Exercise: Areal Data and Spatial Autocorrelation

D G Rossiter
Cornell University, Section of Soil & Crop Sciences
ISRIC–World Soil Information
南京师范大学地理学学院

March 6, 2019

Contents

世上无难事，只怕有心人
"There are no difficult tasks, only fearful people" – Chinese proverb

"No hay mujeres difíciles, solo hombres incapaces" – Venezuelan proverb

1 Introduction

This tutorial gives an overview of spatial analysis of **areal data**, that is, attributes of polygonal entities on a map. Typical examples are political divisions, census tracts, and ownership or management parcels. The attribute relates to the whole area of the polygon, and can not be further localized. Often the data are **aggregate** measures, e.g., population count; these may already be **normalized** to the area of the polygon, e.g., population density.

After completing this exercise you should be able to:

1. Find nearest-neighbours in a polygon map according to various criteria (§3);
2. Compute spatial weights reflecting the strength of association according to various criteria (§4);
3. Compute global and local Moran's I as measures of spatial autocorrelation (§5);
4. Build a spatial autotregressive model that combines feature-space modelling ("regression") with spatial autocorrelation (§6);
5. Relate these to hypotheses about spatial processes.

A major complication for data analysis is that the **tesselation** (division of the study area) may have been done for a purpose not directly relevant to the analysis. For example, crop yield statistics may be aggregated by political division, but the crop yield may be better modelled by agro-ecological zone, a different tesselation. A further problem is that the analysis depends on the tesselation, and a different spatial scale, even of the same criterion, may show different behaviour. This is the **modifiable areal unit problem**. For example, crop statistics by county may show strong spatial autocorrelation, which becomes much weaker at district or state level, although the underlying process is the same.

This tutorial is based on Bivand et al. [1, Ch. 9], which has a more extensive treatment, especially in the details of the R processing of this kind of data. Some of the code here is adapted from that chapter; the sample dataset from the Syracuse region is their adaptation of the dataset of Waller and Gotway [5].

We eventually want to examine spatial dependence among polygonal areas; but to do that we first need to create spatial weights, i.e., the degree

of "neighbourliness" between areas; but to do that we first need to define spatial neighbours. These are treated then in reverse order.

> **Note:** The code in these exercises was tested with `knitr` package Version: 1.21 [6] `sp` package Version: 1.3-1, `spdep` package Version: 1.0-2, `gstat` package Version: 1.0-2 on R version 3.5.2 (2018-12-20), running on Mac OS X 10.14.3. So, the text and graphical output you see here was automatically generated and incorporated into LaTeX by running the code through R and its packages. Then the LaTeX document was compiled into the PDF version you are now reading. Your output may be slightly different on different versions and on different platforms.

2 Example dataset

The sample data is 281 USA census tracts for eight central New York State counties[1] developed by Waller and Gotway [5] and adapted by Bivand et al. [1]; the area is about 160 km N-S and 120 km E-W. The dataset is provided at the ASDAR book website[2] under the "Data sets download" tab as "New York leukemia dataset"[3].

> **Note:** In the USA census tracts have 1 500–8 000 people (optimum size 4 000). They are designed to be socio-economically and demographically fairly homogeneous. Each tract has several block groups; these are made up of 20–40 individual blocks. The tract is usually large enough to compile reliable statistics.

To orient you to this region, Figure 1 shows a map of the counties .

Task 1 : Locate this file, unpack it in a working directory, start R and connect to that directory. •

After connecting to your working directory and unpacking `NY_data.zip` to its own subdirectory, you should see the following files:

```
list.files("./NY_data")
```

```
 [1] "NY_nb.gal"     "NY8_utm18.dbf" "NY8_utm18.gal" "NY8_utm18.gwt"
 [5] "NY8_utm18.prj" "NY8_utm18.shp" "NY8_utm18.shx" "NY8cities.dbf"
 [9] "NY8cities.fix" "NY8cities.prj" "NY8cities.qix" "NY8cities.shp"
[13] "NY8cities.shx" "TCE.dbf"       "TCE.prj"       "TCE.shp"
[17] "TCE.shx"
```

> **Note:** Path specification in R follows Unix conventions for a hierarchical file system: `"."` stands for the current directory[4], `"./"` symbolizes descent into a named sub-directory, here `NY_data`, and `".."` symbolizes ascent to the directory above the current one in the directory hierarchy.
>
> The named subdirectory should have been created as `NY_data.zip` was unpacked. If the data files are in a different directory, just substitute its name, or if they are in the already-connected directory, leave off the `/NY_data`.

[1] Broome, Cayuga, Chenango, Cortland, Madison, Onondaga, Tioga, Tompkins
[2] http://www.asdar-book.org/
[3] `NY_data.zip`
[4] which you can see with `getwd()` command

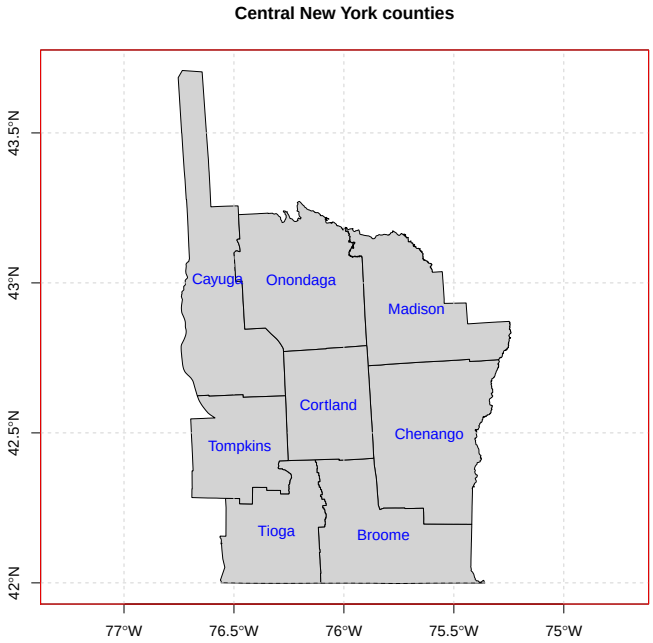


Figure 1: Central New York State counties

> You can change the working directory, i.e., the one symbolized by `"."`, with the `setwd` function. This can be **relative** to the current working directory, e.g., `setwd("../../ds/NY_data")` to go up two levels and then back down one level to a subdirectory named `ds` and then another level down to a sub-subdirectory named `NY_data`. Or you can start at the root of the file system for an **absolute** path name, by starting the path specification with `/`, e.g., `setwd("/data")`; you can also include a drive name on Windows systems, e.g., `setwd("D:/data")`.

Most of the analysis in this tutorial is carried out by functions in the `spdep` "Spatial dependence" package from Roger Bivand. This depends on the `sp` "spatial classes" package. We also use the `rgdal` "R interface to GDAL" package to import the shapefiles in the dataset.

TASK 2 : Load the required packages. •

We use the `require` function; this loads the package if it is not already in the workspace:

```
require(sp)
require(rgdal)
require(spdep)
```

TASK 3 : Import the polygon data into R, along with point files showing cities. •

Polygon shapefiles can be imported to R with the `readOGR` function of the `rgdal` package. We then examine the class of the imported object with

the `class` function, and see its slots with the `str` "structure" function.

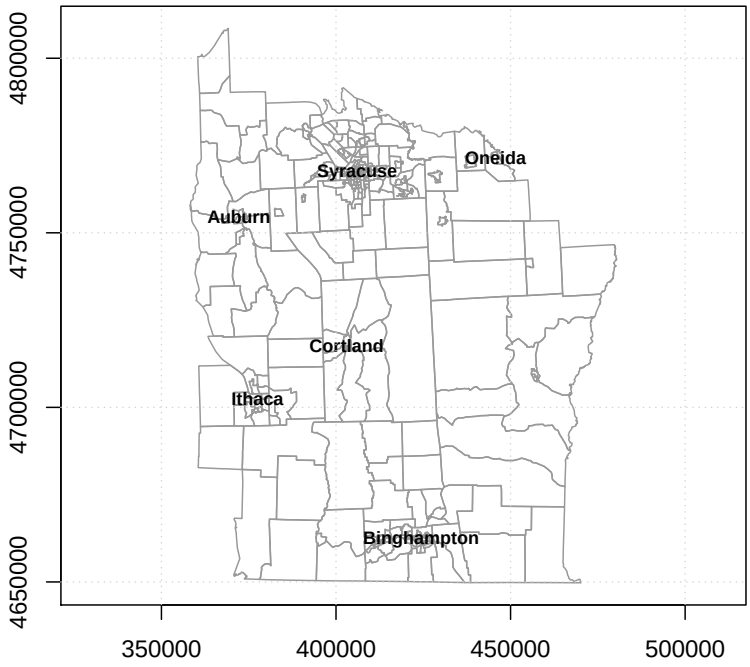
```
NY8 <- readOGR("./NY_data", "NY8_utm18")
class(NY8)
str(NY8, max.level=2)
cities <- readOGR("./NY_data", "NY8cities")
class(cities)
str(cities, max.level=2)
```

```
OGR data source with driver: ESRI Shapefile
Source: "/Users/rossiter/data/edu/dgeostats/ex/ds/ASDAR/NY_data", layer: "NY8_utm18"
with 281 features
It has 17 fields
[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"
Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame': 281 obs. of  17 variables:
  ..@ polygons   :List of 281
  ..@ plotOrder  : int [1:281] 75 83 81 82 252 79 106 104 278 258 ...
  ..@ bbox       : num [1:2, 1:2] 358242 4649755 480393 4808545
  .. ..- attr(*, "dimnames")=List of 2
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
OGR data source with driver: ESRI Shapefile
Source: "/Users/rossiter/data/edu/dgeostats/ex/ds/ASDAR/NY_data", layer: "NY8cities"
with 6 features
It has 1 fields
[1] "SpatialPointsDataFrame"
attr(,"package")
[1] "sp"
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame': 6 obs. of  1 variable:
  ..@ coords.nrs : num(0)
  ..@ coords     : num [1:6, 1:2] 424374 377506 403020 372237 406070 ...
  .. ..- attr(*, "dimnames")=List of 2
  ..@ bbox       : num [1:2, 1:2] 372237 4662141 445728 4771698
  .. ..- attr(*, "dimnames")=List of 2
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
```

As you can see, `readOGR` reports the feature type of the source file, and converts these to `sp` objects. @

TASK 4 : Plot the polygons, with cities overlaid and labelled. •

```
plot(NY8, border="grey60", axes=TRUE, asp=1)
text(coordinates(cities), labels=as.character(cities$names), font=2, cex=0.75)
grid()
```



Note: Whoever compiled the data is obviously not a Central New Yorker; only Long Islanders write "Bing**hamp**ton", by analogy with East**hamp**ton etc., rather than "Bing**ham**ton", i.e., Bing**ham**'s Town[5]. Although, the "ham" in Bingham has the same meaning as the "hamp" in the various Hamptons, deriving from Germanic root that is now German '*heim*' and English 'home'.

Note: The discrepancy between this map and Figure 1 in the NW is because the county map includes part of Lake Ontario to the Canadian border in the middle of the lake.

Q1 : *What is the georeference of this dataset? This is referred to as the Coördinate Reference System (CRS).* *Jump to A1* •

The `proj4string` function extracts this information from objects of one of the `sp` classes, here a `SpatialPolygonsDataFrame`:

```
proj4string(NY8)

[1] "+proj=utm +zone=18 +ellps=WGS84 +units=m +no_defs"

proj4string(cities)

[1] "+proj=utm +zone=18 +ellps=WGS84 +units=m +no_defs"
```

[5] named for William Bingham, a Philadelphia politician and land speculator who bought the land, then part of Tioga County, in 1792.

We first examine the spatial organization of the area data ("Spatial neighbours", §3) and how to weight the neighbourhood relations ("Spatial weights", §4). We then describe the attributes of each area and how to discover the autocorrelation structure ("Spatial autocorrelation", §5).

3 Spatial neighbours

Supplementary reading:

- Bivand et al. [1, §9.2]: *Spatial neighbours & spatial weights*

3.1 Importing a neighbour list in GAL format

The `spdep` package represents neighbour relationships by an object of class `nb`; this stores a list of each polygon, each with a list of the index numbers of neighbours. These can be created from polygons as explained in §3.2, below, but here we have one already created for us and provided with the `NY_data` dataset as file `NY_nb.gal`, in the so-called "GAL lattice" format[6].

TASK 5 : Import the neighbour list and summarize it. •

The `read.gal` function imports GAL lattice files to `nb` objects. Here we use the optional `region.id` argument to specify the labels for each region. We get them in this case from the `row.names` of the `NY8` object's `data` slot of the `SpatialPolygonsDataFrame` object, i.e., the data frame with its attributes.

```
NY8_nb <- read.gal("./NY_data/NY_nb.gal", region.id=row.names(NY8@data))
```

```
summary(NY8_nb)

Neighbour list object:
Number of regions: 281
Number of nonzero links: 1522
Percentage nonzero weights: 1.927534
Average number of links: 5.41637
Link number distribution:

 1  2  3  4  5  6  7  8  9 10 11
 6 11 28 45 59 49 45 23 10  3  2
6 least connected regions:
55 97 100 101 244 245 with 1 link
2 most connected regions:
34 82 with 11 links
```

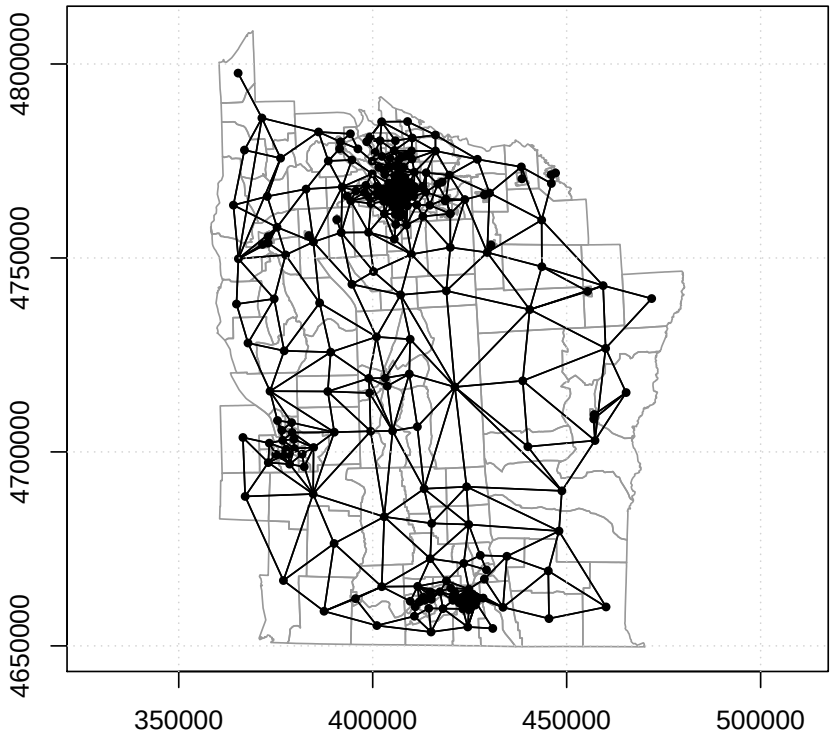
Q2 : *How many links are there between regions? What is the average number of links for an arbitrary polygon? How many polygons have only one link to another polygon?* *Jump to A2* •

TASK 6 : Plot the polygons with the links superimposed. •

[6] These are Luc Anselin's GeoDa files; see the help for `read.gal` for details

The generic `plot` method specializes to `plot.nb` when asked to plot an object of class `nb`. Note the `ADD=TRUE` argument to add this to the polygon plot.

```
plot(NY8, border="grey60", axes=TRUE)
plot(NY8_nb, coordinates(NY8), pch=19, cex=0.6, add=TRUE)
grid()
```



Q3 : *How are these first-order (direct) links defined? Are they of approximately equal length? What accounts for the difference? Should all neighbours be weighted equally when considering spatial influence between polygons?* *Jump to A3*

•

This is a fairly large dataset; to make the analysis faster and the figures easier to understand, we subset the data.

Q4 : *What field in the dataframe of* `NY8` *gives the geographic names? How many are there? What is the factor name for Syracuse city?* *Jump to A4* •

We can find the field names with the `names` function, and then the area

names with the `levels` function applied to the relevant field; to avoid showing the list we use the `tail` function, since we know Syracuse is probably towards the end of the list.

```
names(NY8)

 [1] "AREANAME"   "AREAKEY"    "X"          "Y"          "POP8"
 [6] "TRACTCAS"   "PROPCAS"    "PCTOWNHOME" "PCTAGE65P"  "Z"
[11] "AVGIDIST"   "PEXPOSURE"  "Cases"      "Xm"         "Ym"
[16] "Xshift"     "Yshift"

tail(levels(NY8$AREANAME))

[1] "Remainder of Van Bure" "Skaneateles village"
[3] "Solvay village"        "Spafford town"
[5] "Syracuse city"         "Vestal town"
```

TASK 7 : Extract the subset of the data covering the city of Syracuse[7] and plot it, along with its neighbour links. •

We subset the data with a logical expression to select matrix rows; this works on the `@data` slot, but brings along all the polygon topology with the selected records.

