Tutorial: Areal Data and Spatial Autocorrelation

D G Rossiter
Cornell University, Section of Soil & Crop Sciences

March 7, 2024

Contents

世上无难事，只怕有心人
"There are no difficult tasks, only fearful people" – Chinese 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 (§4);
2. Compute spatial weights reflecting the strength of association according to various criteria (§6);
3. Compute global and local Moran's I as measures of spatial autocorrelation (§7);
4. Identify "hot" and "cold" spots (§7.2.3);
5. Build spatial autotregressive models that combines feature-space modelling ("regression") with spatial autocorrelation (§8);
6. 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 [8] `sp` package , `spdep` package , `spatialreg` package , `gstat` package on R version 4.2.3 (2023-03-15), running on Mac OS X 14.3.1. 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 Setup

Most of the analysis in this tutorial is carried out by functions in the `spdep` "Spatial dependence" and `spatialreg` "Spatial regression" packages from Roger Bivand. These depend on the `sp` "spatial classes" package, but also work with the newer `sf` package spatial representation.

Most of the graphics use the `ggplot2` package for "Grammar of Graphics" [7] figures, as implemented in the R package `ggplot2` [6].

Task 1 : Load the required packages. •

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

```
library(sf)             # the "Simple Features' representation of spatial objects
library(sp)             # an older representation of spatial objects
library(spdep)          # spatial dependency of areal data
library(spatialreg)     # spatial regression on areal data
library(RColorBrewer)   # colour palettes
library(gstat)          # classical geostatistics
library(ggplot2)        # Grammar of Graphics
library(gridExtra)      # arrange plots on a page
```

3 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 in the compressed file `lat_bundle.zip` at the ASDAR book website[2] under the "Data sets and scripts" heading, as the data set bundle for Chapter 9. It may be provided with this Tutorial as compressed file `NY_data.zip`.

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.

Task 2 : Locate this file, unpack it in a working directory, start R and connect to that directory. List the shapefiles in the directory. •

[1] Broome, Cayuga, Chenango, Cortland, Madison, Onondaga, Tioga, Tompkins
[2] http://www.asdar-book.org/

For this, use the `list.files` function:

```
list.files(path = ".", pattern="*.shp")
```

```
## [1] "NY8_utm18_centroids.shp" "NY8_utm18.shp"
## [3] "NY8cities.shp"           "TCE.shp"
```

Task 3 : Import the polygon data from the shapefile `NY8_utm18.shp` into R, along with point files `NY8cities.shp` showing cities, and examine their structures and data summaries. •

Polygon shapefiles can be imported to R with the `st_read` function of the `sf` package. It recognizes many file formats, including ESRI Shapefiles, as in this sample data. We then examine the class of the imported object with the `class` function, and see its slots with the `str` "structure" function.

```
NY8 <- st_read(dsn = ".", layer = "NY8_utm18")
class(NY8)
str(NY8, max.level=2)
summary(NY8)
NY8$AREANAME <- as.factor(NY8$AREANAME)
#
cities <- st_read(dsn = ".", layer = "NY8cities")
class(cities)
print(cities)
```

```
## Reading layer `NY8_utm18' from data source
##   `/Users/rossiter/Library/Mobile Documents/com~apple~CloudDocs/Documents/data/edu/AreaData/ds/ASDA
##   using driver `ESRI Shapefile'
## Simple feature collection with 281 features and 17 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 358241.9 ymin: 4649755 xmax: 480393.1 ymax: 4808545
## Projected CRS: WGS 84 / UTM zone 18N
## Classes 'sf' and 'data.frame': 281 obs. of  18 variables:
##  $ AREANAME  : chr  "Binghamton city" "Binghamton city" "Binghamton city" "Binghamton city" ...
##  $ AREAKEY   : chr  "36007000100" "36007000200" "36007000300" "36007000400" ...
##  $ X         : num  4.07 4.64 5.71 7.61 7.32 ...
##  $ Y         : num  -67.4 -66.9 -67 -66 -67.3 ...
##  $ POP8      : num  3540 3560 3739 2784 2571 ...
##  $ TRACTCAS  : num  3.08 4.08 1.09 1.07 3.06 1.06 2.09 0.02 2.04 0.02 ...
##  $ PROPCAS   : num  0.00087 0.001146 0.000292 0.000384 0.00119 ...
##  $ PCTOWNHOME: num  0.328 0.427 0.338 0.462 0.192 ...
##  $ PCTAGE65P : num  0.147 0.235 0.138 0.119 0.142 ...
##  $ Z         : num  0.142 0.356 -0.582 -0.296 0.457 ...
##  $ AVGIDIST  : num  0.237 0.209 0.171 0.141 0.158 ...
##  $ PEXPOSURE : num  3.17 3.04 2.84 2.64 2.76 ...
##  $ Cases     : num  3.08 4.08 1.09 1.07 3.06 ...
##  $ Xm        : num  4069 4639 5709 7614 7316 ...
##  $ Ym        : num  -67353 -66862 -66978 -65996 -67318 ...
##  $ Xshift    : num  423391 423961 425031 426935 426638 ...
##  $ Yshift    : num  4661502 4661993 4661878 4662859 4661537 ...
##  $ geometry  :sfc_POLYGON of length 281; first list element: List of 1
##   ..$ : num [1:48, 1:2] 421840 422095 422308 422456 422464 ...
##   ..- attr(*, "class")= chr [1:3] "XY" "POLYGON" "sfg"
##  - attr(*, "sf_column")= chr "geometry"
##  - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...
##   ..- attr(*, "names")= chr [1:17] "AREANAME" "AREAKEY" "X" "Y" ...
##    AREANAME           AREAKEY                X
##  Length:281         Length:281         Min.   :-55.482
##  Class :character   Class :character   1st Qu.:-19.460
##  Mode  :character   Mode  :character   Median :-12.469
##                                        Mean   :-11.309
##                                        3rd Qu.: -1.213
##                                        Max.   : 53.509
##        Y               POP8          TRACTCAS        PROPCAS
```

```
##  Min.   :-75.29   Min.   :    9   Min.   :0.000   Min.   :0.0000000
##  1st Qu.:-30.60   1st Qu.: 2510   1st Qu.:0.310   1st Qu.:0.0000930
##  Median : 31.97   Median : 3433   Median :1.890   Median :0.0004130
##  Mean   :  4.98   Mean   : 3764   Mean   :2.107   Mean   :0.0005947
##  3rd Qu.: 39.12   3rd Qu.: 4889   3rd Qu.:3.080   3rd Qu.:0.0009170
##  Max.   : 56.41   Max.   :13015   Max.   :9.290   Max.   :0.0069930
##    PCTOWNHOME          PCTAGE65P              Z
##  Min.   :0.0008224   Min.   :0.004044   Min.   :-1.9206
##  1st Qu.:0.4588745   1st Qu.:0.099926   1st Qu.:-0.7168
##  Median :0.6508585   Median :0.126415   Median :-0.2876
##  Mean   :0.5872621   Mean   :0.137262   Mean   :-0.2157
##  3rd Qu.:0.7560976   3rd Qu.:0.160963   3rd Qu.: 0.2498
##  Max.   :1.0000000   Max.   :0.505050   Max.   : 4.7105
##     AVGIDIST         PEXPOSURE          Cases
##  Min.   :0.01847   Min.   :0.6134   Min.   :0.00014
##  1st Qu.:0.02703   1st Qu.:0.9942   1st Qu.:0.30928
##  Median :0.03238   Median :1.1749   Median :1.88876
##  Mean   :0.14919   Mean   :1.8042   Mean   :2.10676
##  3rd Qu.:0.13008   3rd Qu.:2.5656   3rd Qu.:3.08284
##  Max.   :3.52637   Max.   :5.8654   Max.   :9.28601
##        Xm               Ym             Xshift           Yshift
##  Min.   :-55482   Min.   :-75291   Min.   :363839   Min.   :4653564
##  1st Qu.:-19460   1st Qu.:-30601   1st Qu.:399862   1st Qu.:4698254
##  Median :-12469   Median : 31970   Median :406852   Median :4760825
##  Mean   :-11309   Mean   :  4980   Mean   :408013   Mean   :4733835
##  3rd Qu.: -1213   3rd Qu.: 39123   3rd Qu.:418108   3rd Qu.:4767978
##  Max.   : 53509   Max.   : 56410   Max.   :472830   Max.   :4785265
##           geometry
##  POLYGON      :281
##  epsg:NA      :  0
##  +proj=utm ...:  0
##
##
##
## Reading layer `NY8cities' from data source
##   `/Users/rossiter/Library/Mobile Documents/com~apple~CloudDocs/Documents/data/edu/AreaData/ds/ASDA
##   using driver `ESRI Shapefile'
## Simple feature collection with 6 features and 1 field
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 372236.8 ymin: 4662141 xmax: 445727.9 ymax: 4771698
## Projected CRS: WGS 84 / UTM zone 18N
##     names                   geometry
##  Length:6           POINT        :6
##  Class :character   epsg:NA      :0
##  Mode  :character   +proj=utm ...:0
## Simple feature collection with 6 features and 1 field
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 372236.8 ymin: 4662141 xmax: 445727.9 ymax: 4771698
## Projected CRS: WGS 84 / UTM zone 18N
##         names                geometry
## 1 Binghampton POINT (424373.9 4662141)
## 2      Ithaca   POINT (377506 4702477)
## 3    Cortland POINT (403019.9 4717570)
## 4      Auburn POINT (372236.8 4754523)
## 5    Syracuse POINT (406070.4 4767274)
## 6      Oneida POINT (445727.9 4771698)
```

As you can see, `st_read` reports the feature type of the source file, and converts these to `sf` objects. There are 281 census blocks in this dataset. @

> **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[3]. Although, the

[3] named for William Bingham, a Philadelphia politician and land speculator who bought

"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'.

Correct it:

```
ix <- which(cities$names == "Binghampton")
cities$names[ix] <- "Binghamton"
```

3.1 Check for validity

Check that the geometry of the polygon shapefile is valid. The shapefile format is not strict, we need a valid geometry for some geometric operations on the polygons. The `st_is_valid` function of the `sf` package performs this check, and the `st_make_valid` function is able to correct some problems.

```
# find errors
tmp <- st_is_valid(NY8, reason = TRUE)
ix <- which(tmp != "Valid Geometry")
print(data.frame(polygonID = ix, reason = tmp[ix]))

##   polygonID                                                reason
## 1        24      Self-intersection[430016.188394857 4675077.19105705]
## 2        28      Self-intersection[430016.188394857 4675077.19105705]
## 3       173 Ring Self-intersection[418270.385382628 4781224.46209681]
## 4       210 Ring Self-intersection[402604.228645007 4766868.58988568]
## 5       224 Ring Self-intersection[411825.337970521 4767465.38843592]
```

Yes, there are some errors, probably due to sloppy digitalization of source maps. Fix these:

```
# fix it
NY8 <- st_make_valid(NY8)
all(st_is_valid(NY8))

## [1] TRUE
```

3.2 Plot

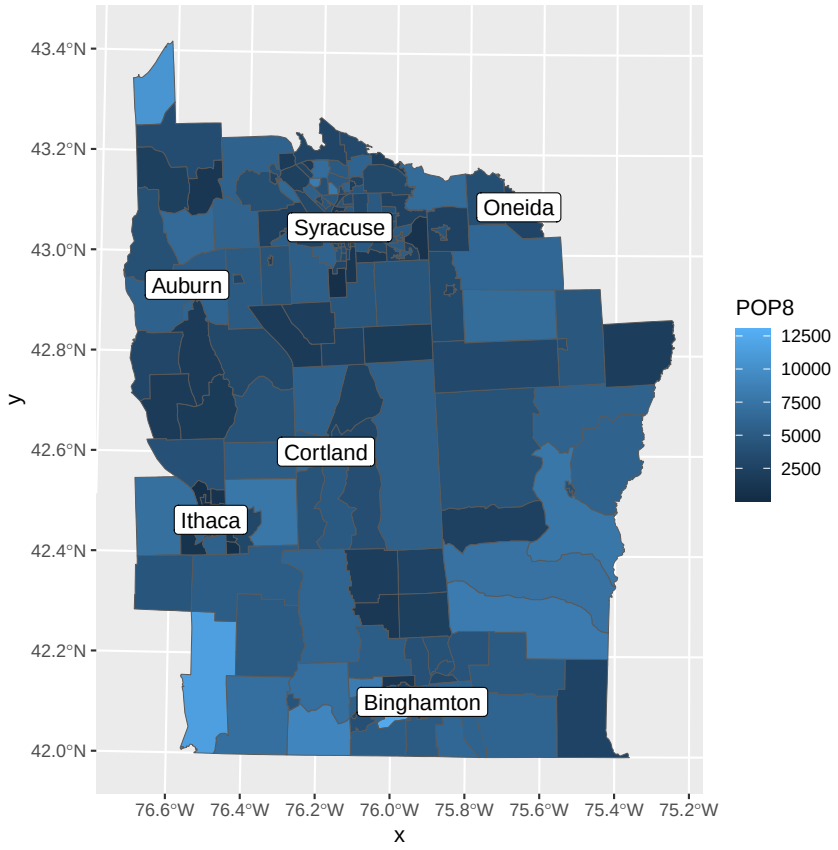
Task 4 : Plot the polygons, coloured by population, with cities overlaid and labelled. •

We use the "Grammar of graphics" R package `ggplot2`[4] functions for this. Function `ggplot` sets up the graphics system, and `geom_sf` adds "Simple Features" geometry objects to the plot.

```
ggplot(data = NY8) +
  geom_sf(aes(fill = POP8)) +
  geom_sf(data = cities) +
  geom_sf_label(data = cities, aes(label = names))
```

the land, then part of Tioga County, in 1792.

[4] https://ggplot2-book.org/



Q1 : *What is the georeference of this dataset? This is referred to as the Coördinate Reference System (CRS). Note that can't be determined from the previous plot, because the* `geom_sf` *function uses geographic coordinates by default.* *Jump to A1* •

The `st_crs` function extracts this information from objects of the `sf` classes.

```
st_crs(NY8)$proj4string

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

st_crs(cities)$proj4string

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

We first examine the spatial organization of the area data ("Spatial neighbours", §4) and how to weight the neighbourhood relations ("Spatial weights", §6). We then describe the attributes of each area and how to discover the autocorrelation structure ("Spatial autocorrelation", §7).

4 Spatial neighbours

Supplementary reading:

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

The concept of "neighbour" is crucial in area data analysis, especially in

the spatial autoregressive models, which include neighbour effects in linear models (§9). But what is a "neighbour"? Depending on the process, i.e., how neighbours influence each other, there are several reasonable definitions, which we now will review.

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 are created from polygons as explained in the following subsections.

Note: They can also be provided in the "GAL lattice" format[5].

4.1 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 5 : Compute a neighbours list for the 8-counties. Display its summary, and for the first polygon, the number of (using the `card` function) and identity of its neighbours. •

The default is "queen" connectivity, based on adjacency, not distance.

