" "132"
[25] "133" "134" "135" "136" "137" "138" "139" "140" "141"
> rownames(DistanceRAMMinOst) = RAMO
> colnames(DistanceRAMMinOst) = RAMO
> DistanceRAMMinOst[1:5,1:5]
    109 110 111 112 113
109   0  77 100  94  79
110  77   0 108 124  28
111 100 108   0 129 126
112  94 124 129   0 131
113  79  28 126 131   0
> isSymmetric(DistanceRAMMinOst)
[1] TRUE
```

- **Distance-based weights matrix (minutes by car with threshold of 90 min.) and weights list object (row-standardized distance-based weights for RAMs)**

```
> WdistMinORAM = 1/DistanceRAMMinOst
> class(WdistMinORAM)
[1] "matrix"
> dim(WdistMinORAM)
[1] 33 33
> WdistMinORAM[1:5,1:5]
           109         110         111         112         113
109        Inf 0.012987013 0.010000000 0.010638298 0.012658228
110 0.01298701         Inf 0.009259259 0.008064516 0.035714286
111 0.01000000 0.009259259         Inf 0.007751938 0.007936508
112 0.01063830 0.008064516 0.007751938         Inf 0.007633588
113 0.01265823 0.035714286 0.007936508 0.007633588         Inf
> diag(WdistMinORAM) = 0
> WdistMinORAM[1:5,1:5]
           109         110         111         112         113
109 0.00000000 0.012987013 0.010000000 0.010638298 0.012658228
110 0.01298701 0.000000000 0.009259259 0.008064516 0.035714286
111 0.01000000 0.009259259 0.000000000 0.007751938 0.007936508
112 0.01063830 0.008064516 0.007751938 0.000000000 0.007633588
113 0.01265823 0.035714286 0.007936508 0.007633588 0.000000000
> which(is.infinite(WdistMinORAM))
integer(0)

> 1/90
[1] 0.01111111
> Wdist90MinORAM = WdistMinORAM
> Wdist90MinORAM[WdistMinORAM<1/90] = 0
> class(Wdist90MinORAM)
[1] "matrix"
> dim(Wdist90MinORAM)
[1] 33 33
```

```
> WdistMinORAM[1:5,1:5]
           109         110         111         112         113
109 0.00000000 0.012987013 0.010000000 0.010638298 0.012658228
110 0.01298701 0.000000000 0.009259259 0.008064516 0.035714286
111 0.01000000 0.009259259 0.000000000 0.007751938 0.007936508
112 0.01063830 0.008064516 0.007751938 0.000000000 0.007633588
113 0.01265823 0.035714286 0.007936508 0.007633588 0.000000000
> Wdist90MinORAM[1:5,1:5]
           109        110 111 112        113
109 0.00000000 0.01298701   0   0 0.01265823
110 0.01298701 0.00000000   0   0 0.03571429
111 0.00000000 0.00000000   0   0 0.00000000
112 0.00000000 0.00000000   0   0 0.00000000
113 0.01265823 0.03571429   0   0 0.00000000

> Wdist90MinORAM.lw = mat2listw(Wdist90MinORAM, style="W")
> class(Wdist90MinORAM.lw)
[1] "listw" "nb"
> summary(Wdist90MinORAM.lw)
Characteristics of weights list object:
Neighbour list object:
Number of regions: 33
Number of nonzero links: 206
Percentage nonzero weights: 18.91644
Average number of links: 6.242424
Link number distribution:
 3  4  5  6  7  8  9 10
 2  7  6  6  2  3  2  5
2 least connected regions:
121 128 with 3 links
5 most connected regions:
114 130 132 135 136 with 10 links
Weights style: W
Weights constants summary:
   n   nn S0       S1       S2
W 33 1089 33 12.48829 134.7847
```