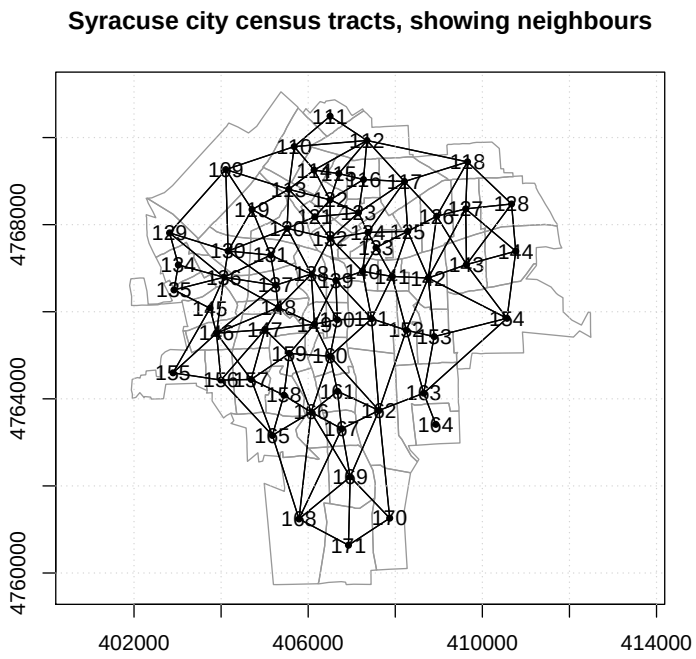
```
Syr <- NY8[NY8$AREANAME == "Syracuse city",]
summary(Syr)

Object of class SpatialPolygonsDataFrame
Coordinates:
        min       max
x  401899.8  412491.4
y 4759733.4 4771050.5
Is projected: TRUE
proj4string :
[+proj=utm +zone=18 +ellps=WGS84 +units=m +no_defs]
Data attributes:
                  AREANAME         AREAKEY         X
 Syracuse city        :63   36067000100: 1   Min.   :-16.723
 Auburn city          : 0   36067000200: 1   1st Qu.:-14.032
 Baldwinsville village: 0   36067000300: 1   Median :-12.852
 Barker town          : 0   36067000400: 1   Mean   :-12.862
 Bayberry-Lynelle Mead: 0   36067000500: 1   3rd Qu.:-11.663
 Binghamton city      : 0   36067000600: 1   Max.   : -8.704
 (Other)              : 0   (Other)    :57
       Y              POP8         TRACTCAS        PROPCAS
 Min.   :31.78   Min.   :   9   Min.   :0.000   Min.   :0.0000000
 1st Qu.:36.22   1st Qu.:1696   1st Qu.:0.040   1st Qu.:0.0000150
 Median :37.99   Median :2662   Median :1.040   Median :0.0005060
 Mean   :37.66   Mean   :2700   Mean   :1.675   Mean   :0.0007322
 3rd Qu.:39.40   3rd Qu.:3184   3rd Qu.:2.560   3rd Qu.:0.0010995
 Max.   :41.74   Max.   :9393   Max.   :7.090   Max.   :0.0069930

   PCTOWNHOME          PCTAGE65P              Z
 Min.   :0.0008224   Min.   :0.004044   Min.   :-1.44174
 1st Qu.:0.1728601   1st Qu.:0.083726   1st Qu.:-0.57247
 Median :0.3711340   Median :0.144109   Median :-0.06904
 Mean   :0.3746292   Mean   :0.148330   Mean   : 0.03775
 3rd Qu.:0.5019480   3rd Qu.:0.175184   3rd Qu.: 0.43847
 Max.   :1.0000000   Max.   :0.505050   Max.   : 4.71053

    AVGIDIST         PEXPOSURE          Cases
 Min.   :0.02447   Min.   :0.8949   Min.   :0.00014
 1st Qu.:0.02663   1st Qu.:0.9793   1st Qu.:0.04514
```

[7] http://www.syracuse.ny.us/

```
 Median :0.02761   Median :1.0154   Median :1.04486
 Mean   :0.02763   Mean   :1.0148   Mean   :1.67566
 3rd Qu.:0.02868   3rd Qu.:1.0537   3rd Qu.:2.55945
 Max.   :0.03101   Max.   :1.1318   Max.   :7.08252

       Xm               Ym            Xshift           Yshift
 Min.   :-16723   Min.   :31784   Min.   :402599   Min.   :4760639
 1st Qu.:-14032   1st Qu.:36220   1st Qu.:405289   1st Qu.:4765075
 Median :-12852   Median :37992   Median :406470   Median :4766847
 Mean   :-12862   Mean   :37659   Mean   :406460   Mean   :4766514
 3rd Qu.:-11663   3rd Qu.:39397   3rd Qu.:407658   3rd Qu.:4768253
 Max.   : -8704   Max.   :41735   Max.   :410617   Max.   :4770591


Syr_nb <- subset(NY8_nb, NY8$AREANAME == "Syracuse city")
summary(Syr_nb)

Neighbour list object:
Number of regions: 63
Number of nonzero links: 346
Percentage nonzero weights: 8.717561
Average number of links: 5.492063
Link number distribution:

 1  2  3  4  5  6  7  8  9
 1  1  5  9 14 17  9  6  1
1 least connected region:
164 with 1 link
1 most connected region:
136 with 9 links
```

We have reduced the size of the dataset. Now the plot. Note the use of the `row.names` function to extract the census tract numbers, and the `text` function to place text on the plot.

```
plot(Syr, border="grey60", axes=TRUE)
title("Syracuse city census tracts, showing neighbours")
plot(Syr_nb, coordinates(Syr), pch=19, cex=0.6, add=TRUE)
text(coordinates(Syr), row.names(Syr))
grid()
```



TASK 8 : Display the distribution of the number of links. •

We apply the `length` "length of vector" function across the list of points; this returns the number of neighbours of the point; then convert the list to a vector with the `unlist` function. Then the `table` function shows the number of census tracts with each number of neighbours.

```
(n.nb <- unlist(lapply(Syr_nb, length)))

 [1] 5 5 2 7 7 5 4 5 7 5 5 7 5 7 6 6 6 6 6 4 4 7 6 8 4 3 3 9 6 8 6 6 6
[34] 8 6 4 4 8 6 6 7 5 7 6 4 5 3 5 6 4 6 7 4 8 5 1 5 8 5 5 6 3 3

table(n.nb)

n.nb
 1  2  3  4  5  6  7  8  9
 1  1  5  9 14 17  9  6  1
```

Q5 : *What is the most common number of neighbours?* *Jump to A5* •

TASK 9 : Identify a polygon with only one neighbour, and one with a large number of neighbours; we will use these for comparing how spatial weights are computed in the next section. •

We find the polygons with the minimum and maximum number of neighbours with the `which` function and a logical condition using either `min` or `max` and the == "is equal to" operator, and then display their information in the polygon object by row selection.

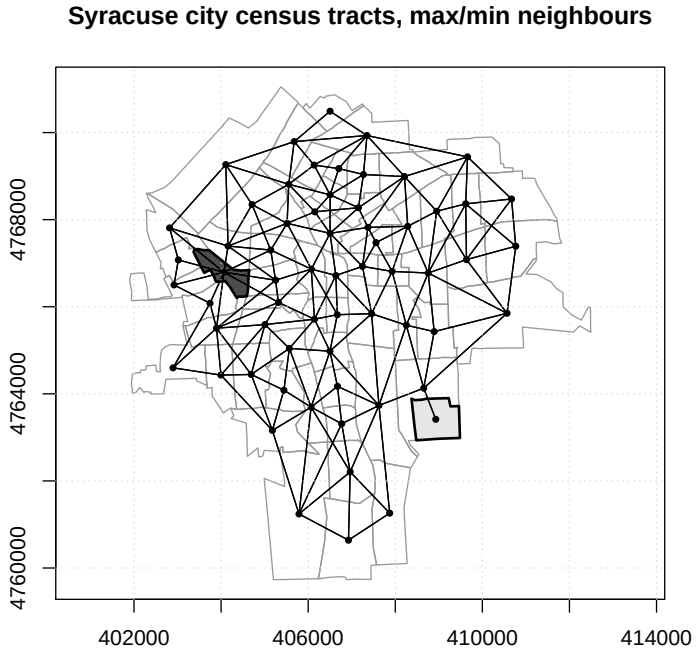
```
min(n.nb); max(n.nb)

[1] 1
[1] 9

(ix.min <- which(n.nb==min(n.nb)))

[1] 56

Syr@data[ix.min,]

         AREANAME     AREAKEY        X      Y POP8 TRACTCAS PROPCAS
164 Syracuse city 36067005602 -10.5923 34.676 2720     0.04 1.5e-05
    PCTOWNHOME  PCTAGE65P        Z  AVGIDIST PEXPOSURE   Cases
164 0.00082237 0.00404412 -0.96141 0.0267115  0.982509 0.04104
          Xm    Ym   Xshift  Yshift
164 -10592.3 34676 408729.3 4763531

(ix.max <- which(n.nb==max(n.nb)))

[1] 28

Syr@data[ix.max,]

         AREANAME     AREAKEY        X        Y POP8 TRACTCAS PROPCAS
136 Syracuse city 36067002900 -15.4437 38.02805 1189     0.01   8e-06
    PCTOWNHOME PCTAGE65P        Z  AVGIDIST PEXPOSURE   Cases
136  0.3841699 0.1488646 -0.16316 0.0294166  1.078974 0.01794
          Xm       Ym   Xshift  Yshift
136 -15443.7 38028.05 403877.9 4766883
```

In this case there is only one polygon with minimum and maximum neighbours.

Q6 : *How many polygons have only one neighbour? What is the maximum number of neighbours for any polygon? What are their indices in the list of polygons for Syracuse? What are their census codes (see field `AREAKEY`)? What are their indices in the 8-county dataset (these are the `row.names`)?* *Jump to A6* •

We plot the locations of these two polygons:

```
plot(Syr, border="grey60", axes=TRUE)
title("Syracuse city census tracts, max/min neighbours")
plot(Syr[ix.min,], border="black", col=grey(.9), lwd=2, add=T)
plot(Syr[ix.max,], border="black", col=grey(.3), lwd=2, add=T)
plot(Syr_nb, coordinates(Syr), pch=19, cex=0.6, add=TRUE)
grid()
```



3.1.1 * Geographic setting

It always helps understanding to see the geographic setting of a dataset; in this optional sectio we show how to display the Syracuse census tracts on Google Earth.

TASK 10 : Export the Syracuse polygons in KML format and display in Google Earth. •

The `writeOGR` function of the `rgdal` package exports in many formats. For display in Google Earth, coordinates must be in Longitude/Latitude on the WGS84 ellipsoid; we use the `spTransform` method to transform from the original UTM to this system, defined with the `CRS` function to specify the `@proj4string` field.

```
Syr.ll <- spTransform(Syr, CRS("+proj=longlat +ellps=WGS84"))
writeOGR(Syr.ll, dsn="./Syr.kml", layer="Syracuse", driver="KML", overwrite_layer=TRUE)
```

```
Warning in fld_names == attr(res, "ofld_nms"):  longer object length is not a multiple
of shorter object length
```

Opening the KML in Google Earth and adjusting the symbology, we see Figure 2.

3.2 Creating neighbours from polygons

In this example a neighbour list was provided, but in many situations we have the polygons but must create our own neighbour list. This also gives flexibility in defining what is a “neighbour”.

Figure 2: Syracuse census tracts

3.2.1 Neighbours based on contiguity

The `poly2nb` function of the `spdep` package takes a `SpatialPolygons` or `SpatialPolygonsDataFrame` object and finds neighbours, returning a neighbours list of class `nb`. This function defines a "neighbour" as a polygon that shares a boundary ("rook" and "queen" neighbour) or boundary point ("queen" neighbour with the target polygon.

TASK 11 : Compute a neighbours list for Syracuse. •

We check whether this list matches the imported GAL file with the `all.equal` "are all equal?" function:

```
Syr_nb2 <- poly2nb(Syr)
all.equal(Syr_nb, Syr_nb2, check.attributes = F)

[1] TRUE
```

This gives the identical list. However, `poly2nb` has two optional arguments that can greatly affect the list:

- `snap`: boundary points less than a "snap" distance apart are considered to be contiguous; default a very small machine-dependent quantity, effectively zero.
- `queen`: a single shared boundary point meets the contiguity condition (so, in chess, the queen could move between the polygons, but a rook could not); default `TRUE`.

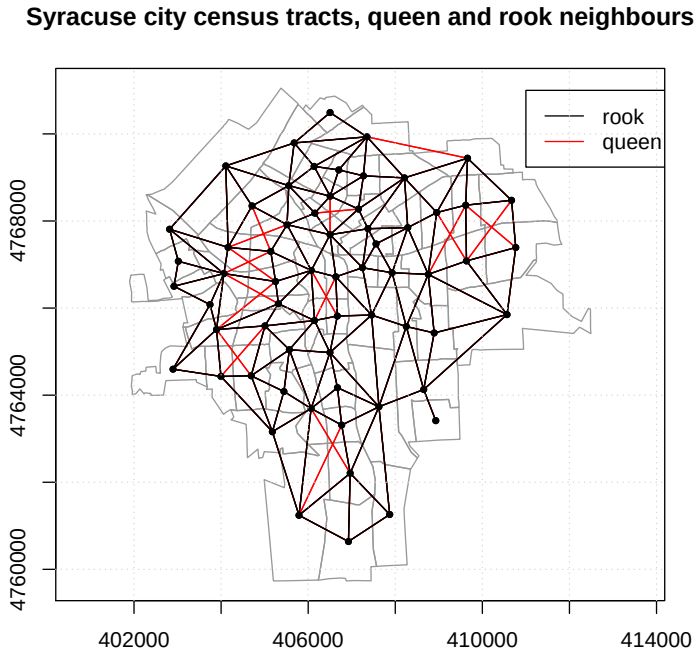
The `snap` argument is useful for (1) poorly-digitized maps; (2) to skip over small polygons, e.g., a small river or highway that is given as a separate polygon. Setting the `queen` argument to `FALSE` reduces the number of neighbours and requires a shared boundary line.

TASK 12 : Compute the neighbour list with rook (not queen) contiguity; plot the polygon map with the rook links in black and the deleted queen links in red. •

```
Syr_nb2 <- poly2nb(Syr, queen=FALSE)
summary(Syr_nb2)

Neighbour list object:
Number of regions: 63
Number of nonzero links: 308
Percentage nonzero weights: 7.760141
Average number of links: 4.888889
Link number distribution:

 1  2  3  4  5  6  7  8
 1  1  7 18 15 11  9  1
1 least connected region:
164 with 1 link
1 most connected region:
162 with 8 links
```

```
plot(Syr, border="grey60", axes=TRUE)
title("Syracuse city census tracts, queen and rook neighbours")
plot(Syr_nb, coordinates(Syr), pch=19, cex=0.6, add=TRUE, col="red")
```

```
plot(Syr_nb2, coordinates(Syr), pch=19, cex=0.6, add=TRUE)
legend(411000, 4771000, c("rook","queen"), lty=1, col=c("black","red"))
grid()
```



Q7 : *How many links were deleted? Which set of links more realistically represents the concept of "neighbour" in a city?* *Jump to A7* •

3.2.2 Neighbours based on distance between centroids

There are other concepts of neighbours. One is by distance: a "neighbour" polygon is not necessarily contiguous with the target polygon, but whose centroid is within a given distance band of the target polygon's centroid, will be considered a neighbour.

This is appropriate if the spatial process is hypothesized to depend on distance rather than contiguity, and there is a radius over which the process is hypothesized to operate.

TASK 13 : Find the neighbours of each polygon within 1.2 km. •

The `dnearneigh` function computes this, using a point set as the target. These can be arbitrary points, but here we want the centroids of the target polygons. The `coordinates` method applied to a `SpatialPolygons` object returns the polygon centroids.

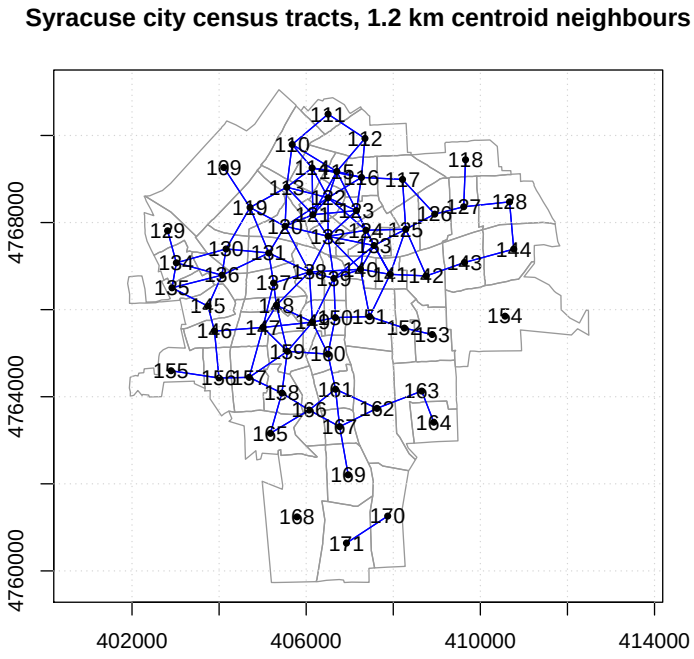
```
Syr_nb_d <- dnearneigh(coordinates(Syr), d1=0, d2=1200,
                       row.names=row.names(Syr))
Syr_nb_d
```

```
Neighbour list object:
Number of regions: 63
Number of nonzero links: 252
Percentage nonzero weights: 6.349206
Average number of links: 4
2 regions with no links:
154 168
```

There is no requirement that the `d1` "closest distance" be zero; this function can be used to find "neighbours" in any distance band.

TASK 14 : Plot the polygon map with these neighbour links. •

```
plot(Syr, border="grey60", axes=TRUE)
title("Syracuse city census tracts, 1.2 km centroid neighbours")
plot(Syr_nb_d, coordinates(Syr), pch=19, cex=0.6, col="blue", add=TRUE)
text(coordinates(Syr), row.names(Syr))
grid()
```



3.2.3 Nearest neighbours based on distance

Another possibility is to define a fixed number of nearest neighbours, again based on centroid distance.

This is a appropriate if the spatial process is hypothesized to depend on a fixed set of nearest neighbours, no matter their distances.