```
NY8_nb <- poly2nb(NY8)
class(NY8_nb)

## [1] "nb"

summary(NY8_nb)

## Neighbour list object:
## Number of regions: 281
## Number of nonzero links: 1624
## Percentage nonzero weights: 2.056712
## Average number of links: 5.779359
## Link number distribution:
##
##  1  2  3  4  5  6  7  8  9 10 11
##  6  6 16 37 58 61 48 30 14  3  2
## 6 least connected regions:
## 56 98 101 102 245 246 with 1 link
## 2 most connected regions:
## 35 83 with 11 links

card(NY8_nb)[1]

## [1] 8

NY8_nb[[1]]

## [1]  2 13 14 15 47 48 49 50
```

Notice the summary informarion on the neighbourhood structure.

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

The `poly2nb` function has two optional arguments that can greatly affect the neighbours list based on connectivity:

- `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 6 : 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. •

Again this is based on adjacency, not distance.

```
NY8_nb2 <- poly2nb(NY8, queen=FALSE)
summary(NY8_nb2)

## Neighbour list object:
## Number of regions: 281
## Number of nonzero links: 1528
## Percentage nonzero weights: 1.935133
## Average number of links: 5.437722
## Link number distribution:
##
##  1  2  3  4  5  6  7  8  9 10 11
##  6  8 18 53 67 48 49 20  8  2  2
## 6 least connected regions:
## 56 98 101 102 245 246 with 1 link
## 2 most connected regions:
## 35 83 with 11 links

NY8_nb[[1]]

## [1]  2 13 14 15 47 48 49 50

NY8_nb2[[1]]

## [1]  2 13 14 15 48 49 50
```

Note:

It is also possible to use the `st_touches` function to obtain neighbour lists for "Queen" connectivity.

```
# st_touches includes single points of contact
NY8_st <- st_touches(NY8)
NY8_st[[1]]

## [1]  2 13 14 15 47 48 49 50

class(NY8_st)

## [1] "sgbp" "list"
```

This can be converted to `nb` objects simply by assigning that class, since the structures are identical.

```
NY8_st_nb <- NY8_st
class(NY8_st_nb) <- "nb"
```

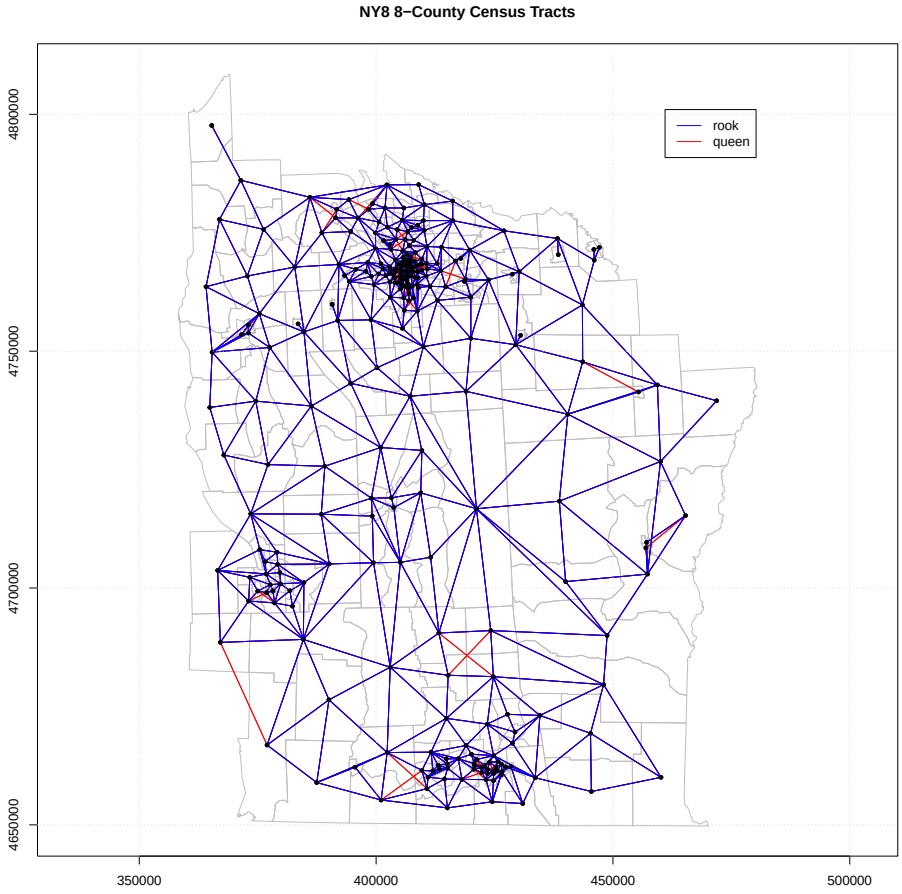
Q2 : *How many links were deleted? Which set of links more realistically represents the concept of "neighbour" between census tracts?* *Jump to A2*
•

To answer this, we sum the length of the link list, within the list of census blocks, using the `lapply` 'apply a function to a list' function:

```
(n.links.queen <- sum(unlist(lapply(NY8_nb, length))))

## [1] 1624

(n.links.rook <- sum(unlist(lapply(NY8_nb2, length))))

## [1] 1528

print(n.links.queen - n.links.rook)

## [1] 96
```

Task 7 : Plot the polygons with the links superimposed. •

```
plot(st_geometry(NY8), border="grey",
     main = "NY8 8-County Census Tracts",
     axes = TRUE)
coords_centers <- st_coordinates(st_centroid(st_geometry(NY8)))
plot(NY8_nb, coords_centers, add=TRUE,
     pch=19, cex = 0.6, col="red")
plot(NY8_nb2, coords_centers, add=TRUE,
     pch=19, cex = 0.6,col="blue")
legend(461000, 4801000, c("rook","queen"),
       lty=1, col=c("blue","red"))
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* •

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(NY8_nb, length)))
```

```
##   [1]  8  6  6  4  6  6  5  4  9  5  6  6  6  6  5  7  5  3  6  8  7
##  [22]  5  7  4  4  7  5  7  5  3  3  8  4  7 11  6  9  7  5  3  6  9
##  [43]  5  4  5  6  5  7  3  7  5  8  5  6  5  1  4  5  4  4  5  9  6
##  [64]  3  4  7  5  3  6  7  6  5  4  6  5  3  3  4  7  4  5  6 11  5
##  [85]  7  7  5  4  6  4  8  4  6  2  2  4  5  1  5  7  1  1  8  6  5
## [106]  8  3  6  2  8  6  4  8  7  5  4  5  8  7  5  7  5  7  6  6  6
## [127]  6  6  7  7  7  6  8  4  5  5  9  6  8  6  6  6  8  6  6  5  8
## [148]  6  6  7  5  7  6  4  7  6  6  6  4  6  7  4  9  7  3  6  8  5
## [169]  7  6  5  5  4  3  7  8  5  7  4  4  7  5  9  5  6  6  3  7  4
## [190]  9  5  8  9  5  5  5  6  7 10  3  8  5  4  5  8  9  4  7  5  7
```

```
## [211]  4  7  9  3  8  7  5  5  8  7  5  9  6  9  7  6  6  2  8  6  2
## [232]  7  2  6  8  6  8  5  7 10  6  8  7  7  1  1  8  6  5  7  6  8
## [253]  5  9  6  3  5  4  6  8  5  5  6  5  8  4  7  6  4  5  5  7  8
## [274]  4  4  7  7 10  4  4  6
```

```
table(n.nb)
```

```
## n.nb
##  1  2  3  4  5  6  7  8  9 10 11
##  6  6 16 37 58 61 48 30 14  3  2
```

Q4 : *What is the most common number of neighbours?* *Jump to A4* •

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] 11
```

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

```
## [1]  56  98 101 102 245 246
```

```
NY8[ix.min.nb,]
```

```
## Simple feature collection with 6 features and 17 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 360387.1 ymin: 4752012 xmax: 440217.7 ymax: 4808545
## Projected CRS: WGS 84 / UTM zone 18N
##                AREANAME     AREAKEY         X        Y  POP8 TRACTCAS
## 56                   NA 36011990100 -50.24266 36.81422 10494     3.85
## 98    Canastota village 36053030300  18.89833 41.54373  4773     0.44
## 101                  NA 36053030403   8.49700 38.27917  4295     1.39
## 102   Cazenovia village 36053030501  10.82350 24.17720  2599     3.24
## 245   Marcellus village 36067016502 -28.68800 30.85230  1870     1.03
## 246 Skaneateles village 36067016600 -35.98330 26.86590  2786     1.04
##      PROPCAS PCTOWNHOME  PCTAGE65P        Z  AVGIDIST PEXPOSURE
## 56  0.000367  0.5365118 0.03220888 -0.77182 0.0774664  2.047259
## 98  0.000092  0.5683716 0.15482925 -1.19833 0.0206926  0.727191
## 101 0.000324  0.7482353 0.11105937 -0.58616 0.0226724  0.818563
## 102 0.001247  0.5618091 0.14736437  0.48944 0.0332935  1.202777
## 245 0.000551  0.4950860 0.15187166  0.08210 0.0525526  1.659229
## 246 0.000373  0.6508585 0.19059584 -0.31166 0.0918058  2.217090
##       Cases        Xm       Ym   Xshift  Yshift
## 56  3.83889 -50242.66 36814.22 369078.9 4765669
## 98  0.43958  18898.33 41543.73 438219.9 4770399
## 101 1.39555   8497.00 38279.17 427818.6 4767134
## 102 3.23935  10823.50 24177.20 430145.1 4753032
## 245 1.02822 -28688.00 30852.30 390633.6 4759707
## 246 1.04204 -35983.30 26865.90 383338.3 4755721
##                            geometry
## 56  POLYGON ((369143 4807833, 3...
## 98  POLYGON ((438903.1 4768869,...
```

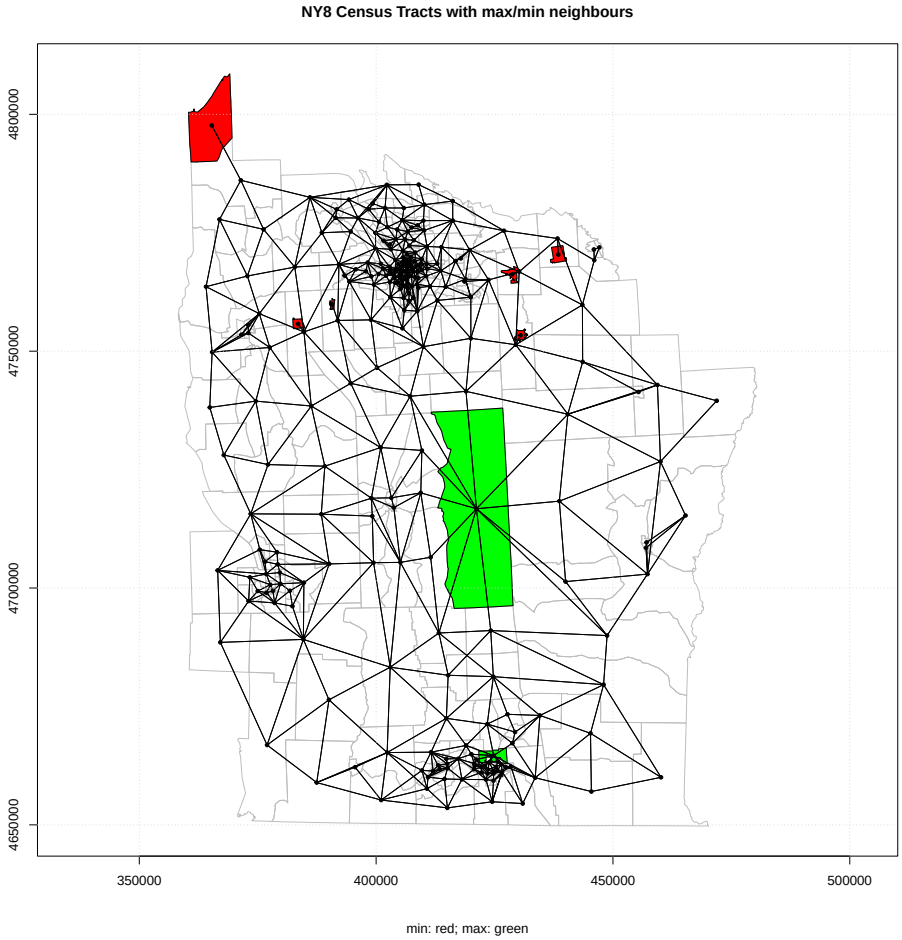
```
## 101 POLYGON ((430465.2 4767073,...
## 102 POLYGON ((429880.3 4754495,...
## 245 POLYGON ((391274.5 4760874,...
## 246 POLYGON ((382461.1 4755744,...

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

## [1] 35 83

NY8[ix.max.nb,]

## Simple feature collection with 2 features and 17 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 411636.9 ymin: 4662952 xmax: 428909.5 ymax: 4737992
## Projected CRS: WGS 84 / UTM zone 18N
##    AREANAME     AREAKEY        X        Y POP8 TRACTCAS  PROPCAS
## 35       NA 36007012800 6.219888 -64.6361 5594     5.13 0.000917
## 83       NA 36023990100 1.861683 -16.6013 5532     3.35 0.000606
##    PCTOWNHOME PCTAGE65P        Z  AVGIDIST PEXPOSURE   Cases       Xm
## 35  0.6905334 0.1973543  0.09150 0.1459107  2.680410 5.13091 6219.888
## 83  0.6787440 0.1231020 -0.24037 0.0763459  2.032689 3.33995 1861.683
##          Ym   Xshift  Yshift                       geometry
## 35 -64636.1 425541.5 4664219 POLYGON ((421936.5 4665583,...
## 83 -16601.3 421183.3 4712254 POLYGON ((411675.8 4736791,...
```

There are several polygons with minimum and maximum neighbours.