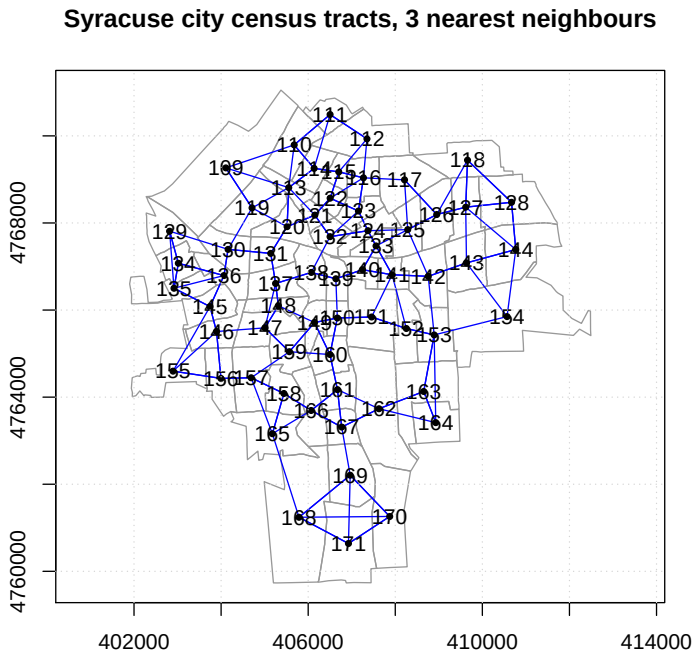
TASK 15 : Create a neighbours list including the three nearest neighbours of each polygon. •

The `knearneigh` function finds the nearest neighbours as a matrix, then the `knn2nb` function converts this to a neighbours list.

```
knn3 <- knearneigh(coordinates(Syr), k=3)
str(knn3$nn)

 int [1:63, 1:3] 11 6 4 8 6 7 6 7 8 19 ...

knn3$nn[1:3,]

     [,1] [,2] [,3]
[1,]   11    5    2
[2,]    6    5    3
[3,]    4    2    6

Syr_nb_3nn <- knn2nb(knn3, row.names=row.names(Syr))
Syr_nb_3nn

Neighbour list object:
Number of regions: 63
Number of nonzero links: 189
Percentage nonzero weights: 4.761905
Average number of links: 3
Non-symmetric neighbours list
```

TASK 16 : Plot the polygon map with these neighbour links. •

```
plot(Syr, border="grey60", axes=TRUE)
title("Syracuse city census tracts, 3 nearest neighbours")
plot(Syr_nb_3nn, coordinates(Syr), pch=19, cex=0.6, col="blue", add=TRUE)
text(coordinates(Syr), row.names(Syr))
grid()
```



4 Spatial weights

Supplementary reading:

- Bivand et al. [1, §9.2]: *Spatial neighbours & spatial weights*

Spatial weights extend the list of neighbours for a point, by assigning some value between 0 (no relation) and 1 (full relation). In the simplest case we have only 0's and 1's: a neighbour of a point is either influential (1) or not (0), and all influential neighbours are equally influential in the process we are modelling. This simple view can be modified by assigning different weights to each relationship, but of course we must have knowledge of the underlying process to deviate from the simple 0/1 model.

For example, a weight might be proportional to the distance between polygon centroids (spatial diffusion process) or length of shared boundary (migration process) or size of the neighbour polygon (pressure process) – in this last case, the weights would be *asymmetric*.

Spatial weights are represented in `spdep` as a list of lists: (1) points, (2) neighbours of that point, with weights on $[0 \ldots 1]$. They can also be represented as a matrix: rows for the source point and columns for the target. An entry of 0 means the points are not neighbours.

Task 17 : Create a weights object for the Syracuse polygons, with the default (queen's) neighbour list. •

The `nb2listw` function of the `spdep` package converts a neighbours list object (class `nb`) to a weights object (class `listw`, an extension of `nb`).

There are various conversion styles as an optional argument; the default is `style="W"`, in which the weights for each areal entity must sum to unity along rows of the weights matrix; this is the inverse of the number of neighbours. This may give a false impression at the edges of the study area, where fewer neighbours are expected. We discuss some other spatial weight styles in §5.1.1.

We create the weights object, summarize it, and examine its structure:

```
Syr_lw_W <- nb2listw(Syr_nb)
print(Syr_lw_W)

Characteristics of weights list object:
Neighbour list object:
Number of regions: 63
Number of nonzero links: 346
Percentage nonzero weights: 8.717561
Average number of links: 5.492063

Weights style: W
Weights constants summary:
   n   nn S0       S1      S2
W 63 3969 63 24.78291 258.564

str(Syr_lw_W, max.level=1)
```

```
List of 3
 $ style     : chr "W"
 $ neighbours:List of 63
  ..- attr(*, "region.id")= chr [1:63] "109" "110" "111" "112" ...
  ..- attr(*, "class")= chr "nb"
  ..- attr(*, "GeoDa")=List of 2
  ..- attr(*, "gal")= logi TRUE
  ..- attr(*, "call")= logi TRUE
  ..- attr(*, "sym")= logi TRUE
 $ weights   :List of 63
  ..- attr(*, "mode")= chr "binary"
  ..- attr(*, "W")= logi TRUE
  ..- attr(*, "comp")=List of 1
 - attr(*, "class")= chr [1:2] "listw" "nb"
 - attr(*, "region.id")= chr [1:63] "109" "110" "111" "112" ...
 - attr(*, "call")= language nb2listw(neighbours = Syr_nb)
 - attr(*, "GeoDa")=List of 2
```

```
print(Syr_lw_W$neighbours[[1]])
```

```
[1]  2  5 11 21 22
```

```
print(Syr_lw_W$weights[[1]])
```

```
[1] 0.2 0.2 0.2 0.2 0.2
```

The object is composed of three lists:

1. the weights style `style`, here `style="W"`;
2. a list of the regions, each having a vector of its neighbours' region numbers;
3. a list of the regions, each having a vector of the weights given to each neighbouring region.

In the above code, we see that region 1 has five neighbouring regions (`2, 5, 11, 21, 22`), and each has equal weight `1/5 = 0.2`. To extract these, we used the `[[]]` list extraction operator.

Q8 : *What are the weights of each link for the one-neighbour polygon? for the nine-neighbour polygon?* *Jump to A8* •

These are the respective lists for the two identified polygons:

```
Syr_lw_W$weights[[ix.min]]
```

```
[1] 1
```

```
Syr_lw_W$weights[[ix.max]]
```

```
[1] 0.1111111 0.1111111 0.1111111 0.1111111 0.1111111 0.1111111
[7] 0.1111111 0.1111111 0.1111111
```

The weights are stored in list format because so many of them will be zero, i.e., a full matrix is **sparse**. However, it is possible to "unwrap" the list into a full matrix.

Task 18 : **Optional**: Convert the weights into a full matrix and display the upper 9 x 9 corner, i.e., the weights between the first ten regions. •

We write a small `function` to do the conversion from an object of class `nb`; we can use this with any neighbour list.

> **Note:** To make the result more interpretable, we format the `matrix` as a `data.frame`, and name the rows and columns of the data frame using the `row.names` and `colnames` functions (yes, one has a `.` and one does not ...). The polygon names are found in the `region.id` attribute of the neighbours object; we extract these with the `attr` "get attributes" function.

```
build.wts.matrix <- function(wts.list) {
    # set up a matrix to receive the weights, initially all 0
    len <- length(wts.list$weights)
    wts.matrix <- as.data.frame(matrix(0, nrow=len, ncol=len))
    row.names(wts.matrix) <- attr(wts.list$neighbours, "region.id")
    colnames(wts.matrix) <- attr(wts.list$neighbours, "region.id")
    for (i in 1:len) { # each item in the weights list
        nl <- wts.list$neighbours[[i]]  ## one row's neighbours
        wl <- wts.list$weights[[i]]     ## one row's weights
        if (nl[1] != 0) {  # empty neighbour lists have a single `0' element
            # fill in this row of the weights matrix
            for (j in 1:length(nl))  wts.matrix[i, nl[j]] <- wl[j]
            }
    }
    return(wts.matrix)
}
tmp <- build.wts.matrix(Syr_lw_W)
tmp[1,]

    109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124
109   0 0.2   0   0 0.2   0   0   0   0   0 0.2   0   0   0   0   0
    125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
109   0   0   0   0 0.2 0.2   0   0   0   0   0   0   0   0   0   0
    141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156
109   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    157 158 159 160 161 162 163 164 165 166 167 168 169 170 171
109   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

round(tmp[1:9,1:9], 4)

       109    110    111    112 113    114    115    116    117
109 0.0000 0.2000 0.0000 0.0000 0.2 0.0000 0.0000 0.0000 0.0000
110 0.2000 0.0000 0.2000 0.2000 0.2 0.2000 0.0000 0.0000 0.0000
111 0.0000 0.5000 0.0000 0.5000 0.0 0.0000 0.0000 0.0000 0.0000
112 0.0000 0.1429 0.1429 0.0000 0.0 0.1429 0.1429 0.1429 0.1429
113 0.1429 0.1429 0.0000 0.0000 0.0 0.1429 0.0000 0.0000 0.0000
114 0.0000 0.2000 0.0000 0.2000 0.2 0.0000 0.2000 0.0000 0.0000
115 0.0000 0.0000 0.0000 0.2500 0.0 0.2500 0.0000 0.2500 0.0000
116 0.0000 0.0000 0.0000 0.2000 0.0 0.0000 0.2000 0.0000 0.2000
117 0.0000 0.0000 0.0000 0.1429 0.0 0.0000 0.0000 0.1429 0.0000

rm(tmp)
```

Notice that the full weights matrix for weights style `W` is *not* symmetric; the number of neighbours of a target polygon is not the same as the number of other polygons with which this one shares influence on a different target.

Note that the analyst can directly create a weights matrix based on knowledge of the assumed data generating process.

5 Spatial autocorrelation

Supplementary reading:

- Bivand et al. [1, §9.3]: *Testing for spatial autocorrelation'*

Now that we have neighbours and their weights, we can determine whether there is any **spatial autocorrelation**: are attribute values in neighbouring polygons (suitably weighted) similar? Note we are not yet trying to determine causes, although the results of this step may motivate a hypothesis. For example, neighbouring polygons could influence each other; alternately, a geographic factor common to adjacent areas could influence them both.

However, we first need to describe the feature-space attributes of each area. These are all reported on the basis of 1980 census tracts.

`Cases` : the number of leukaemia cases 1978–1982; some cases had insufficient georeference, these were added proportionally to tracts, so some "counts" are not integers.

`Z` : log-transformed rate, i.e., normalized by census tract population: $Z_i = \log(1000[\text{Cases} + 1]/n)$

`PEXPOSURE` : "potential exposure", computed as the logarithm of 100 times the inverse of the distance between a census tract centroid and the nearest TCE[8]-producing site[9];

`PCTAGE65P` : percent older than 65 years; this could represent long-term exposure to any environmental factor;

`PCTOWNHOME` : percent home ownership; this could indicate lifestyle or economic level.

In this section we examine spatial autocorrelation of the transformed disease incidence, attribute `Z`. Among the various metrics of spatial association, we choose Moran's I [3]. In all such tests, we make several implicit assumptions:

- we assume that there is no spatial patterning due to some underlying but un-modelled factor;
- we assume that the assigned spatial weights (previous §) are those that generated the autocorrelation.

As examples of these:

- If assessing spatial correlation of disease incidence, we assume there are no environmental factors that are spatially-distributed, e.g., industry or different water sources.
- Equal spatial weights of $1/n$ from each of n neighbours assumes that each neighbour is equally influential in the modelled process. If the process depends for example on the "pressure" due to population or area of a polygon, this is unlikely to be true.

[8] Trichloroethylene, an industrial solvent often found in groundwater
[9] see `?NY_data`

So tests such as Moran's I should ideally be applied to **residuals** after removing known spatial patterning, and with weights based on the assumed process that gave rise to autocorrelation. What is left can then be tested to see if there is a real effect of spatial correlation, not one brought on by a "lurking variable".

As an example, we might hypothesize that the crime rate in a city is (at least in part) related to low incomes, low employment, low home ownership, and number of abandoned houses. If we can build a model (non-spatial) relating these factors to crime rate, any apparent spatial correlation in crime rate may disappear in the residuals, because the predictive factors share the same spatial patterning, i.e., the mean model is not a null (average) model but instead has spatially-pattered predictors.

However, in some data sets we don't have the spatially-patterned covariables; or, we want to test if there is any spatial patterning, not considering the cause (perhaps to see if there is any cause); then Moran's I and similar tests can be applied to the variable without attempting to model it with covariables.

Moran's I is defined as:

$$I = \frac{n}{\sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij}(y_i - \bar{y})(y_j - \bar{y})}{\sum_i (y_i - \bar{y})^2} \tag{1}$$

where y_i is the ith of n polygon, $\bar{y}$ is its global mean, and w_{ij} is the spatial weight of the link between polygons i and j, as discussed in the previous section. The first term normalizes by the sum of all weights, so the test is comparable among datasets with different numbers of polygons. The denominator of the second term centres on the mean.

5.1 Global tests

A **global** test summarizes the spatial correlation of an entire map: is there evidence of spatial correlation, on average? We consider the incidence of leukemia, presented as a log-transformed rate Z [1, p. 291]:

$$Z_i = \log \frac{1000(Y_i + 1)}{n_i} \tag{2}$$

where Y_i is the count of cases in a census tract and n_i is its population. This is presented as field Z. We refer to this as leukemia incidence.

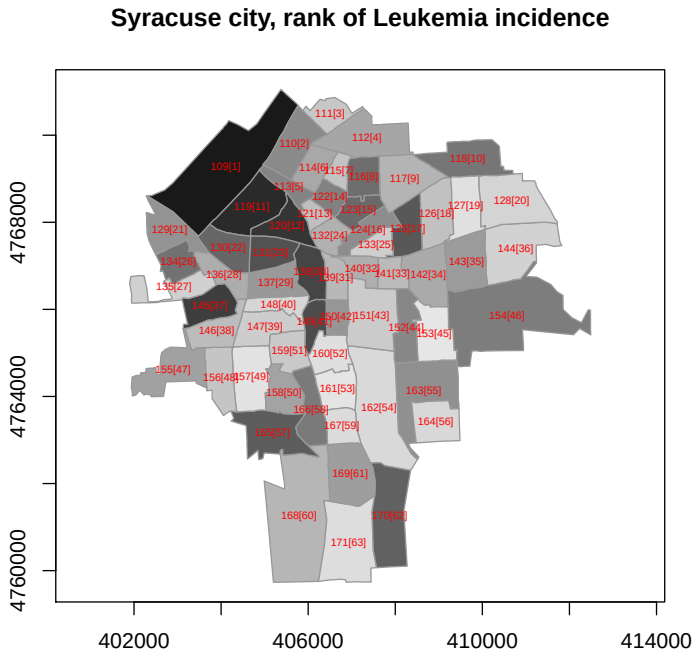
```
summary(Syr$Z)

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.44174 -0.57247 -0.06904  0.03775  0.43847  4.71053
```

TASK 19: Test the assumption that leukemia cases incidence is spatially independent (randomly distributed among census tracts). •

We first visualize the spatial relation with several grey-scale plots. The first is the *rank* of the leukemia incidence in each census tract, from

lowest (lightest shade of grey) to highest (darkest). The `grey.colors` function produces a colour ramp of gray shades with equal perceptive intervals.

```
n <- length(Syr$Z)
shade <- rank(Syr$Z)
plot(Syr, border="grey60",axes=TRUE,
     col=grey.colors(n, start=0.9, end=0.1, gamma=2.2)[shade])
title("Syracuse city, rank of Leukemia incidence")
text(coordinates(Syr), paste0(row.names(Syr),"[", 1:n, "]"),
     col="red", cex=0.5)
```



Another view is the *relative intensity* of incidence, shown with the same grey scale, but with the intensity of the grey proportional to the maximum proportion of cases.

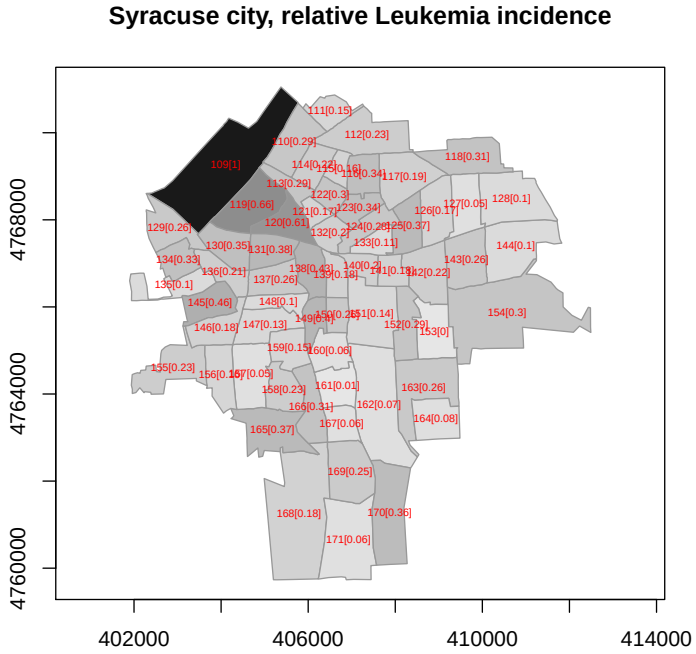
> **Note:** The `-min(Syr$Z)` in the numerator and denominator is to re-scale from zero for grey-shading. Note that field *Z* has some negative numbers; if all were positive the expression `Syr$Z/max(Syr$Z)` would also give a proper sequential gray scale.

> **Note:** The `pmax` "parallel maximum" function ensures that the lowest incidence uses the first grey in the scale, i.e., the lightest; if this were omitted the index would be 0 and give no corresponding colour.

The polygons are labelled with their row number and the relative risk, where 1 is the highest.

```
rel.risk  <- ((Syr$Z-min(Syr$Z))/(max(Syr$Z)-min(Syr$Z)))
shade <- pmax(ceiling(n*rel.risk), 1)
plot(Syr, border="grey60", axes=TRUE,
     col=grey.colors(n, start=0.9, end=0.1, gamma=2.2)[shade])
title("Syracuse city, relative Leukemia incidence")
text(coordinates(Syr),
```

```
     paste0(row.names(Syr), "[", round(rel.risk,2), "]"),
     cex=0.5, col="red")
```



Q9 : *Does leukemia incidence appear to be spatially autocorrelated?* *Jump to A9* •

Now we make the formal test, using the `moran.test` function. We accept the default `alternative="greater"` argument (so, no need to write it explicitly in the command), because we are not interested in determining whether the leukemia incidence is more spatially dispersed than by chance, only if it is more spatially clustered[10].

```
(moran.z <- moran.test(Syr$Z, Syr_lw_W))


Moran I test under randomisation

data:  Syr$Z
weights: Syr_lw_W

Moran I statistic standard deviate = 3.1394, p-value =
0.0008466
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
      0.207583627      -0.016129032       0.005078063
```

Q10 : *Is leukemia incidence provably spatially autocorrelated with this*

[10] The other choices are `"less"` and `"two-sided"`, see `help(moran.test)`

5.1.1 Effect of weights

The above results are for the default weighting: inversely by number of neighbours. Other reasonable weightings would be by inverse distance of the centroids, or by population, or by area, or by shared border length, depending on the process being modelled.

The `nb2listw` function has several options for the `style` optional argument Bivand et al. [1, §9.2.2]:

- W : explained above: inversely proportional to the number of neighbours;
- B : binary: 1 for a neighbour, 0 otherwise;
- C : globally standardized: inversely proportional to the total number of links; that is, all non-zero links get the same weight;
- U : C divided by the number of neighbours.

Row-standardisation (style `W`) favours observations with few neighbours, whereas the other styles favour observations with many neighbours. We can see this by comparing weights for few- and many-neighbour entries in the neighbour list:

```
Syr_lw_W <- nb2listw(Syr_nb)
Syr_lw_W$weights[[ix.min]]

[1] 1

Syr_lw_W$weights[[ix.max]]

[1] 0.1111111 0.1111111 0.1111111 0.1111111 0.1111111 0.1111111
[7] 0.1111111 0.1111111 0.1111111

Syr_lw_B <- nb2listw(Syr_nb, style="B")
Syr_lw_B$weights[[ix.min]]

[1] 1

Syr_lw_B$weights[[ix.max]]

[1] 1 1 1 1 1 1 1 1 1

Syr_lw_C <- nb2listw(Syr_nb, style="C")
Syr_lw_C$weights[[ix.min]]

[1] 0.1820809

Syr_lw_C$weights[[ix.max]]

[1] 0.1820809 0.1820809 0.1820809 0.1820809 0.1820809 0.1820809
[7] 0.1820809 0.1820809 0.1820809

Syr_lw_U <- nb2listw(Syr_nb, style="U")
Syr_lw_U$weights[[ix.min]]

[1] 0.002890173

Syr_lw_U$weights[[ix.max]]
```

```
[1] 0.002890173 0.002890173 0.002890173 0.002890173 0.002890173
[6] 0.002890173 0.002890173 0.002890173 0.002890173
```

We can also compute weights based on any criterion that seems appropriate to the process. One obvious possibility is inverse distance (perhaps to some power) of the area centroids: the further the centroids, the less influence. This is well-established for many processes originating at points, e.g., inverse-square light or sound intensity from point sources. It may be applicable to social processes as well.

Q11 : *Considering the leukemia incidence, why or why not would the inverse-distance weighting represent the underlying process?* *Jump to A11* •

TASK 20 : Compute a weights matrix based on inverse distance of the centroids. •

We use `nbdists` to calculate the distances for an object of class `nb`, from the centroid coordinates of the polygon object returned by the `coordinates` method, then `lapply` to invert the distances; this `lapply` takes an argument of class `function`, which in this case we build ourselves, since there is no "invert" function for vectors. Finally, we pass these to the weight-generating function `nb2listw` with the optional `glist` "general list" argument – this must be a list of lists, one for each area.

We illustrate the calculation with the first-listed polygon, 109, while applying it to the whole dataset with the `lapply` "list apply" function.

```
row.names(Syr@data[1,])

[1] "109"

Syr_nb[[1]]

[1]  2  5 11 21 22

dsts <- nbdists(Syr_nb, coordinates(Syr))
dsts[1]

[[1]]
[1] 1656.873 1514.638 1098.564 1944.120 1871.600

idw <- lapply(dsts, function(x) 1/(x/1000))
idw[1]

[[1]]
[1] 0.6035465 0.6602238 0.9102789 0.5143715 0.5343021

Syr_lw_idwB <- nb2listw(Syr_nb, glist=idw, style="B")
Syr_lw_idwB$weights[[1]]

[1] 0.6035465 0.6602238 0.9102789 0.5143715 0.5343021
```

Here is the summary of the weights:

```
summary(unlist(Syr_lw_idwB$weights))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.3886  0.7374  0.9259  0.9963  1.1910  2.5274
```

And here is the summary of the sums of weights per polygon:

```
summary(sapply(Syr_lw_idwB$weights, sum))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.304   3.986   5.869   5.471   6.737   9.435
```

There is a wide range of total weights assigned to a polygon, very unlike the "W" style weights.

TASK 21 : Re-compute Moran's *I* with this weighting. •

Again we use the `moran.test` function, with the new weights matrices:

```
(moran.z.idwB <- moran.test(Syr$Z, Syr_lw_idwB))


Moran I test under randomisation

data:  Syr$Z
weights: Syr_lw_idwB

Moran I statistic standard deviate = 2.9147, p-value = 0.00178
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
      0.195554357      -0.016129032       0.005274558

# (moran.pctage65p.idwB <- moran.test(Syr$PCTAGE65P, Syr_lw_idwB))
```

Q12 : *How did the probabilities of Type I error to reject the null hypothesis of no association change with this weighting?* *Jump to A12*
•

TASK 22 : **Optional**: Compare the weights matrices of the different weighting styles. •

```
tmp <- build.wts.matrix(Syr_lw_W)
round(tmp[1:9,1:9],4)

       109    110    111    112 113    114    115    116    117
109 0.0000 0.2000 0.0000 0.0000 0.2 0.0000 0.0000 0.0000 0.0000
110 0.2000 0.0000 0.2000 0.2000 0.2 0.2000 0.0000 0.0000 0.0000
111 0.0000 0.5000 0.0000 0.5000 0.0 0.0000 0.0000 0.0000 0.0000
112 0.0000 0.1429 0.1429 0.0000 0.0 0.1429 0.1429 0.1429 0.1429
113 0.1429 0.1429 0.0000 0.0000 0.0 0.1429 0.0000 0.0000 0.0000
114 0.0000 0.2000 0.0000 0.2000 0.2 0.0000 0.2000 0.0000 0.0000
115 0.0000 0.0000 0.0000 0.2500 0.0 0.2500 0.0000 0.2500 0.0000
116 0.0000 0.0000 0.0000 0.2000 0.0 0.0000 0.2000 0.0000 0.2000
117 0.0000 0.0000 0.0000 0.1429 0.0 0.0000 0.0000 0.1429 0.0000

tmp <- build.wts.matrix(Syr_lw_B)
round(tmp[1:9,1:9], 4)

    109 110 111 112 113 114 115 116 117
109   0   1   0   0   1   0   0   0   0
110   1   0   1   1   1   1   0   0   0
111   0   1   0   1   0   0   0   0   0
```

```
112   0   1   1   0   0   1   1   1   1
113   1   1   0   0   0   1   0   0   0
114   0   1   0   1   1   0   1   0   0
115   0   0   0   1   0   1   0   1   0
116   0   0   0   1   0   0   1   0   1
117   0   0   0   1   0   0   0   1   0
```

```
tmp <- build.wts.matrix(Syr_lw_C)
round(tmp[1:9,1:9], 4)
```

```
       109    110    111    112    113    114    115    116    117
109 0.0000 0.1821 0.0000 0.0000 0.1821 0.0000 0.0000 0.0000 0.0000
110 0.1821 0.0000 0.1821 0.1821 0.1821 0.1821 0.0000 0.0000 0.0000
111 0.0000 0.1821 0.0000 0.1821 0.0000 0.0000 0.0000 0.0000 0.0000
112 0.0000 0.1821 0.1821 0.0000 0.0000 0.1821 0.1821 0.1821 0.1821
113 0.1821 0.1821 0.0000 0.0000 0.0000 0.1821 0.0000 0.0000 0.0000
114 0.0000 0.1821 0.0000 0.1821 0.1821 0.0000 0.1821 0.0000 0.0000
115 0.0000 0.0000 0.0000 0.1821 0.0000 0.1821 0.0000 0.1821 0.0000
116 0.0000 0.0000 0.0000 0.1821 0.0000 0.0000 0.1821 0.0000 0.1821
117 0.0000 0.0000 0.0000 0.1821 0.0000 0.0000 0.0000 0.1821 0.0000
```

```
tmp <- build.wts.matrix(Syr_lw_U)
round(tmp[1:9,1:9], 4)
```

```
       109    110    111    112    113    114    115    116    117
109 0.0000 0.0029 0.0000 0.0000 0.0029 0.0000 0.0000 0.0000 0.0000
110 0.0029 0.0000 0.0029 0.0029 0.0029 0.0029 0.0000 0.0000 0.0000
111 0.0000 0.0029 0.0000 0.0029 0.0000 0.0000 0.0000 0.0000 0.0000
112 0.0000 0.0029 0.0029 0.0000 0.0000 0.0029 0.0029 0.0029 0.0029
113 0.0029 0.0029 0.0000 0.0000 0.0000 0.0029 0.0000 0.0000 0.0000
114 0.0000 0.0029 0.0000 0.0029 0.0029 0.0000 0.0029 0.0000 0.0000
115 0.0000 0.0000 0.0000 0.0029 0.0000 0.0029 0.0000 0.0029 0.0000
116 0.0000 0.0000 0.0000 0.0029 0.0000 0.0000 0.0029 0.0000 0.0029
117 0.0000 0.0000 0.0000 0.0029 0.0000 0.0000 0.0000 0.0029 0.0000
```

```
tmp <- build.wts.matrix(Syr_lw_idwB)
round(tmp[1:9,1:9], 4)
```

```
       109    110    111    112    113    114    115    116    117
109 0.0000 0.6035 0.0000 0.0000 0.6602 0.0000 0.0000 0.0000 0.0000
110 0.6035 0.0000 0.9265 0.5963 1.0111 1.4139 0.0000 0.0000 0.0000
111 0.0000 0.9265 0.0000 0.9858 0.0000 0.0000 0.0000 0.0000 0.0000
112 0.0000 0.5963 0.9858 0.0000 0.0000 0.7191 1.0020 1.1118 0.7829
113 0.6602 1.0111 0.0000 0.0000 0.0000 1.3676 0.0000 0.0000 0.0000
114 0.0000 1.4139 0.0000 0.7191 1.3676 0.0000 1.7476 0.0000 0.0000
115 0.0000 0.0000 0.0000 1.0020 0.0000 1.7476 0.0000 1.7162 0.0000
116 0.0000 0.0000 0.0000 1.1118 0.0000 0.0000 1.7162 0.0000 1.0592
117 0.0000 0.0000 0.0000 0.7829 0.0000 0.0000 0.0000 1.0592 0.0000
```

5.2 Local tests

Global tests for spatial autocorrelation are aggregated from local relationships (see the formula for Moran's I). This local information can be aggregated locally, rather than over the whole map, to detect "hotspots" where there is strong autocorrelation of high values, and "cold spots" where there is strong autocorrelation of low values. In geostatistical terms, the spatial process may *not* be stationary.

5.2.1 Local Moran's I

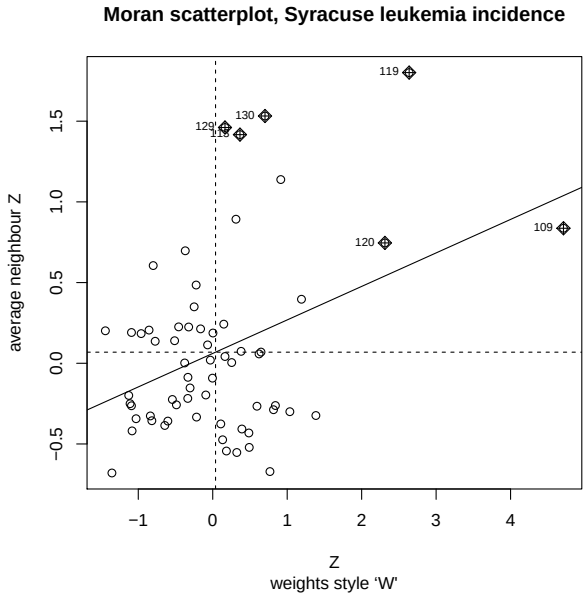
One good way to visualize the relation between the global and local measures is to plot a so-called **Moran scatterplot**: the target variable on the

x-axis, and the (spatially-weighted) sum of neighbouring values on the y-axis; these are called the **spatially lagged** values.

TASK 23 : Plot the local Moran's I scatterplot for the Syracuse leukemia incidence, with the default `W` weighting. •

The `moran.plot` function takes two arguments: the vector of values and the neighbour list with weights:

```
mp <- moran.plot(Syr$Z, Syr_lw_W, xlab="Z",
                 ylab="average neighbour Z")
title(main="Moran scatterplot, Syracuse leukemia incidence",
      sub="weights style `W'")
```



The regression line is the global Moran's *I*. Points with high influence are identified by a special symbol and their row number in the original (8-county) dataset.

TASK 24 : Identify the high-influence areas; find their neighbour relations. •

The `is.inf` "is influential" field in the list resulting from the `moran.plot` function gives six measures of each observation's influence on the plotted regression line; these are computed by the `influence.measures` method, which is often applied to linear models. The `any` logical function returns `TRUE` if any of the six measures for an observation are `TRUE`; we use the `apply` function to apply this function row-wise (as shown by the `MARGIN=1` argument), resulting in an array of logical values: `TRUE` if the observation is influential by any measure.

```
ix <- which(infl <- apply(mp$is.inf, MARGIN=1, any))
cbind(Syr@data[ix,c("AREAKEY","Z")],ix)

        AREAKEY       Z ix
```

```
109 36067000100 4.71053  1
113 36067000500 0.36591  5
119 36067001100 2.63806 11
120 36067001200 2.31264 12
129 36067002000 0.16464 21
130 36067002100 0.70212 22
```

```
Syr_lw_W$neighbours[ix]
```

```
[[1]]
[1]  2  5 11 21 22

[[2]]
[1]  1  2  6 11 12 13 14

[[3]]
[1]  1  5 12 22 23

[[4]]
[1]  5 11 13 22 23 24 30

[[5]]
[1]  1 22 26 28

[[6]]
[1]  1 11 12 21 23 28 29
```

```
Syr@data[Syr_lw_W$neighbours[ix][[1]],c("AREAKEY","Z")]
```

```
        AREAKEY       Z
110 36067000200 0.31195
113 36067000500 0.36591
119 36067001100 2.63806
129 36067002000 0.16464
130 36067002100 0.70212
```

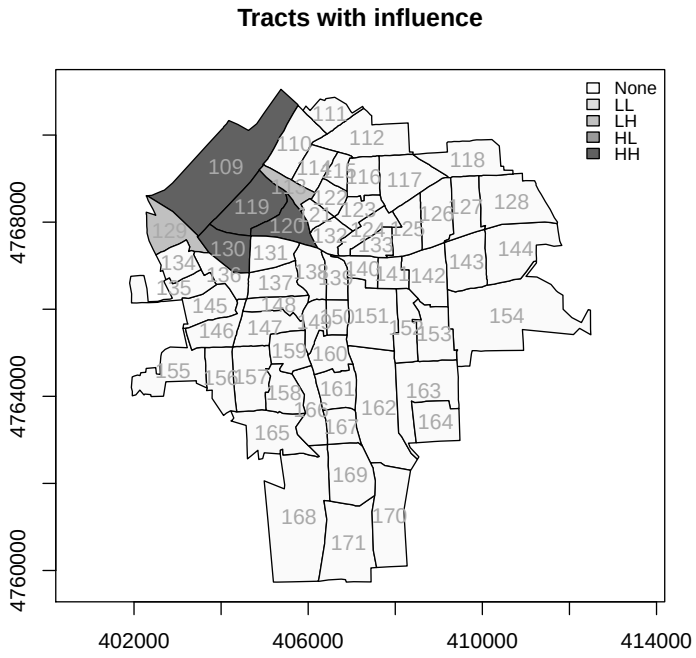
Q13 : *Which areas strongly influenced the global Moran's* I *line? Are these high-influence area neighbours?* *Jump to A13* •

TASK 25 : Plot these as shaded polygons, with a four-way legend: no influence, high proportion with low proportion neighbours ("HL"), the reverse ("LH"), and both high ("HH"). We define the break between "low" and "high" as the third quartile. The `cut` function slices the incidence and lagged incidence into high and low, and then the `interaction` function makes a crossed factor of these two. •

> **Note:** The `lag` method applied to an object of class `listw`, i.e., a neighbour weight list, specialized to the `spdep` function `lag.listw`. This takes the neighbour weight list and an attribute (here, the leukemia incidence) and returns an attribute vector, but with the original attribute values replaced with those from the weighted neighbours. It is the function used to position points on the local Moran's *I* plot shown just above. Here we use it to find the relation between leukemia incidence in one district and its neighbours.

```
x <- Syr$Z
lhx <- cut(x, breaks=c(min(x), quantile(x,.75), max(x)),
           labels=c("L", "H"), include.lowest=TRUE)
wx <- lag(Syr_lw_W, Syr$Z)
print(paste("Z in district ", row.names(Syr)[1], ": ",
```

```
            round(Syr$Z[1],3), sep=""))

[1] "Z in district 109: 4.711"

print(paste("Weighted average Z of district ",
            row.names(Syr)[1], " neighbours: ",
            round(wx[1],3), sep=""))

[1] "Weighted average Z of district 109 neighbours: 0.837"

lhwx <- cut(wx, breaks=c(min(wx), quantile(wx,.75), max(wx)),
            labels=c("L", "H"), include.lowest=TRUE)
## mean(wx)
lhlh <- interaction(lhx, lhwx, infl, drop=TRUE)
cols <- rep(1, length(lhlh))
cols[lhlh == "L.L.TRUE"] <- 2
cols[lhlh == "L.H.TRUE"] <- 3
cols[lhlh == "H.L.TRUE"] <- 4
cols[lhlh == "H.H.TRUE"] <- 5
plot(Syr, col=grey.colors(5, 0.98, 0.38, 2.2)[cols], axes=T)
text(coordinates(Syr), row.names(Syr), col="darkgray")
legend("topright", legend=c("None", "LL", "LH", "HL", "HH"),
       fill=grey.colors(5, 0.98, 0.38, 2.2), bty="n",
       cex=0.8, y.intersp=0.8)
title("Tracts with influence")
```



Q14 : *How does this figure change your interpretation of the spatial autocorrelation of leukemia incidence as expressed by the global Moran's* I *and corresponding figure?* *Jump to A14* •

TASK 26 : Compute the local Moran's *I* for the leukemia incidence. •

Local Moran's I is defined for each area i as:

$$I_i = \frac{(y_i - \bar{y}) \cdot \sum_j (y_j - \bar{y})}{1/n \cdot \sum_i (y_i - \bar{y})^2} \tag{3}$$

where the symbols are defined as in Equation 1. The two expressions in the numerator define a point in the Moran scatterplot, above. The denominator standardizes the local Moran's I so that $\sum_i I_i = I$. Again, we are looking for the probability that rejecting the null hypothesis of no spatial autocorrelation would be a Type I error.