Q5 : *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? What are their census codes (see field* `AREAKEY`*)? What are their indices in the 8-county dataset (these are the* `row.names`*)?* *Jump to A5* •

We plot the locations of these two polygons:

```
plot(st_geometry(NY8), border="grey",
     main = "NY8 Census Tracts with max/min neighbours",
     sub = "min: red; max: green",
     axes = TRUE)
plot(st_geometry(NY8[ix.min.nb,]), border="black", col="red", lwd=1, add=T)
plot(st_geometry(NY8[ix.max.nb,]), border="black", col="green", lwd=1, add=T)
plot(NY8_nb, coords_centers, add=TRUE,
     pch=19, cex = 0.6, col="black", lwd=0.2)
grid()
```



Most of the polygons with only one neighbour are completely inside other polygons – these are villages within towns.

4.2 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. Here we do not yet have a spatial process to evaluate, so we choose an arbitrary distance.

Task 10 : Find the neighbours of each polygon within 12 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.

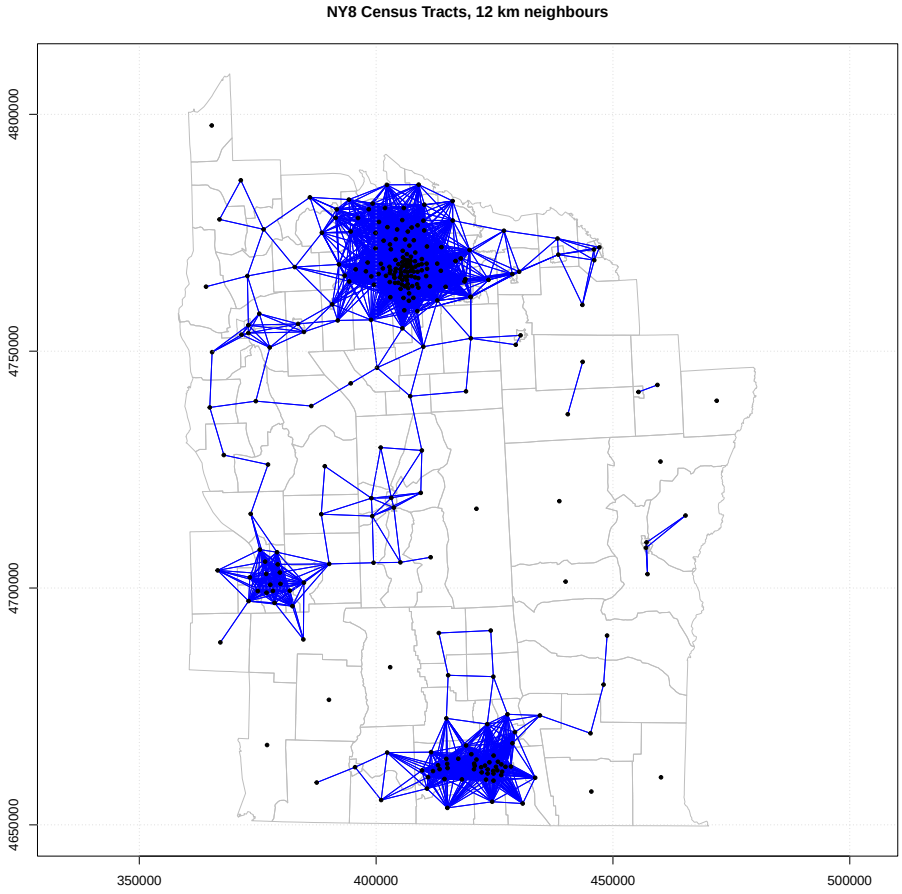
```
NY8_nb_d <- dnearneigh(coords_centers, d1=0, d2=12000,
                       row.names=row.names(NY8))
```

```
summary(NY8_nb_d)

## Neighbour list object:
## Number of regions: 281
## Number of nonzero links: 13102
## Percentage nonzero weights: 16.593
## Average number of links: 46.62633
## 11 regions with no links:
## 30 31 56 74 75 80 83 109 252 253 258
## 16 disjoint connected subgraphs
## Link number distribution:
##
##   0   1   2   3   4   5   6   7   8   9  10  11  14  15  16  17  18
##  11   9  16   9   9  10   2  12   2   2   2   2   3   2   3  15   3
##  19  20  21  23  24  27  29  30  31  32  33  34  35  36  37  38  39
##   2   1   2   5   1   1   2   1   4   4   4   1   3   3   2   7   5
##  40  41  42  43  44  48  54  66  69  71  72  77  79  80  81  83  85
##   3   2   6   4   1   1   1   1   1   1   1   1   2   1   1   1   2
##  86  87  88  89  90  92  93  94  95  96  97  98  99 100 101 102 103
##   1   3   2   1   1   2   3   3   1   4   3   1   4   4   5   8  14
## 104 105 106 107
##  13  10   4   4
## 9 least connected regions:
## 60 81 90 105 106 107 108 257 279 with 1 link
## 4 most connected regions:
## 113 118 119 215 with 107 links
```

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

Task 11 : Plot the polygon map with these neighbour links. •

```
plot(st_geometry(NY8), border="grey",
     main = "NY8 Census Tracts, 12 km neighbours", axes = TRUE)
plot(NY8_nb_d, coords_centers,
     pch=19, cex = 0.6, add=TRUE, col="blue", lwd = 0.2)
grid()
```



These are mostly in the urban areas.

4.3 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 12 : Create a neighbours list including the three nearest neighbours of each polygon. •

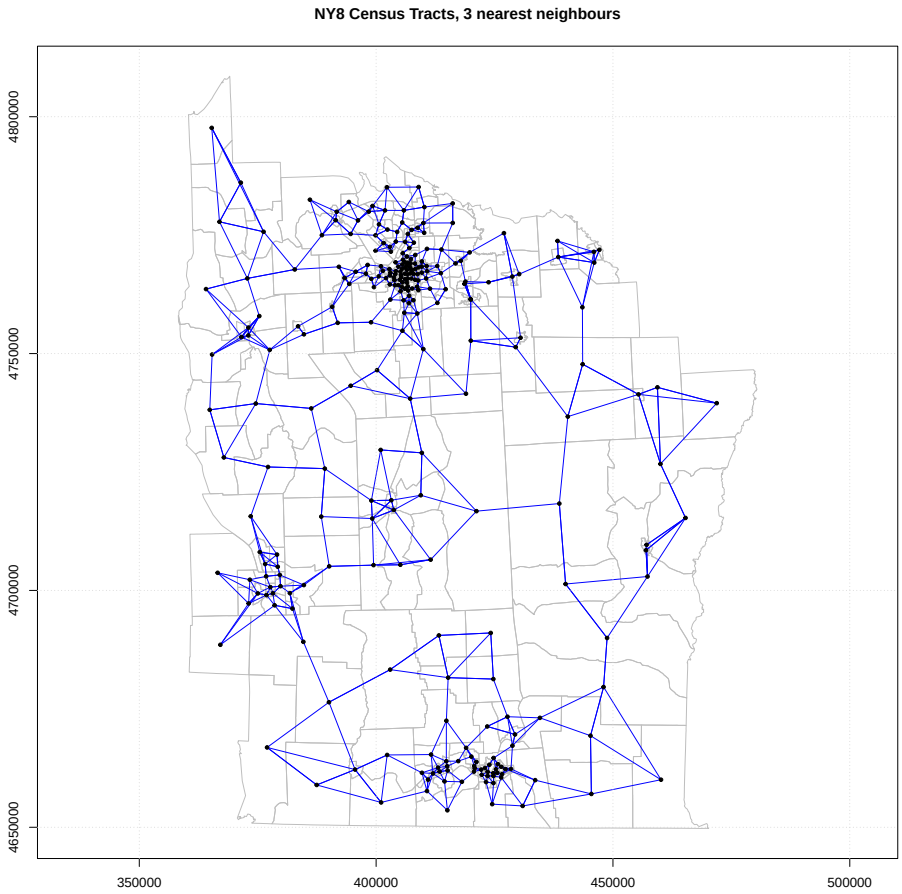
The `knearneigh` function finds the nearest neighbours as a matrix, then the `knn2nb` function converts this to a neighbours list.

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

##  int [1:281, 1:3] 2 1 2 6 10 4 6 7 18 5 ...

knn3$nn[1:3,]

##      [,1] [,2] [,3]
## [1,]    2   15   49
```

```
## [2,]    1   13    3
## [3,]    2   13   35

NY8_nb_3nn <- knn2nb(knn3, row.names=row.names(NY8))
print(NY8_nb_3nn)

## Neighbour list object:
## Number of regions: 281
## Number of nonzero links: 843
## Percentage nonzero weights: 1.067616
## Average number of links: 3
## Non-symmetric neighbours list
```

Task 13 : Plot the polygon map with these neighbour links. •

```
plot(st_geometry(NY8), border="grey",
     main = "NY8 Census Tracts, 3 nearest neighbours", axes = TRUE)
plot(NY8_nb_3nn, coords_centers, pch=19, cex=0.6, col="blue", add=TRUE)
grid()
```



5 Subsetting the dataset

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

Q6 : *What field in the dataframe of* `NY8` *gives the geographic names? How many are there? What is the name for Syracuse city?* *Jump to A6* •

```
names(NY8)
```

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

```
length(unique(NY8$AREANAME))
```

```
## [1] 64
```

```
sort(unique(NY8$AREANAME))
```

```
##  [1] Auburn city           Baldwinsville village Barker town
##  [4] Bayberry-Lynelle Mead Binghamton city       Binghamton town
##  [7] Brookfield town       Camillus village      Canastota village
## [10] Cazenovia village     Chenango town         Colesville town
## [13] Conklin town          Cortland city         East Cayuga Heights C
## [16] East Syracuse village Endicott village      Endwell CDP
## [19] Fairmount CDP         Fayetteville village  Fenton town
## [22] Hamilton village      Ithaca city           Johnson City village
## [25] Kirkwood town         Lafayette town        Lansing village
## [28] Liverpool village     Maine town            Manlius village
## [31] Marcellus village     Minoa village         NA
## [34] Nanticoke town        Newfield town         North Syracuse villag
## [37] Norwich city          Oneida city           Onondaga
## [40] Onondaga town         Otisco town           Owego village
## [43] Pitcher Hill CDP      Pompey town           Remainder of Camillus
## [46] Remainder of Cicero t Remainder of Clay tow Remainder of De Witt
## [49] Remainder of Dryden t Remainder of Geddes t Remainder of Ithaca t
## [52] Remainder of Lansing  Remainder of Lysander Remainder of Manlius
## [55] Remainder of Owego to Remainder of Salina t Remainder of Sullivan
## [58] Remainder of Union to Remainder of Van Bure Skaneateles village
## [61] Solvay village        Spafford town         Syracuse city
## [64] Vestal town
## 64 Levels: Auburn city Baldwinsville village ... Vestal town
```

```
which(NY8$AREANAME == "Syracuse city")
```

```
##  [1] 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
## [17] 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141
## [33] 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157
## [49] 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172
```

Task 14 : Extract the subset of the data covering the city of Syracuse[6] and all its neighbours within the distance radius specified in §4.2. Plot it, along with its adjacency neighbour links. •

Subset the data with a logical expression to select matrix rows for Syracuse, then find the tracts within the specified distance of any of the Syracuse tracts. Then the two together form the subset.

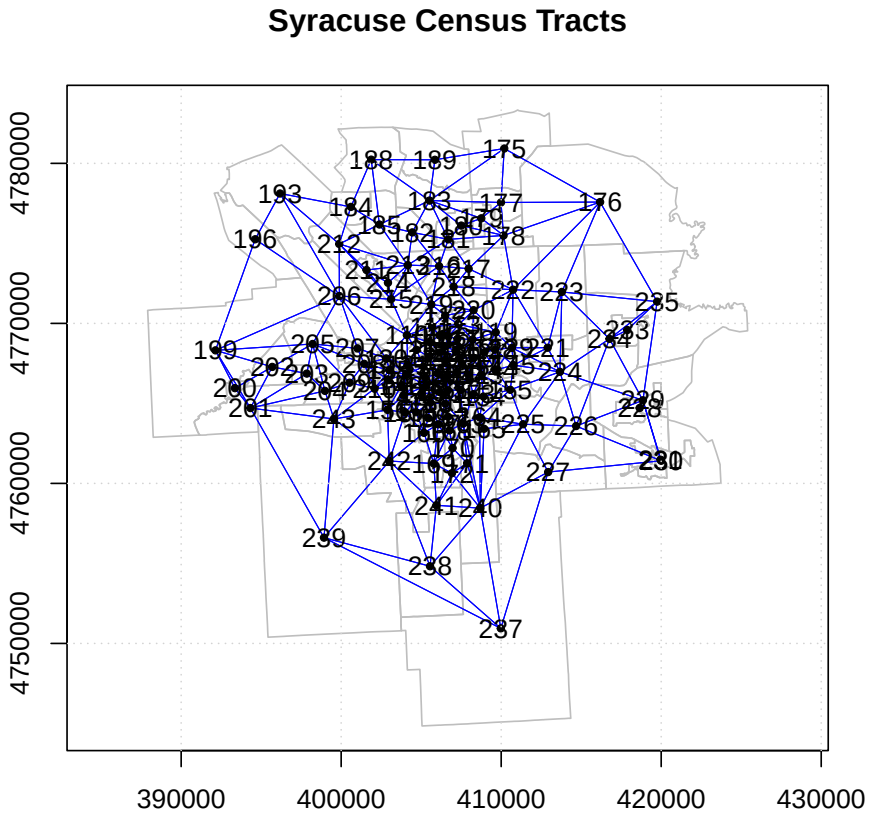
[6] https://www.syr.gov

```
(ix <- grep("Syracuse city", NY8$AREANAME, fixed = TRUE))

##  [1] 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
## [17] 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141
## [33] 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157
## [49] 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172

# find near neighbours by distance
(iy <- unique(unlist(NY8_nb_d[ix])))

##   [1] 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
##  [17] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142
##  [33] 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158
##  [49] 159 160 161 162 163 164 165 166 167 168 169 170 171 172 177 178
##  [65] 179 180 181 182 183 184 185 188 189 193 196 200 201 202 203 204
##  [81] 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
##  [97] 221 222 223 224 225 240 241 242 243 110 226 227 234 175 176 233
## [113] 228 229 235 230 231 199 239 238 237

Syr <- NY8[iz <- union(ix, iy),]
dim(Syr)

## [1] 121  18
```

```
Syr_nb <- poly2nb(Syr)
```

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(st_geometry(Syr), border="grey",
     main = "Syracuse Census Tracts", axes = TRUE)
coords_centers_Syr <- st_coordinates(st_centroid(st_geometry(Syr)))
plot(Syr_nb, coords_centers_Syr, col = "blue", lwd = 0.7,
     pch=19, cex=0.6, add=TRUE)
text(coords_centers_Syr, row.names(Syr))
grid()
```



5.1 * 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 15 : Export the Syracuse polygons in KML format and display in Google Earth. •

The `st_write` function of the `sf` package exports in many formats. For display in Google Earth, coordinates must be in Longitude/Latitude on the WGS84 ellipsoid; we use the `st_transform` method to transform from the original UTM to this system, defined by an EPSG code.

```
Syr.ll <- st_transform(Syr, 4326)
st_write(Syr.ll, "./Syr.kml", driver = "kml", delete_dsn = TRUE)
```

Opening the KML in Google Earth and adjusting the symbology, we see Figure 1.