This test is computed by the `localmoran` function.

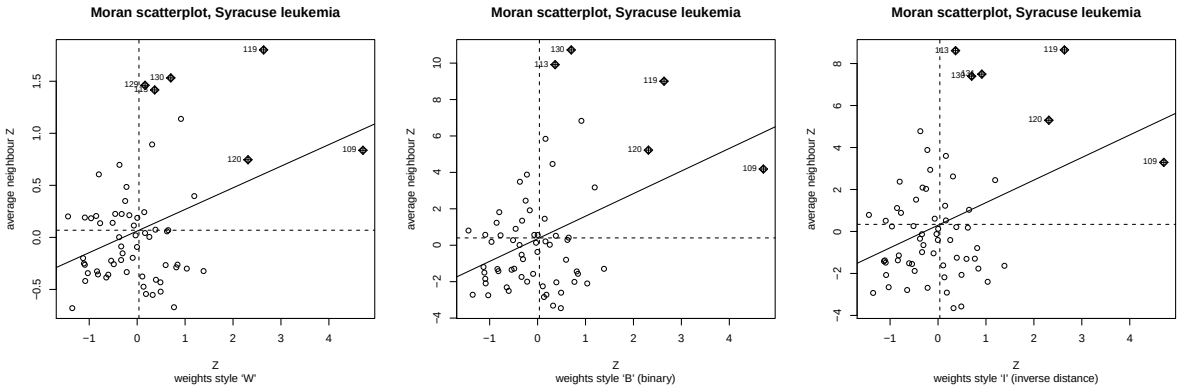
```
lm1 <- localmoran(Syr$Z, Syr_lw_W)
class(lm1)

[1] "localmoran" "matrix"

summary(lm1)

       Ii                E.Ii              Var.Ii
 Min.   :-0.521426   Min.   :-0.01613   Min.   :0.08548
 1st Qu.:-0.132169   1st Qu.:-0.01613   1st Qu.:0.12377
 Median : 0.005406   Median :-0.01613   Median :0.13421
 Mean   : 0.207584   Mean   :-0.01613   Mean   :0.17176
 3rd Qu.: 0.266861   3rd Qu.:-0.01613   3rd Qu.:0.18538
 Max.   : 4.617088   Max.   :-0.01613   Max.   :0.86517
      Z.Ii             Pr(z > 0)
 Min.   :-1.04174   Min.   :0.0000
 1st Qu.:-0.25548   1st Qu.:0.2236
 Median : 0.05174   Median :0.4794
 Mean   : 0.60598   Mean   :0.4322
 3rd Qu.: 0.76014   3rd Qu.:0.6008
 Max.   :11.46006   Max.   :0.8512

ix <- which(lm1[,"Pr(z > 0)"] < 0.05)
cbind(Syr@data[ix,c("AREAKEY","POP8","Z")],ix)

        AREAKEY POP8        Z ix
109 36067000100    9  4.71053  1
119 36067001100  143  2.63806 11
120 36067001200   99  2.31264 12
130 36067002100 1997  0.70212 22
131 36067002200 1211  0.91381 23
161 36067005400 4144 -1.35400 53
```

Q15 : *Is there evidence of local clustering? Could you interpret this from the Moran scatterplot?* *Jump to A15* •

Task 27 : **Optional**: Repeat the above plots and analysis for other weighting styles. •

We have the already computed the required weight matrices (§5.1.1), so we just use these in the call to `moran.plot`. For example, using binary and inverse-distance weightings and comparing with style `W`:

```
par(mfrow=c(1,3))
mp <- moran.plot(Syr$Z, Syr_lw_W, xlab="Z",
                 ylab="average neighbour Z")
title(main="Moran scatterplot, Syracuse leukemia",
      sub="weights style `W'")
mp <- moran.plot(Syr$Z, Syr_lw_B, xlab="Z",
                 ylab="average neighbour Z")
title(main="Moran scatterplot, Syracuse leukemia",
      sub="weights style `B' (binary)")
mp <- moran.plot(Syr$Z, Syr_lw_idwB, xlab="Z",
                 ylab="average neighbour Z")
title(main="Moran scatterplot, Syracuse leukemia",
      sub="weights style `I' (inverse distance)")
par(mfrow=c(1,1))
```



Now the overall test and the influential observations:

```
##
moran.test(Syr$Z, Syr_lw_W)


Moran I test under randomisation

data:  Syr$Z
weights: Syr_lw_W

Moran I statistic standard deviate = 3.1394, p-value =
0.0008466
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
      0.207583627      -0.016129032       0.005078063

lm1 <- localmoran(Syr$Z, Syr_lw_W)
summary(lm1[,"Ii"])

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.521426 -0.132169  0.005406  0.207584  0.266861  4.617088

(ix <- which(lm1[,"Pr(z > 0)"] < 0.05))

109 119 120 130 131 161
  1  11  12  22  23  53

##
moran.test(Syr$Z, Syr_lw_B)


Moran I test under randomisation
```

```
data:  Syr$Z
weights: Syr_lw_B

Moran I statistic standard deviate = 3.5397, p-value =
0.0002003
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
      0.224450751      -0.016129032       0.004619419

lm1 <- localmoran(Syr$Z, Syr_lw_B)
summary(lm1[,"Ii"])

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-2.37565 -0.51602  0.02162  1.23270  1.31989 23.08544

(ix <- which(lm1[,"Pr(z > 0)"] < 0.05))

109 119 120 130 131 161
  1  11  12  22  23  53

##
moran.test(Syr$Z, Syr_lw_idwB)


Moran I test under randomisation

data:  Syr$Z
weights: Syr_lw_idwB

Moran I statistic standard deviate = 2.9147, p-value = 0.00178
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
      0.195554357      -0.016129032       0.005274558

lm1 <- localmoran(Syr$Z, Syr_lw_idwB)
summary(lm1[,"Ii"])

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-2.7109 -0.5859  0.0236  1.0700  1.1088 22.1691

(ix <- which(lm1[,"Pr(z > 0)"] < 0.05))

109 119 120 130 131 161 162
  1  11  12  22  23  53  54

##
```

Q16 : *What are the principal differences between the global Moran's I, the local Moran's I plots, and the influential observations, for these three neighbour weightings?* *Jump to A16* •

5.2.2 Getis-Ord local G statistics *

Another way to visualize "hot" and "cold" spots is local association statistics developed by Ord and Getis [4]. These are symbolized as G_i and G_i^*; the subscript i emphasizes that they are computed separately for each area. These statistics do not attempt to characterize overall spatial dependency; rather, they help identify local areas where there may be dependency. In this it is similar to local Moran's I.

> "These statistics are especially useful in cases where global statistics may fail to alert the researcher to significant pockets of clustering." – [4, p. 287]

There are two variants: G_i and G_i^*, where the 'starred' variant includes the self-weights w_{ii} of each target polygon The first variant G_i shows whether an area is within a surrounding hot or cold spot; the second variant G_i^* shows whether the area itself is part of such a spot.

The G_i^* statistic is:

$$G_i^* = \frac{\sum_{j=1}^{n} w_{i,j}x_j - \bar{x}\sum_{j=1}^{n} w_{i,j}}{s \cdot ([n\sum_{j=1}^{n} w_{i,j}^2 - (\sum_{j=1}^{n} w_{i,j})^2]/[n-1])^{1/2}} \tag{4}$$

It may be interpreted as a Z-score, i.e., a normal variate, where 0 is the global mean of the target variable. Positive Z-scores show clusters of high values, negative Z-scores show clusters of low values.

> **Note:** In Getis and Ord's original formulation G_i depends on a distance band; this more general formulation includes that special case, because the weights matrix **W** can be constructed by distance or by steps to neighbours.

Task 28 : Compute and summarize the G_i statistics for the Syracuse leukemia incidence, using the default neighbour weighting. •

The weighting matrices constructed in §4 did *not* include self-weights, so the statistic computed by the `localG` function, using these weights, is G_i:

```
summary(gi <- localG(Syr$Z, Syr_lw_W))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.5352 -0.8659 -0.1081  0.0927  0.4274  4.4059
```

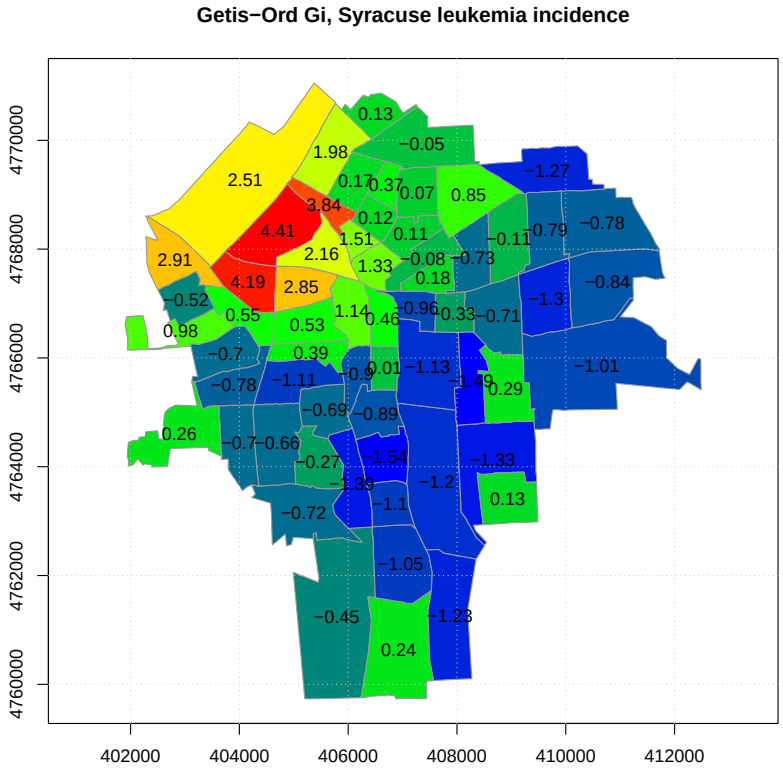
Task 29 : Plot these as coloured polygons, with the red correspond to the positive clusters and blue to the negative ones. •

A colour ramp can be constructed with the `colorRampPalette` function, specifying a range of colours, which will be interpolated into a ramp. We can then select the correct shade out of the ramp for each polygon.

> **Note:** Note the +1, otherwise there would be a shade 0.

```
shade <- as.numeric(round(n*((gi-min(gi))/
                             (max(gi)-min(gi)))))+1
sort(shade)

 [1]  1  1  3  3  4  4  4  5  5  6  6  6  7  7  8  8  8  9  9  9 10 10
[23] 10 10 10 10 10 12 12 14 14 16 16 17 17 18 18 19 19 19 19 19 20 20
[45] 20 21 21 22 23 23 26 28 29 31 33 38 40 44 48 48 58 62 64

colfunc <- colorRampPalette(c("blue", "green", "yellow", "red"))
ramp <- colfunc(n+1)
```

```
plot(Syr, border="grey60", axes=TRUE,
     col=ramp[shade],
     main="Getis-Ord Gi, Syracuse leukemia incidence")
grid()
text(coordinates(Syr), as.character(round(gi,2)), col="black")
```



Q17 : *Where are the clusters? How does this map compare to the local Moran's I map?* *Jump to A17* •

TASK 30 : Compute and summarize the G_i^* statistics for the Syracuse leukemia incidence, using the default neighbour weighting. •

To compute G_i^* we need to create a weights matrix including each target area with a weight. We first add each area's index to its own neighbour list, and then convert these to weights using the `nb2listw` function:

```
Syr_nbi <- Syr_nb
## add the index its own list
for (i in 1:length(Syr_nb)) {
    Syr_nbi[[i]] <- sort(c(Syr_nbi[[i]], i))
}
## convert to weights
Syr_lw_Wi <- nb2listw(Syr_nbi)
print(Syr_lw_W)

Characteristics of weights list object:
Neighbour list object:
Number of regions: 63
```

```
Number of nonzero links: 346
Percentage nonzero weights: 8.717561
Average number of links: 5.492063

Weights style: W
Weights constants summary:
   n   nn S0       S1      S2
W 63 3969 63 24.78291 258.564

print(Syr_lw_W$weights[[1]])

[1] 0.2 0.2 0.2 0.2 0.2

print(Syr_lw_Wi)

Characteristics of weights list object:
Neighbour list object:
Number of regions: 63
Number of nonzero links: 409
Percentage nonzero weights: 10.30486
Average number of links: 6.492063

Weights style: W
Weights constants summary:
   n   nn S0       S1       S2
W 63 3969 63 20.52886 254.7581

print(Syr_lw_Wi$weights[[1]])

[1] 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667
```

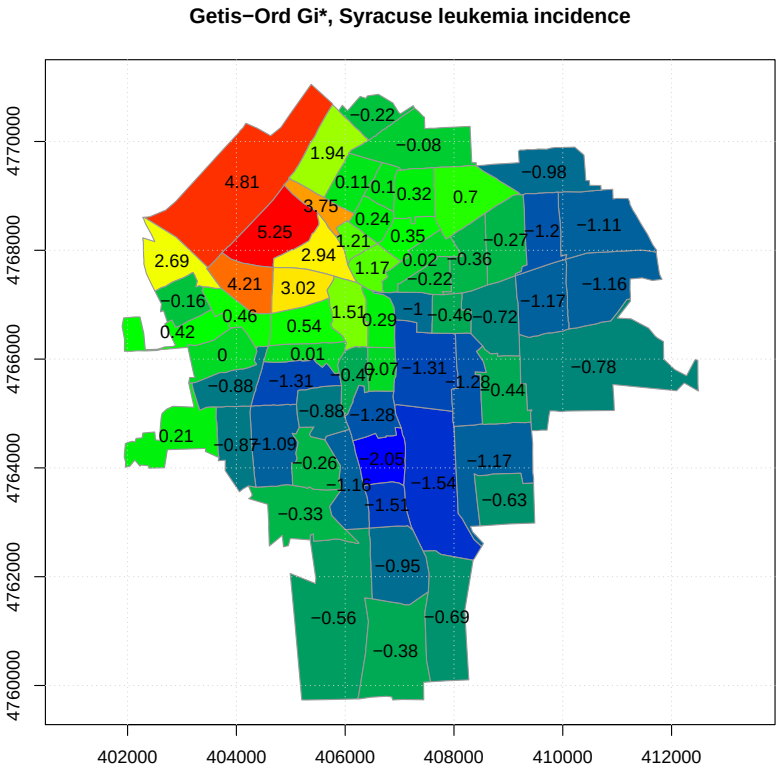
Q18 : *What is the difference between the weights list with and without including the target area?* *Jump to A18* •

Now we can compute and plot G_i^*:

```
summary(gi.star <- localG(Syr$Z, Syr_lw_Wi))

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-2.04689 -0.96436 -0.26810  0.08543  0.33077  5.24567

summary(gi)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.5352 -0.8659 -0.1081  0.0927  0.4274  4.4059

summary(gi.star-gi)

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.763982 -0.281350 -0.059567 -0.007265  0.190800  2.309034
```

Q19 : *What are the differences between G_i^* (including the target area's value in the index) and G_i (not)?* *Jump to A19* •

```
shade <- as.numeric(round(n*((gi.star-min(gi.star))/
                             (max(gi.star)-min(gi.star)))))+1
plot(Syr, border="grey60", axes=TRUE,
     col=ramp[shade],
     main="Getis-Ord Gi*, Syracuse leukemia incidence")
grid()
text(coordinates(Syr), as.character(round(gi.star,2)), col="black")
```



Q20 : *What are the differences between the G_i and G_i^* maps?* *Jump to A20* •

6 Spatial models

Supplementary reading:

- Bivand et al. [1, §9.4] *Fitting models of areal data*

> "Finding spatial autocorrelation is not a goal in itself, be it local or global, but rather just one step in a process leading to a proper model."
>
> –Bivand et al. [1, §9.4]

What does it all mean? What is (are) the process(es) which give rise to the observations? Apparent autocorrelation, such as found in the previous sections, may instead be caused by some underlying factor(s), i.e., the assumed **zero-mean** model is not correct. This is called **model misspecification**. It can arise from a poorly-distributed response variable (e.g., unequal variance across the map) or a wrong (or missing) functional form from (partially) deterministic factors that are also spatially-

distributed.