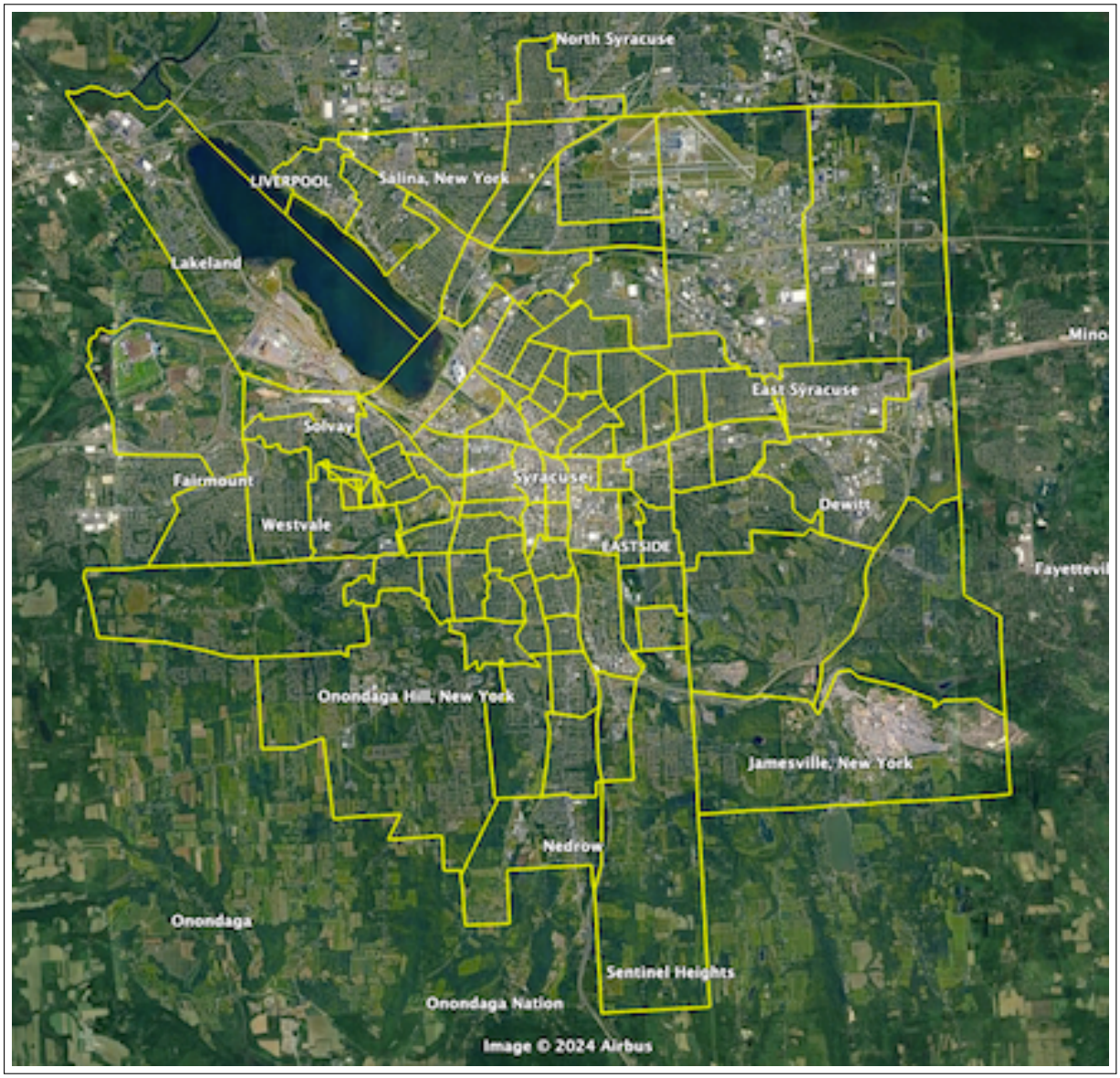


Figure 1: Syracuse area census tracts

6 Spatial weights

Supplementary reading:

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

In models where neighbours are considered, the question is how to apportion the presumed neigbourhood effect among the neighbours of a given polygon, i.e, how to assign **spatial weights**. This should correspond to the presumed process by which the neighbourhood effect occurs.

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 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.

There are many choices of weightings, which we now review.

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 16 : Create a weights object for the New York 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 §7.1.1.

6.1 Weights style W

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

```
NY8_lw_W <- nb2listw(NY8_nb)
print(NY8_lw_W)

## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 281
## Number of nonzero links: 1624
```

```
## Percentage nonzero weights: 2.056712
## Average number of links: 5.779359
##
## Weights style: W
## Weights constants summary:
##     n    nn  S0       S1       S2
## W 281 78961 281 106.6125 1164.157

print(NY8_lw_W$neighbours[[1]])

## [1]  2 13 14 15 47 48 49 50

print(NY8_lw_W$weights[[1]])

## [1] 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125
```

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 8 neighbouring regions, and each has equal weight 0.125. To extract these, we used the `[[]]` list extraction operator.

Q7 : *What are the weights of each link for the polygons with minimum and maximum neighbours?* *Jump to A7* •

These are the respective lists for the two identified polygons:

```
unlist(unique(NY8_lw_W$weights[ix.min.nb]))

## [1] 1

unlist(unique(NY8_lw_W$weights[ix.max.nb]))

##  [1] 0.09090909 0.09090909 0.09090909 0.09090909 0.09090909 0.09090909
##  [7] 0.09090909 0.09090909 0.09090909 0.09090909 0.09090909
```

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 17 : Also make the weights matrix for the Syracuse area. •

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

## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 121
## Number of nonzero links: 714
## Percentage nonzero weights: 4.876716
## Average number of links: 5.900826
##
## Weights style: W
```

```
## Weights constants summary:
##     n    nn  S0      S1       S2
## W 121 14641 121 43.6013 498.1653
```

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,]

##     110   111 112 113   114 115 116 117 118 119   120 121 122 123 124
## 110   0 0.125   0   0 0.125   0   0   0   0   0 0.125   0   0   0   0
##     125 126 127 128 129   130   131 132 133 134 135 136 137 138 139
## 110   0   0   0   0   0 0.125 0.125   0   0   0   0   0   0   0   0
##     140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155
## 110   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##     156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171
## 110   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##     172 177 178 179 180 181 182 183 184 185 188 189 193 196 200 201
## 110   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##     202 203 204 205   206 207 208 209 210 211 212 213 214   215 216
## 110   0   0   0   0 0.125   0   0   0   0   0   0   0   0 0.125   0
##     217 218   219 220 221 222 223 224 225 240 241 242 243 226 227 234
## 110   0   0 0.125   0   0   0   0   0   0   0   0   0   0   0   0   0
##     175 176 233 228 229 235 230 231 199 239 238 237
## 110   0   0   0   0   0   0   0   0   0   0   0   0

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

##         110    111    112    113    114    115   116   117   118
## 110 0.0000 0.1250 0.0000 0.0000 0.1250 0.0000 0.000 0.000 0.000
## 111 0.1667 0.0000 0.1667 0.1667 0.1667 0.1667 0.000 0.000 0.000
## 112 0.0000 0.2500 0.0000 0.2500 0.0000 0.0000 0.000 0.000 0.000
## 113 0.0000 0.1250 0.1250 0.0000 0.0000 0.1250 0.125 0.125 0.125
## 114 0.1429 0.1429 0.0000 0.0000 0.0000 0.1429 0.000 0.000 0.000
## 115 0.0000 0.2000 0.0000 0.2000 0.2000 0.0000 0.200 0.000 0.000
## 116 0.0000 0.0000 0.0000 0.2500 0.0000 0.2500 0.000 0.250 0.000
## 117 0.0000 0.0000 0.0000 0.2000 0.0000 0.0000 0.200 0.000 0.200
## 118 0.0000 0.0000 0.0000 0.1250 0.0000 0.0000 0.000 0.125 0.000
```

```
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.

6.2 Other weights styles

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:

First, find the tracts with the miniumum and maximum number of neighours:

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

## n.nb.s
##  1  2  3  4  5  6  7  8  9 10
##  1  2  5 17 23 29 24 12  7  1

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

## [1] 1
## [1] 10

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

## [1] 113

Syr[ix.min.s,]

## Simple feature collection with 1 feature and 17 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 416999.9 ymin: 4763591 xmax: 420412.2 ymax: 4766288
## Projected CRS: WGS 84 / UTM zone 18N
##                 AREANAME     AREAKEY       X        Y POP8 TRACTCAS
## 228 Fayetteville village 36067015000 -1.2134 35.81025 4709     3.08
##      PROPCAS PCTOWNHOME PCTAGE65P        Z  AVGIDIST PEXPOSURE
## 228 0.000654  0.7092391 0.1299639 -0.14338 0.0232898   0.84543
##       Cases      Xm       Ym   Xshift  Yshift
## 228 3.07105 -1213.4 35810.25 418108.2 4764665
##                           geometry
## 228 POLYGON ((417172.5 4765099,...
```

```r
(ix.max.s <- which(n.nb.s==max(n.nb.s)))

## [1] 103

Syr[ix.max.s,]

## Simple feature collection with 1 feature and 17 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 407359.5 ymin: 4754983 xmax: 409878.6 ymax: 4762986
## Projected CRS: WGS 84 / UTM zone 18N
##          AREANAME     AREAKEY         X        Y POP8 TRACTCAS
## 240 Onondaga town 36067016100 -10.32513 29.87623 1214     1.02
##     PROPCAS PCTOWNHOME PCTAGE65P       Z  AVGIDIST PEXPOSURE   Cases
## 240 0.00084  0.9234234 0.1037891 0.50918 0.0272618  1.002901 1.01831
##            Xm       Ym   Xshift  Yshift
## 240 -10325.13 29876.23 408996.4 4758731
##                           geometry
## 240 POLYGON ((408376.3 4762335,...
```

```r
Syr_lw_W <- nb2listw(Syr_nb)
summary(Syr_lw_W)

## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 121
## Number of nonzero links: 714
## Percentage nonzero weights: 4.876716
## Average number of links: 5.900826
## Link number distribution:
##
##  1  2  3  4  5  6  7  8  9 10
##  1  2  5 17 23 29 24 12  7  1
## 1 least connected region:
## 228 with 1 link
## 1 most connected region:
## 240 with 10 links
##
## Weights style: W
## Weights constants summary:
##     n    nn  S0      S1       S2
## W 121 14641 121 43.6013 498.1653

Syr_lw_B <- nb2listw(Syr_nb, style="B")
summary(Syr_lw_B)

## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 121
## Number of nonzero links: 714
## Percentage nonzero weights: 4.876716
## Average number of links: 5.900826
## Link number distribution:
##
##  1  2  3  4  5  6  7  8  9 10
##  1  2  5 17 23 29 24 12  7  1
## 1 least connected region:
## 228 with 1 link
## 1 most connected region:
## 240 with 10 links
##
## Weights style: B
## Weights constants summary:
##     n    nn  S0   S1    S2
## B 121 14641 714 1428 18224

Syr_lw_C <- nb2listw(Syr_nb, style="C")
summary(Syr_lw_C)
```

```
## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 121
## Number of nonzero links: 714
## Percentage nonzero weights: 4.876716
## Average number of links: 5.900826
## Link number distribution:
##
##  1  2  3  4  5  6  7  8  9 10
##  1  2  5 17 23 29 24 12  7  1
## 1 least connected region:
## 228 with 1 link
## 1 most connected region:
## 240 with 10 links
##
## Weights style: C
## Weights constants summary:
##     n    nn  S0      S1       S2
## C 121 14641 121 41.0112 523.3811
```

```
Syr_lw_U <- nb2listw(Syr_nb, style="U")
summary(Syr_lw_U)
```

```
## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 121
## Number of nonzero links: 714
## Percentage nonzero weights: 4.876716
## Average number of links: 5.900826
## Link number distribution:
##
##  1  2  3  4  5  6  7  8  9 10
##  1  2  5 17 23 29 24 12  7  1
## 1 least connected region:
## 228 with 1 link
## 1 most connected region:
## 240 with 10 links
##
## Weights style: U
## Weights constants summary:
##     n    nn S0         S1         S2
## U 121 14641  1 0.00280112 0.03574763
```

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.

Q8 : *Considering the leukemia incidence, why or why not would the inverse-distance weighting represent the underlying process?* *Jump to A8*

•

Task 19 : 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 computed above, 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.

> **Note:** Note that the function to compute the inverse distance could also have some power. Here we just use distance itself, with linear distance decay of any hypothesized effect.