Q21 : *What could be some spatially-distributed causes of leukemia?* *Jump to A21* •

The aim is to return to a zero-mean model, by removing any feature-space predictors, in this case other variables collected per census tract. Any kind of model can be used; we illustrate this with a linear model:

$$\mathbf{y} = \mathbf{X}^T\beta + \varepsilon \tag{5}$$

where Y is the response vector (one element per area), X is the model matrix, β are the model coefficients (fitted from the data), and ε is the random error vector, for now considered to be identically normally and independently distributed, with zero mean and variance V. We do not yet consider spatial autocorrelation of the residuals.

The database has three possible co-variables (predictors): `PEXPOSURE`, `PCTAGE65P`, and `PCTOWNHOME`; see the beginning of this §5 for details. We are most interested in whether TCE exposure is a risk factor for cancer, if so we should promote cleanup of TCE sites. But cancers may be positively associated with old age, which implies long-term exposure to any environmental factor as well as life style, and negatively with home ownership, which implies a higher economic level and perhaps a healthier lifestyle.

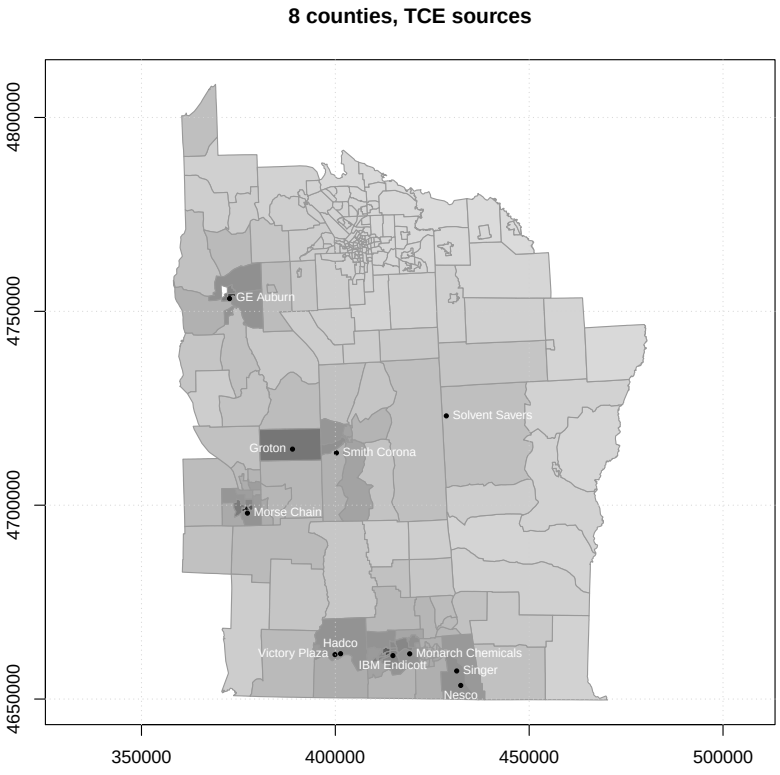
We return to the full 8-county area, because the TCE sources are spread throughout; Syracuse city is too small to have substantially different distance to TCE sources.

Task 31 : Import the map of TCE sources and display them on the 8-county census tracts, shaded by the exposure potential. •

Again we use `readOGR` to import the points shapefile:

```
TCE <- readOGR("./NY_data", "TCE")
```

```
shade <- round(n*NY8$PEXPOSURE/max(NY8$PEXPOSURE))
plot(NY8, border="grey60", col=grey.colors(n, 0.9, 0.1, 2.2)[shade],
     axes=TRUE, asp=1)
points(coordinates(TCE), cex=.5, pch=19)
text(coordinates(TCE), labels=as.character(TCE$name), cex=0.7,
 font=1, pos=c(4,1,4,1,4,4,4,2,3,4,2), offset=0.3, col="white")
grid()
title("8 counties, TCE sources")
```



We see that Syracuse is far from these sources; however Syracuse is an industrial city, so there may be exposure to other chemicals.

TASK 32 : Model the 8-county leukemia incidences by an additive model of three predictors: TCE exposure, proportion of population older than 65 years, and proportion of home ownership. •

To set a baseline, we use the Ordinary Least Squares (OLS) estimate provided by the standard `lm` function.

```
m.z.ppp <- lm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=NY8@data)
summary(m.z.ppp)


Call:
lm(formula = Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data = NY8@data)

Residuals:
    Min      1Q  Median      3Q     Max
-1.7417 -0.3957 -0.0326  0.3353  4.1398

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.51728    0.15856  -3.262  0.00124 **
PEXPOSURE    0.04884    0.03506   1.393  0.16480
PCTAGE65P    3.95089    0.60550   6.525 3.22e-10 ***
PCTOWNHOME  -0.56004    0.17031  -3.288  0.00114 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6571 on 277 degrees of freedom
```

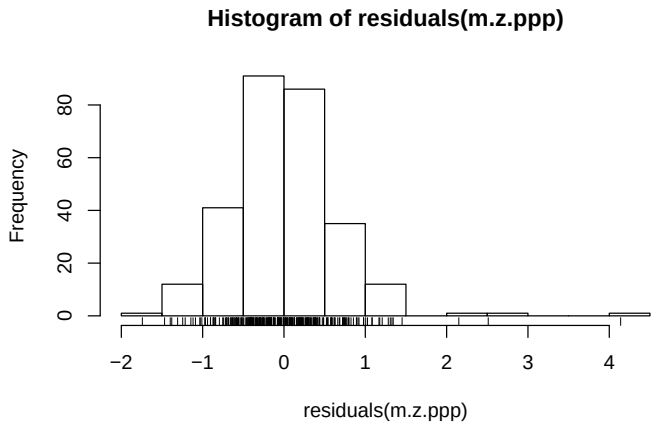
```
Multiple R-squared:  0.1932,Adjusted R-squared:  0.1844
F-statistic:  22.1 on 3 and 277 DF,  p-value: 7.306e-13
```

Q22 : *How much of the variability between census tracts in leukemia incidence is explained by this model? Which factors are significant in the linear model? Was the zero-model assumed in previous sections valid? Can you interpret these as possible processes?* *Jump to A22* •

TASK 33 : Plot a histogram of the residuals; identify the extreme outlier and display its database entry. •

The `residuals` function extracts a vector of residuals from a `lm` object:

```
hist(residuals(m.z.ppp))
rug(residuals(m.z.ppp))
ix <- which.max(residuals(m.z.ppp))
NY8@data[ix,]

         AREANAME     AREAKEY         X       Y POP8 TRACTCAS PROPCAS
109 Syracuse city 36067000100 -15.3264 40.5083    9        0       0
    PCTOWNHOME PCTAGE65P       Z  AVGIDIST PEXPOSURE   Cases       Xm
109        0.5 0.3333333 4.71053 0.0284377  1.045131 0.00014 -15326.4
         Ym   Xshift  Yshift
109 40508.3 403995.2 4769363
```



This is our old friend, tract 109.

Q23 : *Why is this residual so extreme?* *Jump to A23* •

Now we can check this model for spatial correlation of the residuals, with the `lm.morantest` function. This requires a model (which we just built) and a weights list (see §4).

TASK 34 : Build a weights list, from the default (queen's) neighbour list, using binary weights. Apply the Moran's test of the residuals, using these weights. •

```
NY8listwB <- nb2listw(NY8_nb, style = "B")
(m.z.ppp.moran.test <- lm.morantest(m.z.ppp, NY8listwB))


Global Moran I for regression residuals

data:
model: lm(formula = Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
data = NY8@data)
weights: NY8listwB

Moran I statistic standard deviate = 2.638, p-value = 0.004169
alternative hypothesis: greater
sample estimates:
Observed Moran I      Expectation         Variance
     0.083090278     -0.009891282      0.001242320
```

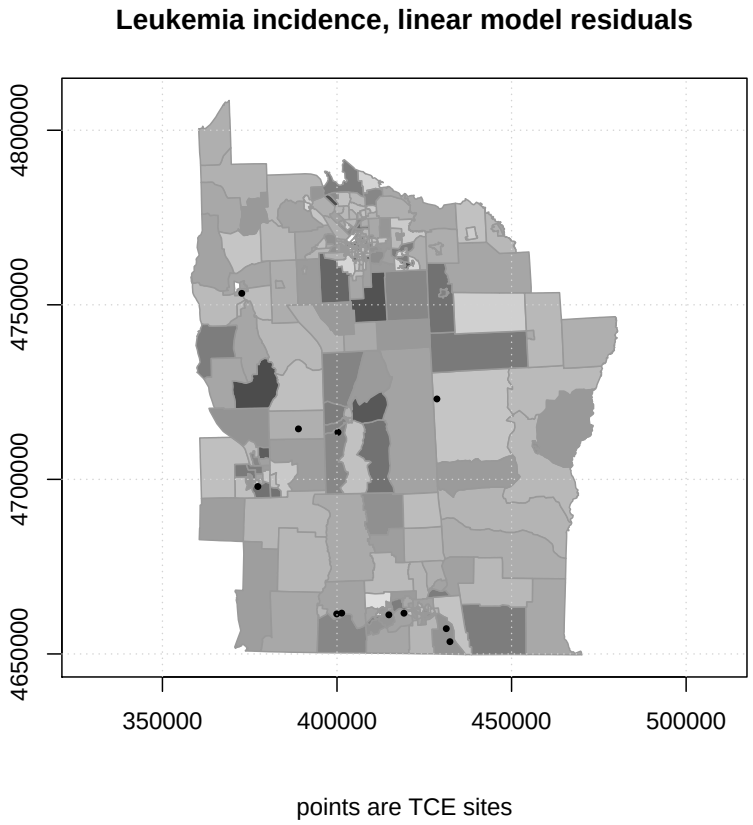
Q24 : *Is there evidence that the residuals are spatially correlated?* *Jump to A24* •

TASK 35 : Visualize the regression residuals as a map of the census tracts, with the residuals represented by a grey scale. •

We add the residuals to the data frame (i.e., attribute table) of the polygons, then map these. Residuals are defined as (actual - modelled), so darker greys are larger under-predictions.

> **Note:** For a better visualization, we set the upper end of the scale at the 99% quantile, to exclude tract 109 from the stretch.

```
NY8$lmresid <- residuals(m.z.ppp)
summary(NY8$lmresid)

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.74174 -0.39572 -0.03258  0.00000  0.33527  4.13982

shade <- round(n*(NY8$lmresid-min(NY8$lmresid))/
                  (quantile(NY8$lmresid,.99)-min(NY8$lmresid)))
plot(NY8, border="grey60",
     col=grey.colors(n, 0.9, 0.1, 2.2)[shade], axes=TRUE, asp=1)
points(coordinates(TCE), cex=.5, pch=19)
grid()
title("Leukemia incidence, linear model residuals",
      "points are TCE sites")
```



Q25 : *Is there visual evidence that the residuals are spatially correlated? What does this suggest about our model?* *Jump to A25* •

7 Autoregressive Models

Supplementary reading:

- Bivand et al. [1, §9.4.1]: *Spatial statistics approaches*

In the linear model of the previous section we did not account for spatial autocorrelation of the residuals, which indeed was present. So the linear model violated one of its assumptions, i.e., independent residuals. To account for this, we should refit the model as an **autoregressive** mode which accounts for spatial autocorrelation.

There are three main forms of Simultaneous Autoregressive (SAR) models, with different explanations on the source of the autocorrelation of the OLS residuals:

spatial error : These imply that there are underlying spatially-correlated predictors which are not included in the linear model predictor list. Either we don't suspect they are present, or else we have not measured them.

For example, leukemia incidence has been suspected to have some relation to extremely low frequency electromagnetic energy. This would be a predictor similar to TCE exposure, which is included in our model, and would be similar spatially-concentrated. This suspected predictor is not included in our model, since we have no information on its sources in the study area.

spatial lag : These imply that the response variable is influenced by the same response variable in neighbouring areas.

In this example it would mean that leukemia incidence in one area is influenced by incidence in nearby areas. This makes sense for infectious diseases, for example, feline leukemia, which can be transmitted between cats by saliva or nasal secretions, and the local nature of cat-to-cat interactions suggests that such "spillover" would occur between neighbouring areas. However, human leukemia is not known to be infectious, so this model is difficult to justify here.

spatial Durbin : These imply that the response variable is influenced both by the target variable *and* by the feature-space predictors in the model specification not only within each area separately, but also by the same predictors from neighbouring areas.

In this example it's hard to imagine how home ownership or proportion of older people or TCE exposure in *neighbouring* tracts could affect leukemia in a target tract.

> **Note:** "Durbin" models are named for the British statistician James Durbin, following his formulation of autoregressive time series models [2].

We now see how these model specifications can be applied to our example.

7.1 Spatial Error SAR model

We start with the spatial error SAR model.

The concept here is that the linear model residuals are no longer considered independent, instead they are modelled by a regression on the residuals from adjacent areas:

$$e_i = \sum_{i=1}^{m} b_{ij} e_i + \varepsilon_i \tag{6}$$

where the ε_i are the independent $\mathcal{N} \sim (0, 1)$ errors; these have a diagonal covariance matrix (so no interactions) Σ_ε with elements $\sigma^2_{e_i}$, which are often considered identical. The b values express the spatial dependence; note that $b_{ii} \doteq 0$ – an area can not depends on itself.

This formulation then adds a term to the linear model, to account for the autoregression of Equation 6:

$$\mathbf{y} = \mathbf{X}^T\beta + (\lambda\mathbf{W})(\mathbf{y} - \mathbf{X}^T\beta) + \varepsilon \tag{7}$$

where λ is the strength of this autoregression term and W is a weights matrix. This is the same kind of list, of class `listw` we've created in §4. This depends on the neighbour list and a weighting model; we have already build one using binary weights as `NY8listwB`. The autoregression term $(\lambda\mathbf{W})$ multiples the linear model residuals $(\mathbf{y} - \mathbf{X}^T\beta)$.

This formula shows why a SAR model is called "simultaneous". We can't solve for λ (strength of correlation of the residuals) without first computing the coefficients β, but we can't compute the β without knowing λ. However by assuming that the covariance matrix of the spatially-correlated errors is diagonal $\Sigma_\varepsilon = \sigma^2\mathbf{I}$, i.e., no correlation between these errors with equal variance, the variance of the response variable Y can be written:

$$\mathrm{Var}(\mathcal{Y}) = \sigma^2(\mathbf{I} - \lambda\mathbf{W})^{-1})(I - \lambda\mathbf{W}^T)^{-1} \tag{8}$$

and this can be used to find an optimal λ by maximum likelihood, which then allows solution of Equation 7 by generalized least squares (GLS).

> **Note:** See [1, §9.4.1.1] for derivation of the maximum-likelihood estimation of the regression coefficients for these models and how these are solved numerically.

Package `spdep` provides function `spautolm` to compute according to these formulas.

Task 36 : Recompute the linear model of the previous section, taking into account spatial autocorrelation of the residuals in a spatial error SAR model. •

```
m.z.ppp.sar <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
                        data=NY8, listw=NY8listwB)
summary(m.z.ppp.sar)


Call:
spautolm(formula = Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data = NY8,
    listw = NY8listwB)

Residuals:
     Min       1Q   Median       3Q      Max
-1.56754 -0.38239 -0.02643  0.33109  4.01219

Coefficients:
             Estimate Std. Error z value  Pr(>|z|)
(Intercept) -0.618193   0.176784 -3.4969 0.0004707
PEXPOSURE    0.071014   0.042051  1.6888 0.0912635
PCTAGE65P    3.754200   0.624722  6.0094 1.862e-09
PCTOWNHOME  -0.419890   0.191329 -2.1946 0.0281930

Lambda: 0.040487 LR test value: 5.2438 p-value: 0.022026
Numerical Hessian standard error of lambda: 0.017199

Log likelihood: -276.1069
ML residual variance (sigma squared): 0.41388, (sigma: 0.64333)
Number of observations: 281
Number of parameters estimated: 6
AIC: 564.21
```

An important information in this summary is the **likelihood ratio test**, marked `LR test value` in the summary output. This compares the models with and without spatial autocorrelation: the likelihood of the observed values of the response variable, given the values of the predictor, with and without taking into account spatial correlation of the residuals. The p-value is as usual the probability that rejecting the null hypothesis that the two models are equally likely, given the data, would be a Type I error. Here the p-value is low, so we confirm the impression from the map of the residuals that indeed they are spatially autocorrelated.