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

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

## [1] "110"

Syr_nb[[1]]

## [1]  2  5 11 21 22 83 92 96

dsts <- nbdists(Syr_nb, coords_centers_Syr)
dsts[1]

## [[1]]
## [1] 1656.873 1514.638 1098.564 1944.120 1871.600 4877.886 2454.290
## [8] 2449.660

idw <- lapply(dsts, function(x) 1/(x/1000))
# could do this, IDW^2
# idw2 <- lapply(dsts, function(x) (1/(x/1000)^2))
idw[1]

## [[1]]
## [1] 0.6035465 0.6602238 0.9102789 0.5143715 0.5343021 0.2050068
## [7] 0.4074498 0.4082199

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 0.2050068
## [7] 0.4074498 0.4082199
```

Here is the summary of the weights:

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

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.08052  0.31557  0.60062  0.70777  0.92669 12.09146
```

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.
##  0.4796  2.1198  3.6456  4.1764  6.2446 12.6601
```

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

Task 20 : **Optional**: Compare the weights matrices of the different weighting styles. •

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

```
##         110    111    112    113    114    115   116   117   118
## 110 0.0000 0.1250 0.0000 0.0000 0.1250 0.0000 0.000 0.000 0.000
## 111 0.1667 0.0000 0.1667 0.1667 0.1667 0.1667 0.000 0.000 0.000
## 112 0.0000 0.2500 0.0000 0.2500 0.0000 0.0000 0.000 0.000 0.000
## 113 0.0000 0.1250 0.1250 0.0000 0.0000 0.1250 0.125 0.125 0.125
## 114 0.1429 0.1429 0.0000 0.0000 0.0000 0.1429 0.000 0.000 0.000
## 115 0.0000 0.2000 0.0000 0.2000 0.2000 0.0000 0.200 0.000 0.000
## 116 0.0000 0.0000 0.0000 0.2500 0.0000 0.2500 0.000 0.250 0.000
## 117 0.0000 0.0000 0.0000 0.2000 0.0000 0.0000 0.200 0.000 0.200
## 118 0.0000 0.0000 0.0000 0.1250 0.0000 0.0000 0.000 0.125 0.000

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

##     110 111 112 113 114 115 116 117 118
## 110   0   1   0   0   1   0   0   0   0
## 111   1   0   1   1   1   1   0   0   0
## 112   0   1   0   1   0   0   0   0   0
## 113   0   1   1   0   0   1   1   1   1
## 114   1   1   0   0   0   1   0   0   0
## 115   0   1   0   1   1   0   1   0   0
## 116   0   0   0   1   0   1   0   1   0
## 117   0   0   0   1   0   0   1   0   1
## 118   0   0   0   1   0   0   0   1   0

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

##         110    111    112    113    114    115    116    117    118
## 110 0.0000 0.1695 0.0000 0.0000 0.1695 0.0000 0.0000 0.0000 0.0000
## 111 0.1695 0.0000 0.1695 0.1695 0.1695 0.1695 0.0000 0.0000 0.0000
## 112 0.0000 0.1695 0.0000 0.1695 0.0000 0.0000 0.0000 0.0000 0.0000
## 113 0.0000 0.1695 0.1695 0.0000 0.0000 0.1695 0.1695 0.1695 0.1695
## 114 0.1695 0.1695 0.0000 0.0000 0.0000 0.1695 0.0000 0.0000 0.0000
## 115 0.0000 0.1695 0.0000 0.1695 0.1695 0.0000 0.1695 0.0000 0.0000
## 116 0.0000 0.0000 0.0000 0.1695 0.0000 0.1695 0.0000 0.1695 0.0000
## 117 0.0000 0.0000 0.0000 0.1695 0.0000 0.0000 0.1695 0.0000 0.1695
## 118 0.0000 0.0000 0.0000 0.1695 0.0000 0.0000 0.0000 0.1695 0.0000

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

##         110    111    112    113    114    115    116    117    118
## 110 0.0000 0.0014 0.0000 0.0000 0.0014 0.0000 0.0000 0.0000 0.0000
## 111 0.0014 0.0000 0.0014 0.0014 0.0014 0.0014 0.0000 0.0000 0.0000
## 112 0.0000 0.0014 0.0000 0.0014 0.0000 0.0000 0.0000 0.0000 0.0000
## 113 0.0000 0.0014 0.0014 0.0000 0.0000 0.0014 0.0014 0.0014 0.0014
## 114 0.0014 0.0014 0.0000 0.0000 0.0000 0.0014 0.0000 0.0000 0.0000
## 115 0.0000 0.0014 0.0000 0.0014 0.0014 0.0000 0.0014 0.0000 0.0000
## 116 0.0000 0.0000 0.0000 0.0014 0.0000 0.0014 0.0000 0.0014 0.0000
## 117 0.0000 0.0000 0.0000 0.0014 0.0000 0.0000 0.0014 0.0000 0.0014
## 118 0.0000 0.0000 0.0000 0.0014 0.0000 0.0000 0.0000 0.0014 0.0000

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

##         110    111    112    113    114    115    116    117    118
## 110 0.0000 0.6035 0.0000 0.0000 0.6602 0.0000 0.0000 0.0000 0.0000
## 111 0.6035 0.0000 0.9265 0.5963 1.0111 1.4139 0.0000 0.0000 0.0000
## 112 0.0000 0.9265 0.0000 0.9858 0.0000 0.0000 0.0000 0.0000 0.0000
## 113 0.0000 0.5963 0.9858 0.0000 0.0000 0.7191 1.0020 1.1118 0.7829
## 114 0.6602 1.0111 0.0000 0.0000 0.0000 1.3676 0.0000 0.0000 0.0000
## 115 0.0000 1.4139 0.0000 0.7191 1.3676 0.0000 1.7476 0.0000 0.0000
## 116 0.0000 0.0000 0.0000 1.0020 0.0000 1.7476 0.0000 1.7162 0.0000
## 117 0.0000 0.0000 0.0000 1.1118 0.0000 0.0000 1.7162 0.0000 1.0592
## 118 0.0000 0.0000 0.0000 0.7829 0.0000 0.0000 0.0000 1.0592 0.0000
```

7 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[7]-producing site[8];

`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.

[7] Trichloroethylene, an industrial solvent often found in groundwater
[8] 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.

7.1 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]:

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

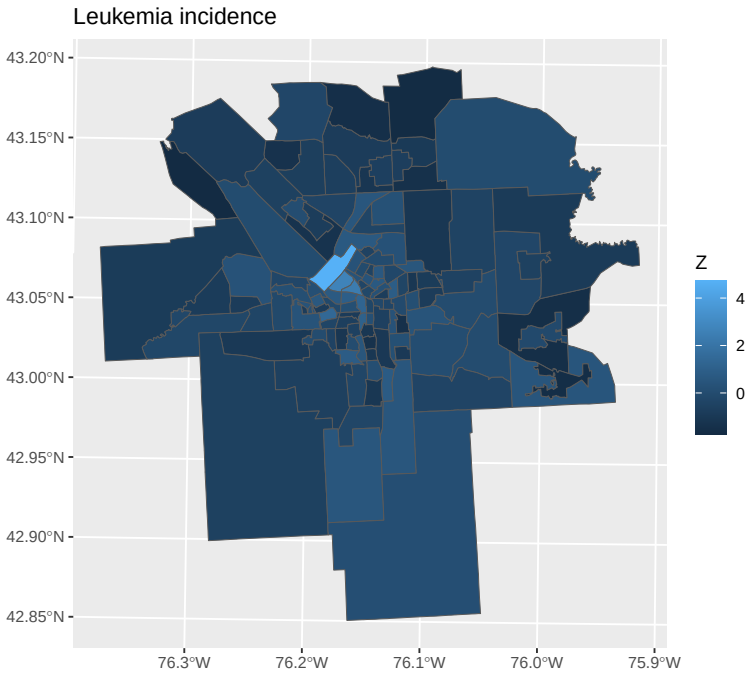
where Y_i is the count of cases in a census tract and n_i is its population. This is presented as field `Z`. We refer to this as leukemia incidence.

```
summary(Syr$Z)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.7390 -0.7994 -0.2928 -0.2059  0.2453  4.7105
```

Task 21 : Test the assumption that leukemia cases incidence is spatially independent (randomly distributed among census tracts). •

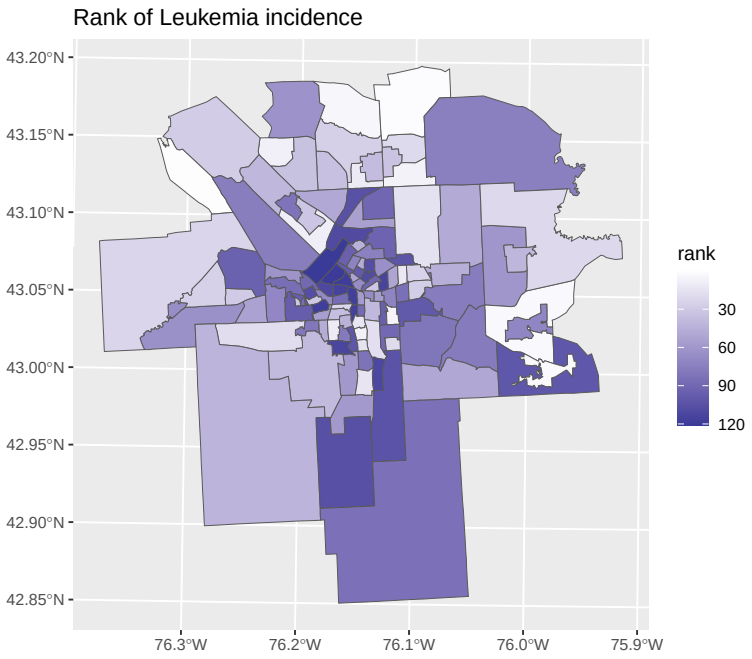
We first visualize the spatial relation with several colour-ramp plots. The first is the incidence itself in each census tract.

```
ggplot(data = Syr) +
  geom_sf(aes(fill = Z)) +
  labs(title = "Leukemia incidence")
```



The second the *rank* of the leukemia incidence in each census tract, from lowest (lightest shade) to highest (darkest).

```
rank <- rank(Syr$Z)
ggplot(data = Syr) +
  geom_sf(aes(fill = rank)) +
  scale_fill_gradient2(guide = guide_colourbar(reverse = TRUE)) +
  labs(title = "Rank of Leukemia incidence")
```

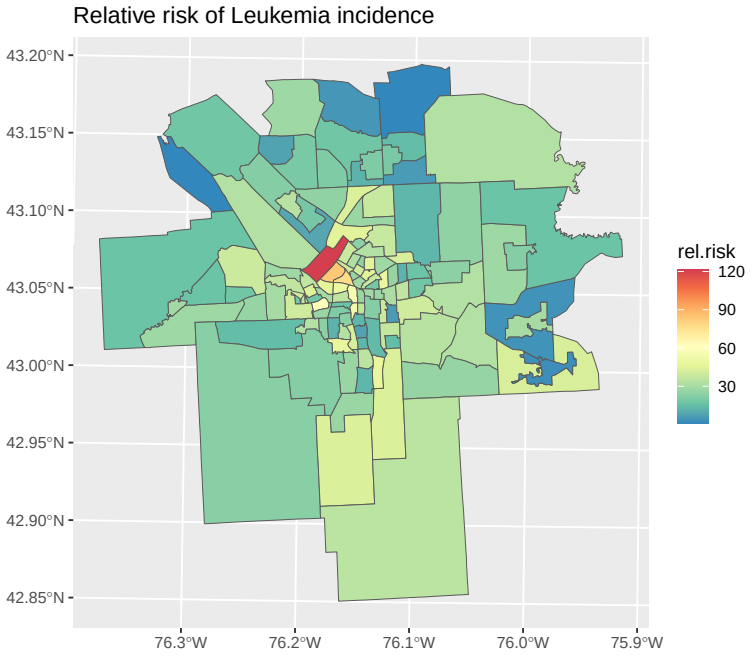


Another view is the *relative intensity* of incidence, shown with the same grey scale, but with the intensity of the grey proportional to the maximum proportion of cases.

> **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.

```
rel.risk  <- ((Syr$Z-min(Syr$Z))/(max(Syr$Z)-min(Syr$Z)))
n <- length(Syr$Z)
rel.risk <- pmax(ceiling(n*rel.risk), 1)
rel.risk.pal <- rev(brewer.pal(9, 'Spectral'))
ggplot(data = Syr) +
  geom_sf(aes(fill = rel.risk)) +
  scale_fill_gradientn(colors = rel.risk.pal) +
  labs(title = "Relative risk of Leukemia incidence")
```



Q9 : *Does leukemia incidence appear to be spatially autocorrelated? Which of the maps shows this best?* *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[9].

```
(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 = 4.9597, p-value =
## 3.531e-07
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.246563167      -0.008333333       0.002641332
```

Q10 : *Is leukemia incidence provably spatially autocorrelated with this weighting?* *Jump to A10* •

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

7.1.1 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.

We computed several weightings in §6.2, here we see their effect on Moran's I.

Task 22 : Re-compute Moran's I with inverse-distance weighting. •

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

```
print(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 = 3.0215, p-value =
## 0.001257
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.210832392      -0.008333333       0.005261246

print(moran.z)

##
##  Moran I test under randomisation
##
## data:  Syr$Z
## weights: Syr_lw_W
##
## Moran I statistic standard deviate = 4.9597, p-value =
## 3.531e-07
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.246563167      -0.008333333       0.002641332

# (moran.pctage65p.idwB <- moran.test(Syr$PCTAGE65P, NY8_lw_idwB))
```

Q11 : *How did the probabilities of Type I error to reject the null hypothesis of no association change with this weighting?* *Jump to A11* •

7.2 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.

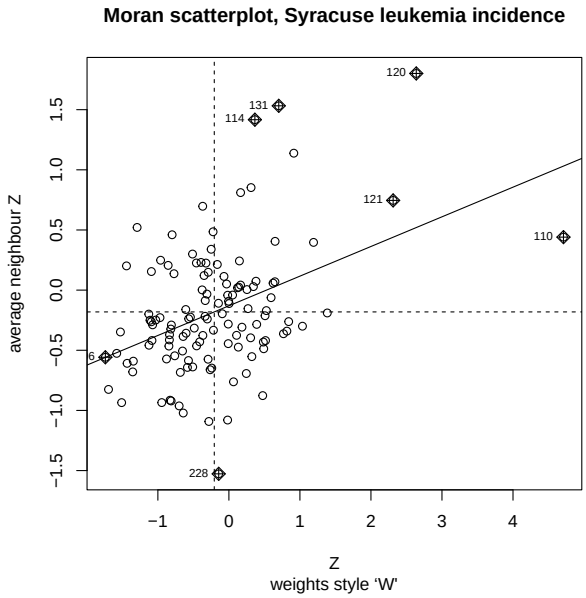
7.2.1 Local Moran's I

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 is `TRUE` if *any* of the six measures of each observation's have high influence on the plotted regression line. These six measures are computed by the `influence.measures` method, which is often applied to linear models. See the help text for this function for their interpretation.

```
str(mp)

## 'data.frame': 121 obs. of  10 variables:
```

```
##  $ x     : num  4.7105 0.312 -0.512 -0.0322 0.3659 ...
##  $ wx    : num  0.4405 0.8523 0.2998 0.0507 1.4167 ...
##  $ is_inf: logi  TRUE FALSE FALSE FALSE TRUE FALSE ...
##  $ labels: chr  "110" "111" "112" "113" ...
##  $ dfb.1_: num  -0.3526 0.1957 0.0955 0.0368 0.3272 ...
##  $ dfb.x : num  -0.87604 0.10488 -0.03764 0.00722 0.19124 ...
##  $ dffit : num  -0.8897 0.2051 0.1135 0.0369 0.3482 ...
##  $ cov.r : num  1.348 0.967 1.003 1.023 0.873 ...
##  $ cook.d: num  0.392065 0.020561 0.006416 0.000684 0.056316 ...
##  $ hat   : num  0.27206 0.01119 0.00929 0.00859 0.01183 ...
##  - attr(*, "xname")= chr "Syr$Z"

mp[1,]

##         x      wx is_inf labels     dfb.1_      dfb.x      dffit
## 1 4.71053 0.44048   TRUE    110 -0.3526264 -0.8760432 -0.8896601
##      cov.r    cook.d       hat
## 1 1.348297 0.3920649 0.2720624
```

Task 25 : Identify the influential observations, i.e., the tracts that most influence the global Moran's I. •

```
ix.infl <- which(infl <- mp$is_inf)
mp[ix.infl, ]

##            x         wx is_inf labels        dfb.1_         dfb.x
## 1    4.71053  0.4404800   TRUE    110 -0.3526264060 -0.8760432122
## 5    0.36591  1.4166557   TRUE    114  0.3272464861  0.1912421149
## 11   2.63806  1.8010020   TRUE    120  0.4798323535  0.9087075928
## 12   2.31264  0.7457357   TRUE    121  0.1032011152  0.1821695904
## 22   0.70212  1.5321171   TRUE    131  0.3638244747  0.3128714827
## 76  -1.73903 -0.5582267   TRUE    196  0.0001076495 -0.0003342337
## 113 -0.14338 -1.5270700   TRUE    228 -0.2666944805 -0.0193683985
##             dffit     cov.r       cook.d         hat
## 1   -0.8896600597 1.3482965 3.920649e-01 0.272062396
## 5    0.3482403952 0.8729336 5.631629e-02 0.011833158
## 11   0.9502949231 0.9802453 4.249224e-01 0.096536251
## 12   0.1927370262 1.0942341 1.866125e-02 0.077491169
## 22   0.4333468506 0.8700929 8.682458e-02 0.017263218
## 76   0.0003843228 1.0527249 7.447788e-08 0.033915838
## 113 -0.2701717847 0.8893430 3.427467e-02 0.008307156

print(cbind(ix.infl, Syr[ix.infl, c("AREAKEY","Z")]))

## Simple feature collection with 7 features and 3 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 392069.5 ymin: 4763591 xmax: 420412.2 ymax: 4778175
## Projected CRS: WGS 84 / UTM zone 18N
##     ix.infl     AREAKEY        Z                       geometry
## 110       1 36067000100  4.71053 POLYGON ((402409.2 4768615,...
## 114       5 36067000500  0.36591 POLYGON ((404918.3 4769246,...
## 120      11 36067001100  2.63806 POLYGON ((403875.9 4768084,...
## 121      12 36067001200  2.31264 POLYGON ((405187.3 4768087,...
## 131      22 36067002100  0.70212 POLYGON ((403479.2 4767678,...
## 196      76 36067011800 -1.73903 POLYGON ((392069.5 4778135,...
## 228     113 36067015000 -0.14338 POLYGON ((417172.5 4765099,...
```

Here are the neighbours that are influenced by the first influential observation in the list:

```
print(cbind(ix.infl[1], Syr[ix.infl[1], c("AREAKEY","Z")]))

## Simple feature collection with 1 feature and 3 fields
## Geometry type: POLYGON
## Dimension:     XY
```

```
## Bounding box:  xmin: 402303.2 ymin: 4767678 xmax: 405768.7 ymax: 4771050
## Projected CRS: WGS 84 / UTM zone 18N
##     ix.infl.1.     AREAKEY       Z                       geometry
## 110          1 36067000100 4.71053 POLYGON ((402409.2 4768615,...

Syr[Syr_lw_W$neighbours[ix.infl][[1]],c("AREAKEY","Z")]

## Simple feature collection with 8 features and 2 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 395980.6 ymin: 4766831 xmax: 407482.7 ymax: 4775489
## Projected CRS: WGS 84 / UTM zone 18N
##         AREAKEY        Z                       geometry
## 111 36067000200  0.31195 POLYGON ((405510.2 4770238,...
## 114 36067000500  0.36591 POLYGON ((404918.3 4769246,...
## 120 36067001100  2.63806 POLYGON ((403875.9 4768084,...
## 130 36067002000  0.16464 POLYGON ((402278.8 4768606,...
## 131 36067002100  0.70212 POLYGON ((403479.2 4767678,...
## 206 36067012800 -0.01696 POLYGON ((395980.6 4775397,...
## 215 36067013700 -1.29225 POLYGON ((400926.7 4773091,...
## 219 36067014100  0.65037 POLYGON ((404576.9 4771617,...
```

Q12 : *Which areas strongly influenced the global Moran's* I *line? Are these high-influence area neighbours?* *Jump to A12* •

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)
summary(lm1)

##        Ii                E.Ii                Var.Ii
##  Min.   :-1.04282   Min.   :-2.660e-01   Min.   :0.0000292
##  1st Qu.:-0.06202   1st Qu.:-7.549e-03   1st Qu.:0.0120096
##  Median : 0.04088   Median :-2.883e-03   Median :0.0568500
##  Mean   : 0.24656   Mean   :-8.333e-03   Mean   :0.1603903
##  3rd Qu.: 0.21858   3rd Qu.:-4.802e-04   3rd Qu.:0.1475865
##  Max.   : 7.53723   Max.   :-1.510e-06   Max.   :2.7793308
##       Z.Ii          Pr(z != E(Ii))
##  Min.   :-2.4991   Min.   :0.0000
##  1st Qu.:-0.4792   1st Qu.:0.1958
##  Median : 0.3526   Median :0.4330
##  Mean   : 0.4049   Mean   :0.4476
##  3rd Qu.: 0.8843   3rd Qu.:0.6976
##  Max.   : 5.5380   Max.   :0.9975

ix <- which(lm1[,"Pr(z != E(Ii))"] < 0.05)
print(cbind(Syr[ix,c("AREAKEY","POP8","Z")],ix))
```

```
## Simple feature collection with 16 features and 4 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 399576 ymin: 4766144 xmax: 422725.2 ymax: 4782255
## Projected CRS: WGS 84 / UTM zone 18N
## First 10 features:
##         AREAKEY POP8        Z ix                       geometry
## 110 36067000100    9  4.71053  1 POLYGON ((402409.2 4768615,...
## 111 36067000200 3704  0.31195  2 POLYGON ((405510.2 4770238,...
## 114 36067000500 1401  0.36591  5 POLYGON ((404918.3 4769246,...
## 120 36067001100  143  2.63806 11 POLYGON ((403875.9 4768084,...
## 121 36067001200   99  2.31264 12 POLYGON ((405187.3 4768087,...
## 122 36067001300 1475 -0.36886 13 POLYGON ((405835.2 4768412,...
## 130 36067002000 2587  0.16464 21 POLYGON ((402278.8 4768606,...
## 131 36067002100 1997  0.70212 22 POLYGON ((403479.2 4767678,...
## 132 36067002200 1211  0.91381 23 POLYGON ((404684.8 4767683,...
## 133 36067002300 2549 -0.22275 24 POLYGON ((406432.7 4768037,...
```

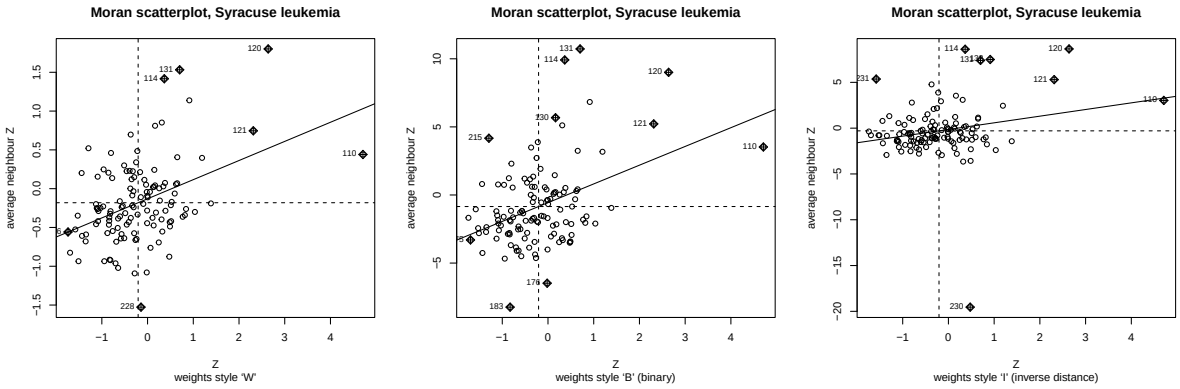
Q13 : *Is there evidence of local clustering? Could you interpret this from the Moran scatterplot?* *Jump to A13* •

7.2.2 Effect of weights

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

We have the already computed the required weight matrices (§7.1.1), 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 = 4.9597, p-value =
## 3.531e-07
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.246563167      -0.008333333       0.002641332

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

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.04282 -0.06202  0.04088  0.24656  0.21858  7.53723

(ix <- which(lm1[,"Pr(z != E(Ii))"] < 0.05))

## 110 111 114 120 121 122 130 131 132 133 139 183 188 215 219 176
##   1   2   5  11  12  13  21  22  23  24  30  70  73  92  96 111

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

##
##  Moran I test under randomisation
##
```

```
## data:  Syr$Z
## weights: Syr_lw_B
##
## Moran I statistic standard deviate = 5.2079, p-value =
## 9.55e-08
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.250184887      -0.008333333       0.002464094

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

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -8.3425 -0.3101  0.2537  1.4763  1.2583 37.6861

(ix <- which(lm1[,"Pr(z != E(Ii))"] < 0.05))

## 110 111 114 120 121 122 130 131 132 133 139 183 188 215 219 176
##   1   2   5  11  12  13  21  22  23  24  30  70  73  92  96 111

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

##
##  Moran I test under randomisation
##
## data:  Syr$Z
## weights: Syr_lw_idwB
##
## Moran I statistic standard deviate = 3.0215, p-value =
## 0.001257
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.210832392      -0.008333333       0.005261246

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

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -15.2502  -0.2104   0.1048   0.8805   0.7373  36.2245

(ix <- which(lm1[,"Pr(z != E(Ii))"] < 0.05))

## 110 111 114 120 121 122 130 131 132 133 183 188 193 176
##   1   2   5  11  12  13  21  22  23  24  70  73  75 111

##
```

Q14 : *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 A14* •

7.2.3 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 §6 did *not* include self-weights, so the statistic computed by the `localG` function, using these weights, is G_i:

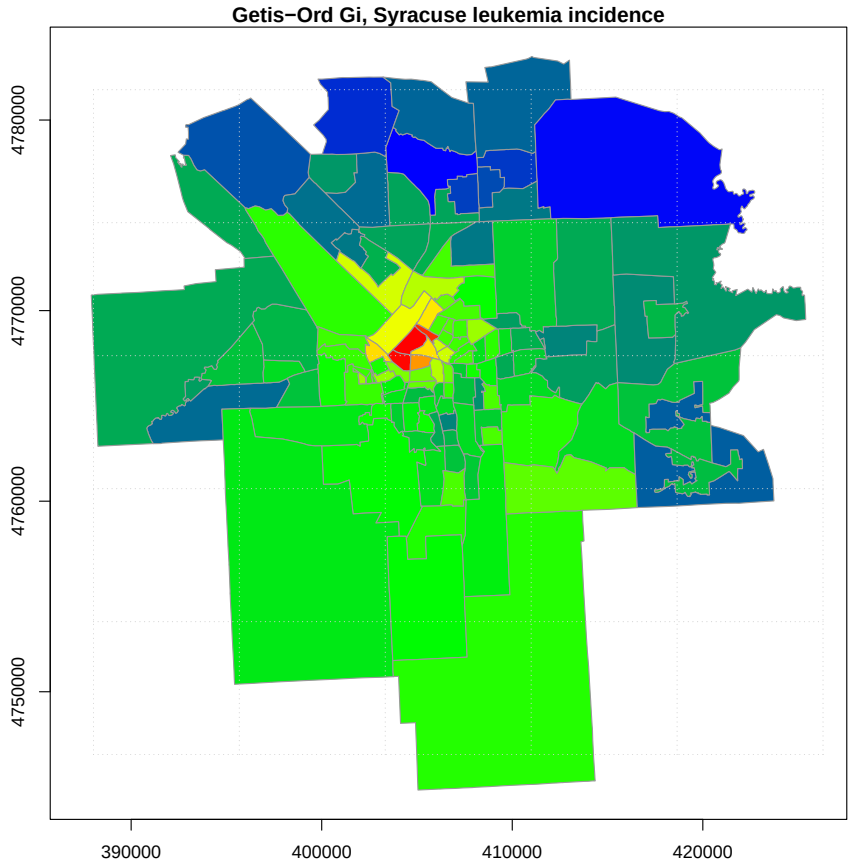
```
summary(gi <- localG(Syr$Z, Syr_lw_W))

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.5450 -0.7591 -0.1495  0.1412  0.8125  5.5380
```

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
colfunc <- colorRampPalette(c("blue", "green", "yellow", "red"))
ramp <- colfunc(n+1)
plot(Syr["Z"], border="grey60", axes=TRUE,
     col=ramp[shade],
     main="Getis-Ord Gi, Syracuse leukemia incidence")
grid()
text(coords_centers_Syr, as.character(round(gi,2)), col="black")
```



Q15 : *Where are the clusters? How does this map compare to the local Moran's I map?* *Jump to A15* •

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: 121
## Number of nonzero links: 714
## Percentage nonzero weights: 4.876716
## Average number of links: 5.900826
##
## Weights style: W
## Weights constants summary:
```

```
##     n    nn  S0      S1       S2
## W 121 14641 121 43.6013 498.1653

print(Syr_lw_W$weights[[1]])

## [1] 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125

print(Syr_lw_Wi)

## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 121
## Number of nonzero links: 835
## Percentage nonzero weights: 5.703162
## Average number of links: 6.900826
##
## Weights style: W
## Weights constants summary:
##     n    nn  S0     S1       S2
## W 121 14641 121 36.775 490.4037

print(Syr_lw_Wi$weights[[1]])