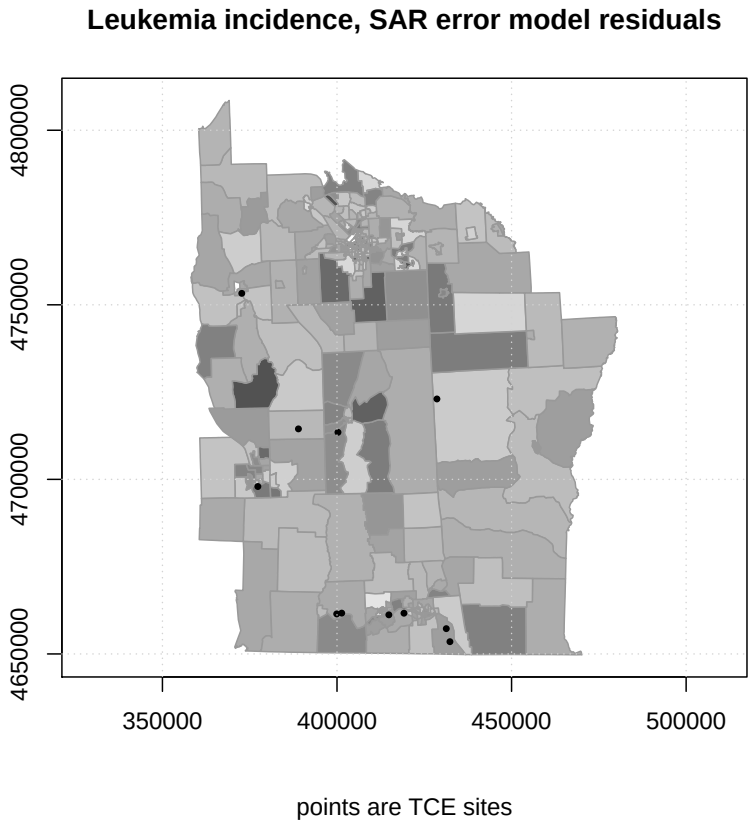
Q26 : *How did the coefficients for the three predictive factors, and their significance, change from the model that did not include simultaneous autoregression?* *Jump to A26* •

We can display these in compact form as fields in the model summaries:

```
round(summary(m.z.ppp)$coefficients[,c(1,4)],4)

            Estimate Pr(>|t|)
(Intercept)  -0.5173   0.0012
PEXPOSURE     0.0488   0.1648
PCTAGE65P     3.9509   0.0000
PCTOWNHOME   -0.5600   0.0011

round(summary(m.z.ppp.sar)$Coef[,c(1,4)],4)

            Estimate Pr(>|z|)
(Intercept)  -0.6182   0.0005
PEXPOSURE     0.0710   0.0913
PCTAGE65P     3.7542   0.0000
PCTOWNHOME   -0.4199   0.0282
```

TASK 37 : Plot these residuals; compute their global Moran's *I*. •

```
NY8$sarresid <- residuals(m.z.ppp.sar)
summary(NY8$sarresid)

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.567536 -0.382389 -0.026430  0.002157  0.331094  4.012191

shade <- round(n*(NY8$sarresid-min(NY8$sarresid))
               /(quantile(NY8$sarresid,.99)-min(NY8$sarresid)))
plot(NY8, border="grey60",
     col=grey.colors(n, 0.9, 0.1, 2.2)[shade],
     axes=TRUE, asp=1)
points(coordinates(TCE), cex=.5, pch=19)
grid()
title("Leukemia incidence, SAR error model residuals",
      "points are TCE sites")
```



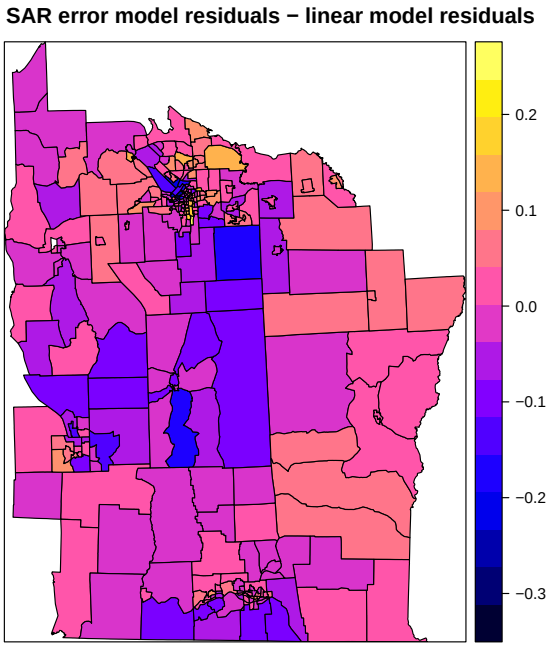
Since `spautolm` does not produce a `lm` object, we can not use `lm.morantest`; instead we use the `moran.test` function directly on the residuals:

```
moran.test(NY8$sarresid, NY8listwB)


Moran I test under randomisation

data:  NY8$sarresid
weights: NY8listwB

Moran I statistic standard deviate = -0.098152, p-value =
0.5391
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
     -0.007052982      -0.003571429       0.001258203
```

Q27 : *Did the simultaneous autoregressive model account for all the spatial autocorrelation in leukemia incidence?* *Jump to A27* •

We can also see where the residuals changed, and by how much:

```
NY8$resid.change <- (NY8$sarresid - NY8$lmresid)
summary(NY8$resid.change)

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.312032 -0.030174  0.004200  0.002157  0.042937  0.237350

spplot(NY8, zcol="resid.change",
       main="SAR error model residuals - linear model residuals")
```



The modelled autocorrelation of the residuals was removed in the SAR model; we see some large decreases in the model residuals in the Onondaga Lake lakefront areas in Syracuse, and large increases in the northeast suburbs of Syracuse. The largest positive residual appears to be in the town of Moravia; this was slightly increased in the SAR model.

We can gain further insight into the model by decomposing the prediction into the **trend** and **stochastic** components according to the model formula of Equation 7, which we repeat here for convenience:

$$\mathbf{y} = \mathbf{X}^T\beta + (\lambda\mathbf{W})(\mathbf{y} - \mathbf{X}\beta) + \varepsilon \tag{9}$$

where the first term is the linear model and the second the correction due to spatial autocorrelation of the linear model residuals. Recall that λ gives the strength of this; $\lambda = 0$ implies no correction.

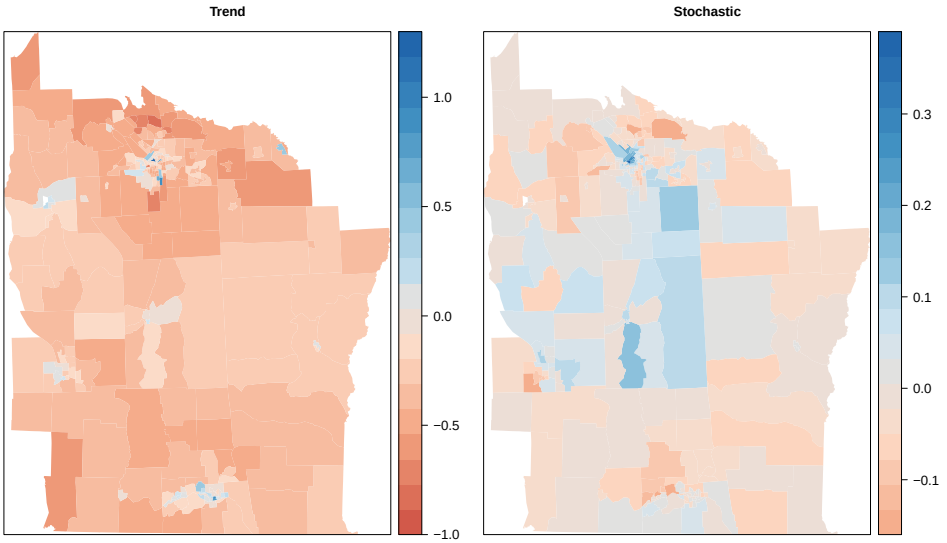
The fitted model is of class `spautolm`. This includes a `fit` field, which itself has two fields, one for each of these components of the fit. So we can just extract these two and plot them.

TASK 38 : Plot the leukemia incidence fit by the model, split into the two components, trend (based on feature space predictors) and spatially-correlated stochastic residuals. •

Following the nice Fig. 9.11 of [1], we use a colour palette provided by the `RColorBrewer` package and built with the `colorRampPalette` function.

```
class(m.z.ppp.sar)

[1] "spautolm"

NY8$sar_trend <- m.z.ppp.sar$fit$signal_trend
```

```
NY8$sar_stochastic <- m.z.ppp.sar$fit$signal_stochastic
library(RColorBrewer)
rds <- colorRampPalette(brewer.pal(8, "RdBu"))
tr_at <- seq(-1, 1.3, length.out=21)
tr_rds <- rds(sum(tr_at >= 0)*2)[-(1:(sum(tr_at >= 0)-sum(tr_at < 0)))]
tr_pl <- spplot(NY8, c("sar_trend"), at=tr_at, col="transparent",
                col.regions=tr_rds, main=list(label="Trend", cex=0.8))
st_at <- seq(-0.16, 0.39, length.out=21)
st_rds <- rds(sum(st_at >= 0)*2)[-(1:(sum(st_at >= 0)-sum(st_at < 0)))]
st_pl <- spplot(NY8, c("sar_stochastic"), at=st_at, col="transparent",
                col.regions=st_rds, main=list(label="Stochastic", cex=0.8))
plot(tr_pl, split=c(1,1,2,1), more=TRUE)
plot(st_pl, split=c(2,1,2,1), more=FALSE)
```



Q28 : *Which of the two components (trend and stochastic residual) gives more information on the predicted leukemia incidence?* *Jump to A28* •

Q29 : *Where is the stochastic residual component most influential in adjusting the trend?* *Jump to A29* •

7.2 * Spatial Lag SAR model

Supplementary reading:

- Bivand et al. [1, §9.4.2]: *Spatial econometrics approaches*

The spatial lag model implies that the response variable is influenced by the same response variable in neighbouring areas. Its formula is:

$$\mathbf{y} = \mathbf{X}^T\beta + \rho\mathbf{W}\mathbf{y} + \varepsilon \tag{10}$$

The parameter ρ controls the degree of autocorrelation of the response variable.

Although it's difficult to imagine how this model could apply to leukemia incidence, to illustrate how this model works we fit and interpret it. The `lagsarlm` function of the `spdep` package fits this model.

TASK 39 : Fit a spatial lag SAR model of leukemia incidence predicted by TCE exposure, proportion of home ownership, and proportion of older people, with the neighbour weights as in the previous models. •

```
m.z.ppp.lagsar <- lagsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
                           data=NY8, listw=NY8listwB)
summary(m.z.ppp.lagsar)


Call:
lagsarlm(formula = Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data = NY8,
    listw = NY8listwB)

Residuals:
      Min        1Q    Median        3Q       Max
-1.586752 -0.391580 -0.022469  0.338017  4.029430

Type: lag
Coefficients: (asymptotic standard errors)
             Estimate Std. Error z value  Pr(>|z|)
(Intercept) -0.514495   0.156154 -3.2948  0.000985
PEXPOSURE    0.047627   0.034509  1.3801  0.167542
PCTAGE65P    3.648198   0.599046  6.0900 1.129e-09
PCTOWNHOME  -0.414601   0.169554 -2.4453  0.014475

Rho: 0.038893, LR test value: 6.9683, p-value: 0.0082967
Asymptotic standard error: 0.015053
    z-value: 2.5837, p-value: 0.0097755
Wald statistic: 6.6754, p-value: 0.0097755

Log likelihood: -275.2447 for lag model
ML residual variance (sigma squared): 0.41166, (sigma: 0.6416)
Number of observations: 281
Number of parameters estimated: 6
AIC: 562.49, (AIC for lm: 567.46)
LM test for residual autocorrelation
test value: 1.4633, p-value: 0.22641
```

Q30 : *What was the strength of the autocorrelation parameter? Is this model better than one without the spatial lag term?* *Jump to A30* •

Q31 : *Which predictors are now significant? Compare to the SAR error model.* *Jump to A31* •

```
round(summary(m.z.ppp.sar)$Coef[,c(1,4)],4)

            Estimate Pr(>|z|)
(Intercept)  -0.6182   0.0005
PEXPOSURE     0.0710   0.0913
PCTAGE65P     3.7542   0.0000
PCTOWNHOME   -0.4199   0.0282

round(summary(m.z.ppp.lagsar)$Coef[,c(1,4)],4)

            Estimate Pr(>|z|)
(Intercept)  -0.5145   0.0010
```

```
PEXPOSURE     0.0476   0.1675
PCTAGE65P     3.6482   0.0000
PCTOWNHOME   -0.4146   0.0145
```

7.3 * Spatial Durbin SAR model

The spatial Durbin model is:

$$\mathbf{y} = \mathbf{X}^T\beta + \rho\mathbf{W}\mathbf{y} + \mathbf{W}\mathbf{X}\gamma + \varepsilon \tag{11}$$

This adds another spatial covariance parameter, γ, which controls the degree of influence of the spatial lag of the covariates (predictors).

TASK 40 : Fit a spatial Durbin SAR model of leukemia incidence predicted by TCE exposure, proportion of home ownership, and proportion of older people, also with the effect of predictor variables in neighbouring areas, with the neighbour weights as in the previous models. •

The `lagsarlm` function of the `spdep` package also fits this model, if the `type` optional argument is specified as `"mixed"`:

```
m.z.ppp.durbin <- lagsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
                           data=NY8, listw=NY8listwB, type="mixed")
summary(m.z.ppp.durbin)


Call:
lagsarlm(formula = Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data = NY8,
    listw = NY8listwB, type = "mixed")

Residuals:
      Min        1Q    Median        3Q       Max
-1.799308 -0.390125 -0.021371  0.346128  3.965251

Type: mixed
Coefficients: (asymptotic standard errors)
                 Estimate Std. Error z value  Pr(>|z|)
(Intercept)     -1.131233   0.249631 -4.5316 5.853e-06
PEXPOSURE        0.218364   0.079301  2.7536  0.005894
PCTAGE65P        3.361158   0.654123  5.1384 2.771e-07
PCTOWNHOME       0.071903   0.253967  0.2831  0.777085
lag.(Intercept)  0.132544   0.056175  2.3595  0.018300
lag.PEXPOSURE   -0.035239   0.015536 -2.2681  0.023322
lag.PCTAGE65P    0.161685   0.223690  0.7228  0.469798
lag.PCTOWNHOME  -0.140681   0.058529 -2.4036  0.016234

Rho: 0.026981, LR test value: 2.558, p-value: 0.10974
Asymptotic standard error: 0.016766
    z-value: 1.6093, p-value: 0.10755
Wald statistic: 2.5899, p-value: 0.10755

Log likelihood: -269.1031 for mixed model
ML residual variance (sigma squared): 0.39587, (sigma: 0.62918)
Number of observations: 281
Number of parameters estimated: 10
AIC: 558.21, (AIC for lm: 558.76)
LM test for residual autocorrelation
test value: 4.908, p-value: 0.026732
```

This model summary shows the coefficients of the three predictors, and

also coefficients for their *lagged* effect, i.e., the effect of the neighbours' values of the predictors.

Q32 : *What was the strength of the autocorrelation parameter? Is this model better than one without the spatial lag term?* *Jump to A32* •

Q33 : *Which predictors are now significant? Compare to the SAR error model. Are any of the the neighbour effects significant? How do these affect the other coefficients?* *Jump to A33* •

```
round(summary(m.z.ppp.sar)$Coef[,c(1,4)],4)

            Estimate Pr(>|z|)
(Intercept)  -0.6182   0.0005
PEXPOSURE     0.0710   0.0913
PCTAGE65P     3.7542   0.0000
PCTOWNHOME   -0.4199   0.0282

round(summary(m.z.ppp.durbin)$Coef[,c(1,4)],4)

                Estimate Pr(>|z|)
(Intercept)      -1.1312   0.0000
PEXPOSURE         0.2184   0.0059
PCTAGE65P         3.3612   0.0000
PCTOWNHOME        0.0719   0.7771
lag.(Intercept)   0.1325   0.0183
lag.PEXPOSURE    -0.0352   0.0233
lag.PCTAGE65P     0.1617   0.4698
lag.PCTOWNHOME   -0.1407   0.0162
```

7.4 * Comparison with point-based modelling

Another way to model polygon data is to consider all attributes to be concentrated at the centroids, and use point-based geostatistical models. Another exercise in this series[11] covers this extensively. Here we compare the coefficients of the OLS linear model based on centroids with that based on polygons – these should be the same. We then examine the spatial dependence of the OLS model residuals; this is the same idea as Moran's I, but based only on distances between centroids. If there is dependence, we then model it and use it to fit coefficients, and re-estimate the spatial correlation structure, using Generalized Least Squares (GLS). We can then compare the GLS coefficients with those from the SAR model from the previous section.

TASK 41 : Make a `SpatialPointsDataFrame` object from the centroids of the polygons and their attributes. •

```
NY8.pts <- SpatialPointsDataFrame(coordinates(NY8), data=NY8@data)
```

TASK 42 : Model the 8-county leukemia incidences by an additive model

[11] `exRKGLS.pdf`

of the same three factors as in the spatial model of polygons, i.e., exposure potential, percent older than 65 years, and percent home ownership. •

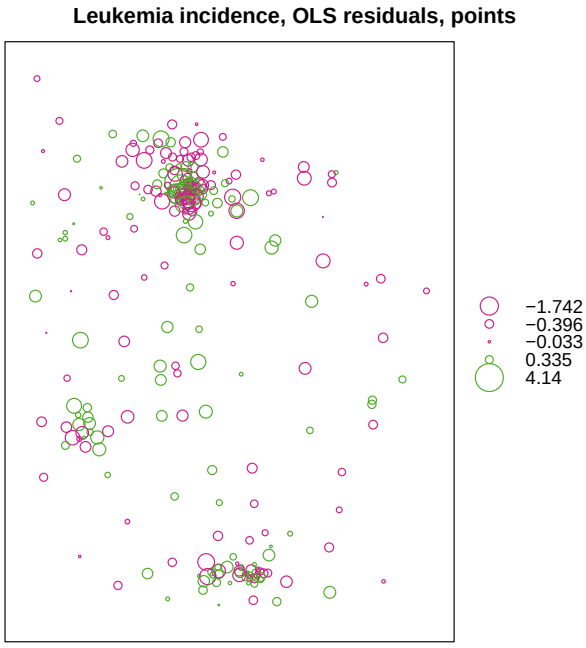
This uses the same non-spatial attributes, and so will have exactly the same coefficients and goodness-of-fit.

```
m.z.ppp.pts <- lm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
       data = NY8.pts@data)
coefficients(m.z.ppp.pts)

(Intercept)   PEXPOSURE   PCTAGE65P  PCTOWNHOME
-0.51727634  0.04883627  3.95088956 -0.56004134

summary(m.z.ppp.pts)$adj.r.squared

[1] 0.1844222
```

This fit assumes independence among model residuals. For a spatial points dataset, we use the variogram of the model residuals to check for this. First, however, we view the a bubble plot to visually assess the spatial correlation.