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

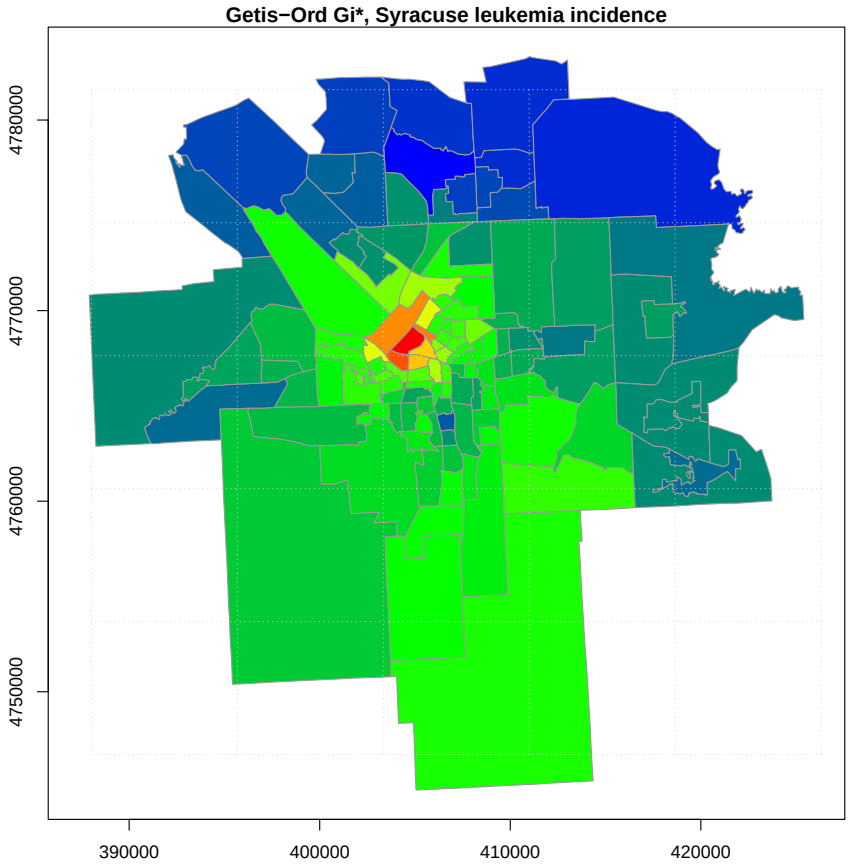
Q16 : *What is the difference between the weights list with and without including the target area?* *Jump to A16* •

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.
## -2.6622 -0.8901 -0.1250  0.1289  0.7777  6.5118

summary(gi)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.5450 -0.7591 -0.1495  0.1412  0.8125  5.5380

summary(gi.star-gi)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.83477 -0.23347 -0.04917 -0.01227  0.18704  2.14370
```

Q17 : *What are the differences between G_i^* (including the target area's value in the index) and G_i (not)?* *Jump to A17* •

```
## these from plot-gi, above
# n <- length(Syr$Z)
# colfunc <- colorRampPalette(c("blue", "green", "yellow", "red"))
# ramp <- colfunc(n+1)
shade <- as.numeric(round(n*((gi.star-min(gi.star))/
                             (max(gi.star)-min(gi.star)))))+1
plot(Syr["Z"], border="grey60", axes=TRUE,
     col=ramp[shade],
     main="Getis-Ord Gi*, Syracuse leukemia incidence")
grid()
text(coords_centers_Syr, as.character(round(gi.star,2)), col="black")
```



Q18 : *What are the differences between the G_i and G_i^* maps?* *Jump to A18* •

8 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 (i.e., possibly spatiallt-correlated random fluctuations around zero) is not correct. This is called **model mis-specification**. 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.

Q19 : *What could be some spatially-distributed causes of leukemia?* *Jump to A19* •

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:

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

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` (exposure), `PCTAGE65P` (proportion of older residents), and `PCTOWNHOME` (proportion of households that own their houses); see the beginning of §7 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.

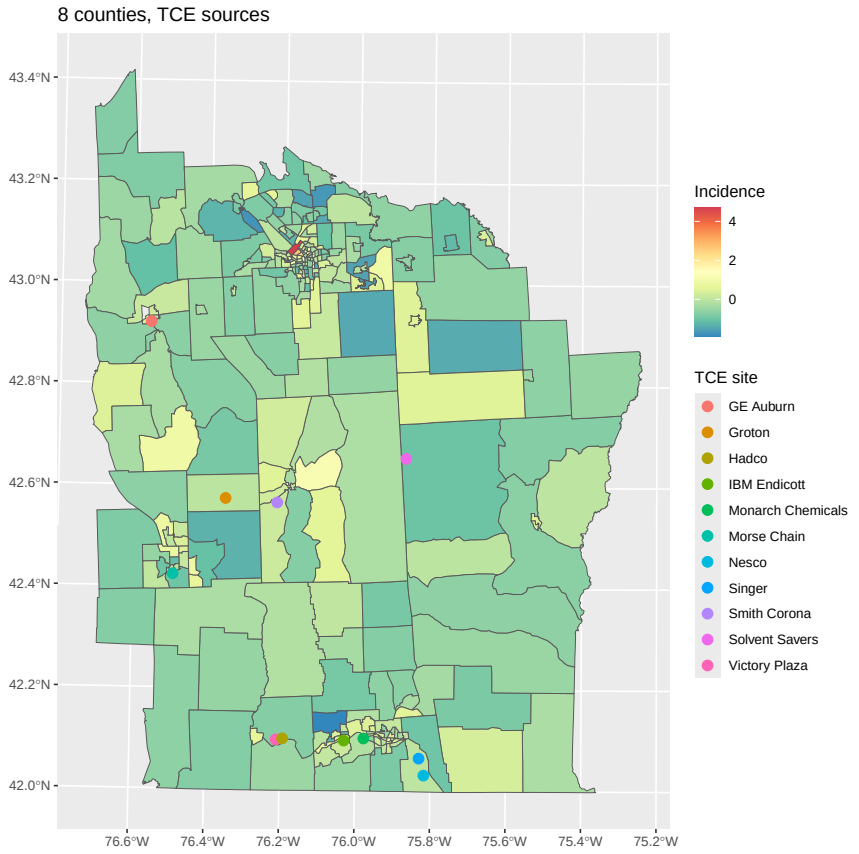
We return to the full 8-county area, because the TCE sources are spread throughout; Syracuse city is too small to have substantially different distance to TCE sources.

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 `st_read` to import the points shapefile:

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

```
ggplot(data = NY8) +
  geom_sf(aes(fill = Z)) +
  scale_fill_gradientn(colors = rel.risk.pal) +
  geom_sf(data = TCE, aes(col = name), cex=3) +
  labs(title = "8 counties, TCE sources",
       fill = "Incidence",
       col = "TCE site")
```



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: an index of TCE exposure (`PEXPOSURE`), proportion of population older than 65 years (`PCTAGE65P`), and the proportion of home ownership (`PCTOWNHOME`).

•

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

```
summary(NY8)

##                   AREANAME     AREAKEY                X
##  NA                   : 83   Length:281         Min.   :-55.482
##  Syracuse city        : 63   Class :character   1st Qu.:-19.460
##  Binghamton city      : 18   Mode  :character   Median :-12.469
##  Remainder of Clay tow:  6                      Mean   :-11.309
##  Johnson City village :  5                      3rd Qu.: -1.213
##  Onondaga town        :  5                      Max.   : 53.509
##  (Other)              :101
##        Y                POP8          TRACTCAS        PROPCAS
##  Min.   :-75.29   Min.   :    9   Min.   :0.000   Min.   :0.0000000
##  1st Qu.:-30.60   1st Qu.: 2510   1st Qu.:0.310   1st Qu.:0.0000930
##  Median : 31.97   Median : 3433   Median :1.890   Median :0.0004130
##  Mean   :  4.98   Mean   : 3764   Mean   :2.107   Mean   :0.0005947
##  3rd Qu.: 39.12   3rd Qu.: 4889   3rd Qu.:3.080   3rd Qu.:0.0009170
##  Max.   : 56.41   Max.   :13015   Max.   :9.290   Max.   :0.0069930
```

```
##
##    PCTOWNHOME          PCTAGE65P              Z
##  Min.   :0.0008224   Min.   :0.004044   Min.   :-1.9206
##  1st Qu.:0.4588745   1st Qu.:0.099926   1st Qu.:-0.7168
##  Median :0.6508585   Median :0.126415   Median :-0.2876
##  Mean   :0.5872621   Mean   :0.137262   Mean   :-0.2157
##  3rd Qu.:0.7560976   3rd Qu.:0.160963   3rd Qu.: 0.2498
##  Max.   :1.0000000   Max.   :0.505050   Max.   : 4.7105
##
##     AVGIDIST         PEXPOSURE          Cases
##  Min.   :0.01847   Min.   :0.6134   Min.   :0.00014
##  1st Qu.:0.02703   1st Qu.:0.9942   1st Qu.:0.30928
##  Median :0.03238   Median :1.1749   Median :1.88876
##  Mean   :0.14919   Mean   :1.8042   Mean   :2.10676
##  3rd Qu.:0.13008   3rd Qu.:2.5656   3rd Qu.:3.08284
##  Max.   :3.52637   Max.   :5.8654   Max.   :9.28601
##
##        Xm               Ym             Xshift           Yshift
##  Min.   :-55482   Min.   :-75291   Min.   :363839   Min.   :4653564
##  1st Qu.:-19460   1st Qu.:-30601   1st Qu.:399862   1st Qu.:4698254
##  Median :-12469   Median : 31970   Median :406852   Median :4760825
##  Mean   :-11309   Mean   :  4980   Mean   :408013   Mean   :4733835
##  3rd Qu.: -1213   3rd Qu.: 39123   3rd Qu.:418108   3rd Qu.:4767978
##  Max.   : 53509   Max.   : 56410   Max.   :472830   Max.   :4785265
##
##           geometry
##  MULTIPOLYGON :  3
##  POLYGON      :278
##  epsg:NA      :  0
##  +proj=utm ...:  0
##
##
##
```

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

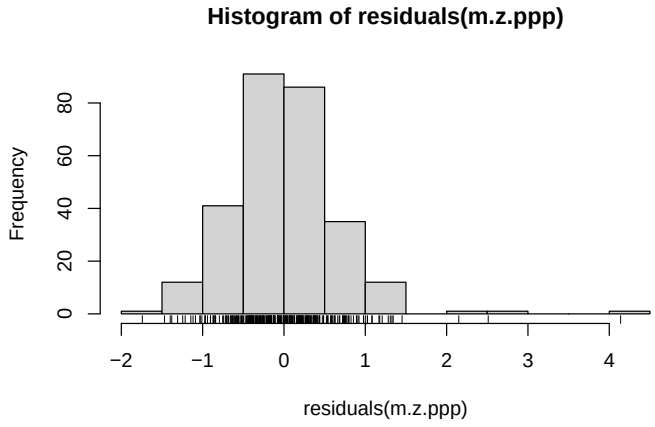
```
##
## Call:
## lm(formula = Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data = NY8)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7417 -0.3957 -0.0326  0.3353  4.1398
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.51728    0.15856  -3.262  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
```

Q20 : *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 A20* •

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[ix,]

## Simple feature collection with 1 feature and 17 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 402303.2 ymin: 4767678 xmax: 405768.7 ymax: 4771050
## Projected CRS: WGS 84 / UTM zone 18N
##          AREANAME     AREAKEY        X       Y POP8 TRACTCAS PROPCAS
## 110 Syracuse city 36067000100 -15.3264 40.5083    9        0       0
##     PCTOWNHOME PCTAGE65P       Z  AVGIDIST PEXPOSURE   Cases       Xm
## 110        0.5 0.3333333 4.71053 0.0284377  1.045131 0.00014 -15326.4
##          Ym   Xshift  Yshift                       geometry
## 110 40508.3 403995.2 4769363 POLYGON ((402409.2 4768615,...
```



This is our old friend, tract 110.

Q21 : *Why is this residual so extreme?* *Jump to A21* •

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 §6).

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)
```

```
## weights: NY8listwB
##
## Moran I statistic standard deviate = 2.3844, p-value =
## 0.008554
## alternative hypothesis: greater
## sample estimates:
## Observed Moran I      Expectation         Variance
##      0.071381909     -0.009884823      0.001161659
```

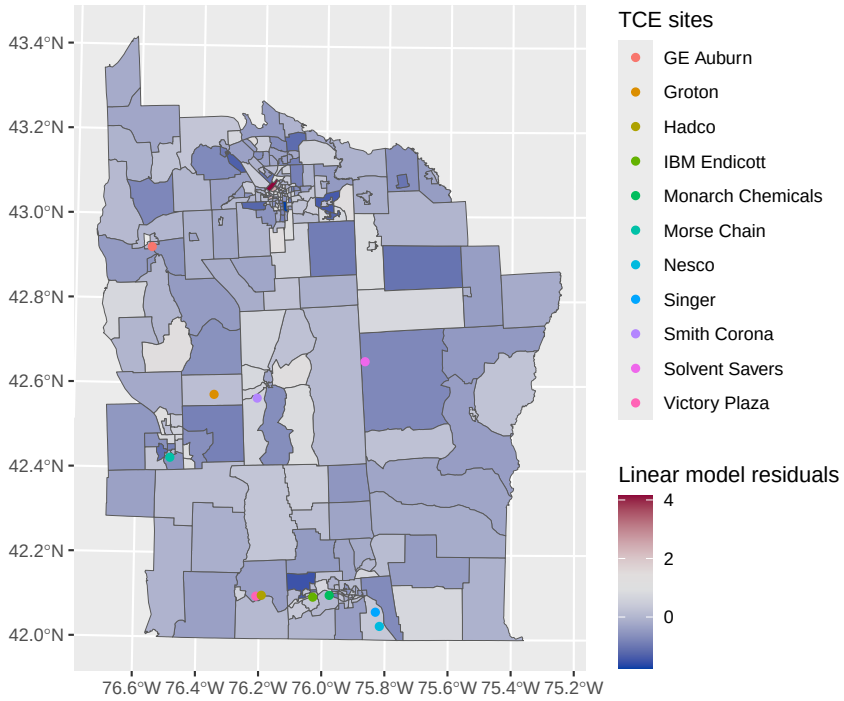
Q22 : *Is there evidence that the residuals are spatially correlated?* *Jump to A22* •

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

ggplot(data = NY8) +
  geom_sf(aes(fill = lmresid)) +
  labs(fill = "Linear model residuals", main = "Central NY State") +
  scale_fill_gradientn(colors = colorspace::diverge_hcl(12)) +
  geom_sf(data = TCE, aes(col = name)) +
  labs(col = "TCE sites")
```



Q23 : *Is there visual evidence that the *residuals* are spatially correlated? What does this suggest about our model?* *Jump to A23* •

9 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.