TASK 43 : Display a bubble plot of the model residuals. •

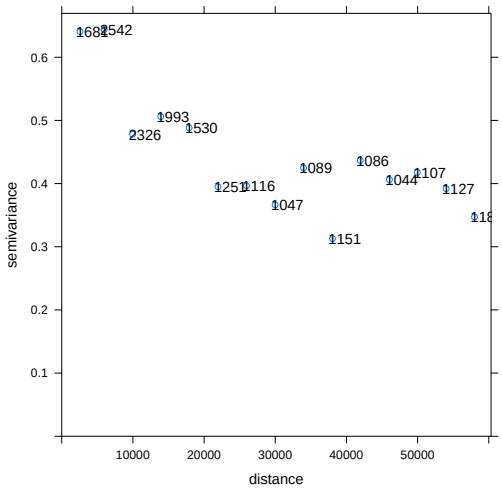
```
NY8.pts$lm.resid <- residuals(m.z.ppp.pts)
bubble(NY8.pts, zcol="lm.resid", pch=1,
       main="Leukemia incidence, OLS residuals, points")
```



Q34 : *Does there appear to be spatial dependence among the residuals?*
Jump to A34 •

TASK 44 : Examine the variogram of the model residuals. •

```
require(gstat)
vr <- variogram(lm.resid ~ 1, loc=NY8.pts)
plot(vr, plot.numbers=T)
```



Q35 : *Does the variogram support the assumption of spatial dependence among the residuals?* *Jump to A35*

•

We conclude that if we consider space as centroids the OLS fit to a linear model is justified.

8 Answers

A1 : *The* `proj4string` *slot of the imported objects is:*

```
+proj=utm +zone=18 +ellps=WGS84 +units=m +no_defs
```

This refers to the PROJ4 format[12]*. This one is easy enough to read: UTM projection from the WGS84 ellipsoid, coordinate system UTM in zone 18N.* *Return to Q1* •

A2 : *There are 1522 total links among the 281 polygons; the average number of links is 5.4; and only six of the polygons have only one link – these must be on the periphery of the area.* *Return to Q2* •

A3 : *(1) The first-order (direct) links are defined by polygon adjacency: if two polygons share any border, or even meet at a single point (see: Newfield and Spencer, lower left corner) they are considered neighbours. The link is drawn between polygon* ***centroids****. (2) The link lengths are quite different: very short*

[12] http://trac.osgeo.org/proj/

to very long. (3) This is mainly because of polygon size: centroids of large polygons are further apart than for small ones – compare inside the cities (e.g., Syracuse) with rural counties (e.g., Chenango, middle-right of figure). (4) Intuitively, it seems that some neighbours should be weighted more than others when considering spatial influence between polygons, because they are closer. Note, we are not yet considering attributes (e.g., rural vs. urban areas, population density) because we don't know what attributes is being analyzed. *Return to Q3* •

A4 : *Field* `AREANAME`*, with 64 names;* `"Syracuse city"` *is the factor name for Syracuse.* *Return to Q4* •

A5 : *The most common number of links is 6; there are 17 census districts with this number of neighbours.* *Return to Q5* •

A6 : *There is one polygon with only one neighbour within the city: 36067002900*[13]*; there is one with nine neighbours: 36067005602*[14]*. These have indices 28 and 56, respectively, and in the 8-county dataset they have row names 136 and 164, respectively.* *Return to Q6* •

A7 : *The number of links is reduced from 346 to 308, i.e., a loss of 38 or 12.34%. There is no "correct" answer to the second part of this question, it depends on the process being modelled.* *Return to Q7* •

A8 : *1, 1/9.* *Return to Q8* •

A9 : *Yes, several high incidences are in the NW (Lakefront neighbourhood and near Westside).* *Return to Q9* •

A10 : *The expectation of Moran's* I *is* $-1/(n-1) = -1/62 = -0.0161290$*; the actual value is of opposite sign and much larger in absolute value; this is quite unlikely to be equal to the expectation of no spatial association. The probability of incorrectly rejecting the null hypothesis of no association (Type I error) is* 8.47×10^{-4}*.* *Return to Q10* •

A11 : *Hypothesized process: there are environmental causes of the leukemia (e.g, industrial pollution) and close-by neighbourhoods have similar distances to these sources. For example, referring to Figure 2, we see the Lakeside neighbourhood (with high incidence) appears to be mostly an industrial area fronting Onondaga Lake.* *Return to Q11* •

A12 : *The probability increased, i.e., the evidence is less strong to reject the null hypothesis. For leukemia incidence, the probability increased from*

[13] Skytop
[14] west part of Near Westside

8.47×10^{-4} *to 0.00178. However, with both weightings the evidence is strong, and we reach the same conclusion.* *Return to Q12* •

A13 : *The highest-leverage area is marked on the graph as original row 109; it has the highest incidence (4.71053) and a moderately-high weighted spatially-lagged proportion; this supports the hypothesis of autocorrelation. The area with row number 109 is adjacent to areas with row numbers 110, 113, 119, 129 and 130 (2, 5, 11, 21 and 22 in the neighbours list); of these 119 also has a high proportion. Areas 113, 129, and 130 have moderately low incidence, but high spatially-lagged proportion; these are the low-incidence neighbourhoods adjacent to high-incidence neighbourhoods. However they have little influence on the slope, because they are almost directly above the centroid.* *Return to Q13* •

A14 : *From this figure it is clear that most of the global Moran's* I *significance comes from the local Moran's* I *from high incidence associated with high incidence, in the Lakefront (NW) area near Onondaga Lake.* *Return to Q14* •

A15 : *Yes, clear evidence; there are 7 areas with local Moran's* I *sufficiently high to reject the null hypothesis with less than a 5% chance of Type I error. Some of these (areas 109, 119, 120, 130) are highlighted in the Moran scatterplot, but others (131, 161) are not – these do not greatly influence the global Moran's* I *but are locally-clustered. An interesting case is area 161 (northern part of Brighton), with a very low incidence (*Z *= -1.354) and low-incidence neighbours.* *Return to Q15* •

A16 : *The global Moran's I are all close to* 0.2 ± 0.005 *and are highly significant; there is not much variability. Note that the expected value is the same because this just depends on the number of observations, not on the weighting. The influential observations are the same five, except for inverse-distance which adds district 162. Looking at the local Moran's I plots, there is some difference in the positions of the influential observations on the y-axis (weighted average neighbour Z); note that the x-axis is the same because this is just the observed Z value in each district. Also, note the different scales of the y-axis because of the different weights; however the position of the horizontal line showing the expected value is the same.* *Return to Q16* •

A17 : *Clusters of high leukemia incidence are in the NW, between Onondaga Lake and downtown. There are a few very high Z-scores, indicating a high probability of clustering. Clusters of low incidence are in the centre of the map, but these Z-scores are much lower, so the apparent clustering is likely not statistically significant. The local Moran's* I *plot identifies the high incidence clusters in the same area.* *Return to Q17* •

A18 : G_i^* *has a wider range than* G_i*, because the value in the target area is included in the weighted sum for each area. However, on average in this case*

$G_i^* < G_i$, because of the negative values of the index. *Return to Q18* •

A19 : *The weights are now distributed across one more polygon for each polygon's weights; this is the target polygon. So, there are more weights and a larger sum of weights.* *Return to Q19* •

A20 : *Including the target area's value emphasizes the hotspots with extreme values of the target variable. In particular the lakeside area now has a much higher Z-score than in the G_i plot.* *Return to Q20* •

A21 : *Leukemia is a form of cancer; its causes are obscure, but seem to include genetic, demographic and environmental factors.*

For the genetic factors, one could think of the ethnic composition of census tracts; this would certainly be true for sickle-cell anaemia, which occurs largely in people with recent west African ancestors.

For the demographic factors, as with almost all cancers leukemia is more prevalent with increasing age, explained by more time to allow something to wrong with cell renewal.

For the environmental factors, industrial chemicals, especially petrochemicals, may increase the risk of leukemia. Smoking may also increase the risk. In both cases, the older a person, the more years they have been exposed to the environmental factor, increasing the natural effect of age just mentioned..
Return to Q21 •

A22 : *18.4%, i.e., about one-fifth. This is significantly different from zero, so the zero-mean model was not justified. The most significant factor is a positive relation with the proportion of older people (suggesting perhaps a link to smoking? or general increased cancer risk with age?) and negative relation with home ownership (suggesting perhaps a higher living standard and healthier lifestyle??); surprisingly, the positive relation with log-distance to TCE source is not significant.* *Return to Q22* •

A23 : *Tract 109 has a very high log-incidence (field Z), because although it has few cases, the population is only 9 people – look at Figure 2: this area is mostly industrial, with almost no homes. This nicely illustrates the modifiable area problem: if this tract were included in a neighbouring tract with a more typical population (several hundred to several thousands), the extreme incidence would disappear or at least be diluted.* *Return to Q23* •

A24 : *Yes, the probability that we would be wrong to reject the null hypothesis of no spatial correlation is only 0.0042.* *Return to Q24* •

A25 : *Yes. Although there are scattered highs and lows, there seems to be a cluster of high residuals (under-predictions) near Ithaca (Morse Chain TCE*

site), another in and around Binghamton and Johnson City (many TCE sites), and another near the Smith-Corona factory in Cortland.

The most under-predicted area is one we've seen before, the Onondaga lake-front industrial area in Syracuse. Perhaps this is a TCE source that was not mapped? Or it produces a different petrochemical linked to leukemia? Hint: what is its population?

There are several areas of near-zero clusters, e.g., in southern Chenango and northern Broome counties (SE corner of the map). The north of Cayuga county (far N) has quite similar moderately positive residuals. All these similar values (whether high, zero or low) contribute to the observed Moran's test value. The inference is that the model is not complete: either there are other spatially-distributed predictive factors (i.e., more information about the census tracts) or that there is really a spatial process independent of predictive factors.

For leukemia, this latter is hard to imagine. But for an insect-borne disease (e.g., a plant virus) spreading through an area by diffusion, this could well be the principal process.

Finally, since the high residuals seem to be linked with TCE sites, perhaps the log-inverse distance weighting was not the most appropriate to represent this process. *Return to Q25* •

A26 : *The positive coefficient for "exposure potential" increased at the expense of the other two factors. It is now significant at the $p < 0.1$ level. The other two factors remain dominant, especially age. Thus by building the SAR model we have more evidence that TCE exposure may be an important factor related to leukemia incidence.* *Return to Q26* •

A27 : *Yes, the p-value of the global Moran's test is quite high; we have no evidence to reject the null hypothesis of no residual autocorrelation; thus the autocorrelation in the linear model residuals has been accounted for.* *Return to Q27* •

A28 : *The trend is much more influential, ranging from about -1 to +1 in normalized incidence (variable Z).* *Return to Q28* •

A29 : *In Syracuse city and in a band from SE Onondaga county through most of Cortland county the effect of accounting for spatial correlation of the residuals increases the predicted incidence. There is not much negative influence, only in the SW Town of Ithaca, some tracts in Binghamton city, and in the Cicero game management area in the centre N; this area has a very small population.* *Return to Q29*
•

A30 : *The strength of spatial association among predictors is $\rho = 0.039$. The LR test shows that there is less than a 1% chance that rejecting the null hypothesis of no improvement from the OLS model due to including autocorrelation in the model would be wrong. This is strong evidence that there is autocorrelation in the response variable, which is accounted for by the SAR lag model.*

Return to Q30 •

A31 : *As in the SAR error model, the proportion of older people is dominant (positive association), and the proportion of homeowners somewhat less so (negative). The coefficients are quite close to those for the SAR error mode but all somewhat closer to zero; thus the predictive factors are somewhat less predictive once the residual autocorrelation of the response variable is accounted for.* Return to Q31 •

A32 : *The strength of spatial association among predictors is $\rho = 0.027$. According to the LR-test we can not reject the null hypothesis that the spatial Durbin SAR model is not superior to the OLS model.* Return to Q32 •

A33 : *The non-lagged coefficients are noticeably different from those in the SAR error model; the sign for* `PCTOWNHOME` *is flipped from negative (as expected) to slightly positive. These changes can be explained by the lagged coefficients. The model proposes that home ownership in neighbouring areas is predictive of leukemia in a target area! TCE exposure becomes significant at the 1–2% level for both the target and neighbouring areas.*

This model is difficult to justify; in fact the LR-test suggests that this model should not be used in preference to the SAR error model. Return to Q33 •

A34 : *No, the red and green circles seem to be intermixed.* Return to Q34 •

A35 : *There is less spatial correlation at close range (e.g., within Syracuse and the smaller cities) than at longer range. So there is no spatial dependence to be removed from the model.* Return to Q35 •

9 Assignment

This is a small self-test of how well you mastered this exercise. You should be able to complete the tasks and answer the questions with the knowledge you have gained from the exercise.

We turn to another USA city: Columbus, Ohio; the data is from Anselin[15], formatted by him and provided as an example in the `spData` package, as a shapefile[16]

TASK 1 : Read the example dataset from a shapefile into a `sp` object, with the `readOGR` function of the `rgdal` package.

```
require(rgdal); require(spData)
columbus <- readOGR(system.file("shapes/columbus.shp",
                                package="spData")[1])
```

•

TASK 2 : View its description (via the help system) and field definitions.

•

Q1 : *Which field in the attribute table represents residential burglaries and vehicle thefts per thousand households in the neighborhood?* •

TASK 3 : Read the prepared neighbour list `columbus.gal` into an `nb` "neighbours" object and summarize it. This list is also stored in the `spData` package. •

The full file name, with path to the place in the R installation where it is stored, can be retrieved with the `system.file` function:

```
system.file("weights/columbus.gal", package="spData")
```

Q2 : *How many polygons? How many total links?* •

TASK 4 : Plot the polygons with the links superimposed, coordinate axes, map aspect, and the neighbourhood ID as a label. •

Q3 : *Can these polygons be easily converted to a KML overlay? Why or why not?* •

[15] Anselin, L. (1988). *Spatial econometrics: methods and models*. Boston: Kluwer Academic

[16] also available from the GeoDa project, at https://geodacenter.github.io/data-and-lab//columbus/

We will analyze the spatial structure of residential burglaries and vehicle thefts per thousand households in the neighborhood ("crime" for short).

Q4 : *Which neighbourhoods have the highest and lowest incidences of residential crime? What are these rates?* •

TASK 5 : Plot the neighbourhoods with the crime frequency represented by a grey scale. •

> **Note:** Use the `ceiling` function on the crime frequency, normalized by the maximum frequency, to ensure that the lowest-crime neighbourhood plots with grey scale 1 and the highest-crime with the maximum.

Q5 : *Is there visual evidence for spatial autocorrelation of the crime rate? Cite some specific evidence.* •

Q6 : *Is the default weighting (inverse of the number of neighbours from a source, style "W") appropriate for this attribute? Why or why not?* •

TASK 6 : Create a binary weighting for this attribute. •

Q7 : *What hypothesis about spatial influence does a binary weighting imply?* •

TASK 7 : Compute the global Moran's I statistic for this attribute. •

Q8 : *Is there evidence for spatial dependence of crime rate?* •

TASK 8 : Plot the Moran's scatterplot for the crime attribute. •

Q9 : *What is the overall pattern of spatial association? Which districts most influence the significant test result?* •

Q10 : *Which of the other attributes might be correlated with ("explain") the crime rate?* •

TASK 9 : Build a simultaneous autoregressive model to explain crime

from one or more of these factors. •

Q11 :

(1) Which factors in your model were most important?

(2) Were the residuals of the ordinary linear model (without accounting for spatial correlation) spatially-autocorrelated? Was it necessary to include the autocorrelation in the model? •

References

[1] Roger S. Bivand, Edzer J. Pebesma, and V. Gómez-Rubio. *Applied Spatial Data Analysis with R*. Springer, 2nd edition, 2013. ISBN 978-1-4614-7617-7; 978-1-4614-7618-4 (e-book). URL http://www.asdar-book.org/. 1, 2, 6, 18, 21, 22, 25, 38, 43, 45, 48, 49

[2] James Durbin. Estimation of parameters in time-series regression-models. *Journal of the Royal Statistical Society Series B*, 22(1):139–153, 1960. 44

[3] P. A. P. Moran. Notes on continuous stochastic phenomena. *Biometrika*, 37(1/2):17–23, 1950. doi: 10.2307/2332142. 21

[4] J. K. Ord and Arthur Getis. Local spatial autocorrelation statistics: distributional issues and an application. *Geographical Analysis*, 27 (4):286–306, 1995. doi: 10.1111/j.1538-4632.1995.tb00912.x. 34, 35

[5] L. A. Waller and C. A. Gotway. *Applied spatial statistics for public health data*. Wiley-Interscience, Hoboken, N.J., 2004. 1, 2

[6] Yihui Xie. `knitr`: Elegant, flexible and fast dynamic report generation with R, 2011. URL http://yihui.name/knitr/. Accessed 04-Mar-2016. 2

Index of R Concepts