9.1 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:

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

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 §6. This depends

on the neighbour list and a weighting model; we have already build one using binary weights as `NY8listwB`. The autoregression term (λW) multiples the linear model residuals ($y - X^T\beta$).

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 $\Sigma_\varepsilon = \sigma^2 I$, i.e., no correlation between these errors with equal variance, the variance of the response variable Y can be written:

$$\text{Var}(y) = \sigma^2(\text{I} - \lambda\text{W})^{-1})(I - \lambda\text{W}^T)^{-1} \tag{8}$$

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

##
## Call:
## spatialreg::spautolm(formula = Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
##     data = NY8, listw = NY8listwB)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.58367 -0.38764 -0.03025  0.34023  4.02902
##
## Coefficients:
##              Estimate Std. Error z value  Pr(>|z|)
## (Intercept) -0.598779   0.174905 -3.4234 0.0006183
## PEXPOSURE    0.065148   0.041639  1.5646 0.1176753
## PCTAGE65P    3.756876   0.623907  6.0215 1.728e-09
## PCTOWNHOME  -0.429711   0.190201 -2.2592 0.0238683
##
## Lambda: 0.03573 LR test value: 4.2281 p-value: 0.03976
## Numerical Hessian standard error of lambda: 0.016955
##
## Log likelihood: -276.6148
## ML residual variance (sigma squared): 0.41606, (sigma: 0.64503)
## Number of observations: 281
## Number of parameters estimated: 6
## AIC: 565.23
```

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.

Q24 : *How did the coefficients for the three predictive factors, and their significance, change from the model that did not include simultaneous autoregression?* *Jump to A24*

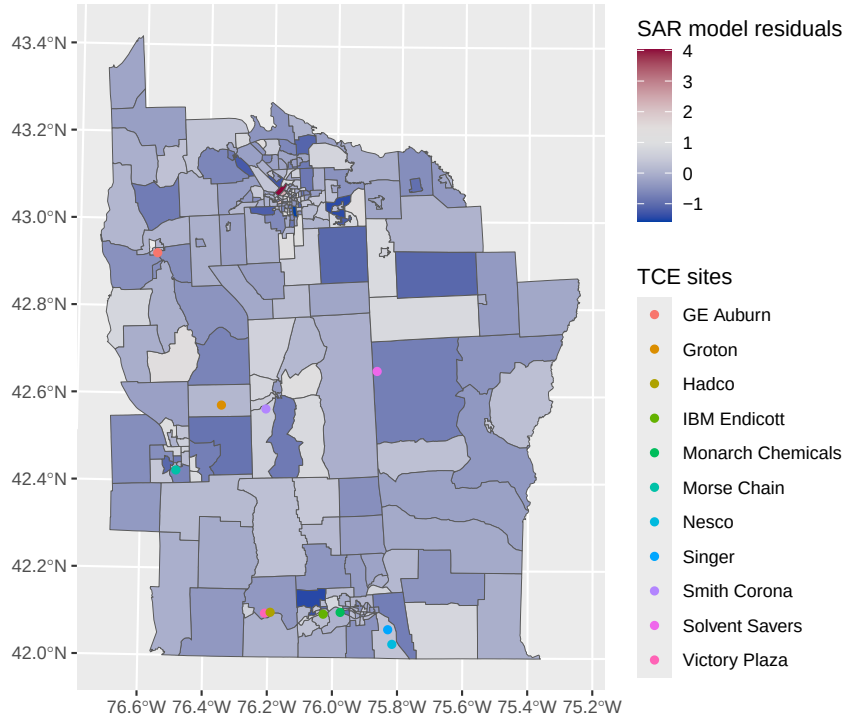
•

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)  -0.5988   0.0006
## PEXPOSURE     0.0651   0.1177
## PCTAGE65P     3.7569   0.0000
## PCTOWNHOME   -0.4297   0.0239
```

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.
## -1.583670 -0.387639 -0.030250  0.000789  0.340226  4.029025

ggplot(data = NY8) +
  geom_sf(aes(fill = sarresid)) +
  labs(fill = "SAR model residuals", main = "Central NY State") +
  scale_fill_gradientn(colors = colorspace::diverge_hcl(12)) +
  geom_sf(data = TCE, aes(col = name)) +
  labs(col = "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:

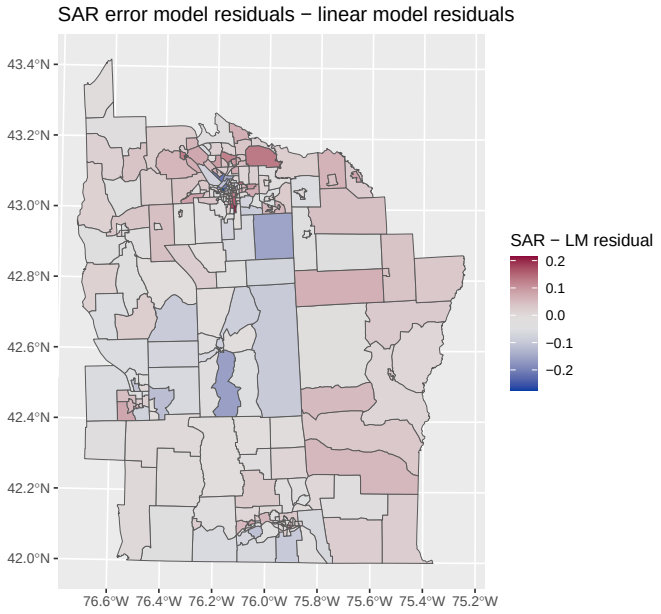
```
moran.test(NY8$sarresid, NY8listwB)

##
##  Moran I test under randomisation
##
## data:  NY8$sarresid
## weights: NY8listwB
##
## Moran I statistic standard deviate = -0.032431, p-value =
## 0.5129
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      -0.004684626      -0.003571429       0.001178181
```

Q25 : *Did the simultaneous autoregressive model account for all the spatial autocorrelation in leukemia incidence?* *Jump to A25* •

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.
## -0.2750716 -0.0298531  0.0045675  0.0007889  0.0366336  0.2148055

ggplot(data = NY8) +
  geom_sf(aes(fill = resid.change)) +
```

```
scale_fill_gradientn(colors = colorspace::diverge_hcl(12)) +
labs(title="SAR error model residuals - linear model residuals",
     fill = 'SAR - LM residual')
```



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**, i.e., "deterministic", and **stochastic** components according to the model formula of Equation 7, which we repeat here for convenience:

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

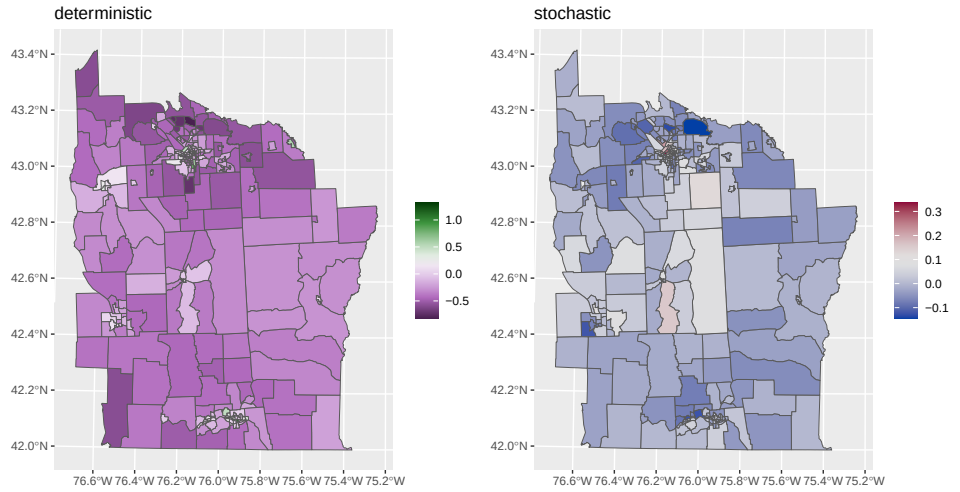
where the first term is the linear model and the second the correction due to spatial autocorrelation of the linear model residuals. Recall that λ gives the strength of this; $\lambda = 0$ implies no correction.

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
rds <- colorRampPalette(brewer.pal(8, "RdBu"))
g1 <- ggplot(data = NY8) +
  geom_sf(aes(fill = sar_trend)) +
  scale_fill_gradientn(colours = hcl.colors(12, "purple-green")) +
  labs(fill = "", title = "deterministic")
#
g2 <- ggplot(data = NY8) +
  geom_sf(aes(fill = sar_stochastic)) +
   scale_fill_gradientn(colours = hcl.colors(12, "blue-red")) +
  labs(fill = "", title = "stochastic")
gridExtra::grid.arrange(g1, g2, nrow = 1)
```



Q26 : *Which of the two components (deterministic "trend" and stochastic residual) gives more information on the predicted leukemia incidence?* *Jump to A26* •

Q27 : *Where is the stochastic residual component most influential in adjusting the trend?* *Jump to A27*
•

9.2 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:

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

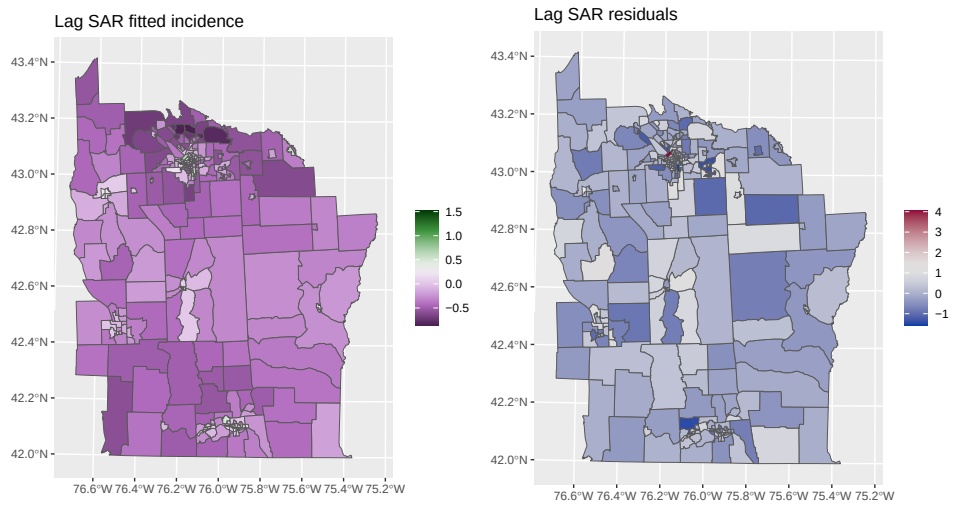
The parameter ρ controls the degree of autocorrelation of the response variable.

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

##
## Call:
## spatialreg::lagsarlm(formula = Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
##     data = NY8, listw = NY8listwB)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -1.587414 -0.387680 -0.029478  0.348490  4.035772
##
## Type: lag
## Coefficients: (asymptotic standard errors)
##              Estimate Std. Error z value  Pr(>|z|)
## (Intercept) -0.513704   0.156073 -3.2914 0.0009968
## PEXPOSURE    0.045623   0.034554  1.3203 0.1867284
## PCTAGE65P    3.636951   0.600085  6.0607 1.355e-09
## PCTOWNHOME  -0.405475   0.170427 -2.3792 0.0173520
##
## Rho: 0.037232, LR test value: 6.6704, p-value: 0.0098027
## Asymptotic standard error: 0.014506
##     z-value: 2.5667, p-value: 0.010266
## Wald statistic: 6.5882, p-value: 0.010266
##
## Log likelihood: -275.3936 for lag model
## ML residual variance (sigma squared): 0.41217, (sigma: 0.64201)
## Number of observations: 281
## Number of parameters estimated: 6
## AIC: 562.79, (AIC for lm: 567.46)
## LM test for residual autocorrelation
## test value: 2.0255, p-value: 0.15468
```

```
NY8$lagsar_fit <- m.z.ppp.lagsar$fit
NY8$lagsar_resid <- m.z.ppp.lagsar$residuals
g1 <- ggplot(data = NY8) +
  geom_sf(aes(fill = lagsar_fit)) +
  scale_fill_gradientn(colours = hcl.colors(12, "purple-green")) +
  labs(fill = "", title = "Lag SAR fitted incidence")
#
g2 <- ggplot(data = NY8) +
  geom_sf(aes(fill = lagsar_resid)) +
  scale_fill_gradientn(colours = hcl.colors(12, "blue-red")) +
  labs(fill = "", title = "Lag SAR residuals")
gridExtra::grid.arrange(g1, g2, nrow = 1)
```



Q28 : *What was the strength of the autocorrelation parameter? Is this model better than one without the spatial lag term?* *Jump to A28* •

Q29 : *Which predictors are now significant? Compare to the SAR error model.* *Jump to A29* •

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

##             Estimate Pr(>|z|)
## (Intercept)  -0.5988   0.0006
## PEXPOSURE     0.0651   0.1177
## PCTAGE65P     3.7569   0.0000
## PCTOWNHOME   -0.4297   0.0239

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

##             Estimate Pr(>|z|)
## (Intercept)  -0.5137   0.0010
## PEXPOSURE     0.0456   0.1867
## PCTAGE65P     3.6370   0.0000
## PCTOWNHOME   -0.4055   0.0174
```

9.3 Spatial Durbin SAR model

The spatial Durbin model is:

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

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 `spatialreg` package also fits this model, if the `type` optional argument is specified as `"mixed"`:

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

##
## Call:
## spatialreg::lagsarlm(formula = Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
##     data = NY8, listw = NY8listwB, type = "mixed")
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -1.765234 -0.360481 -0.014463  0.310558  4.002391
##
## Type: mixed
## Coefficients: (asymptotic standard errors)
##                  Estimate Std. Error z value  Pr(>|z|)
## (Intercept)     -1.094858   0.253452 -4.3198 1.562e-05
## PEXPOSURE        0.198140   0.082388  2.4050  0.016174
## PCTAGE65P        3.351065   0.654743  5.1181 3.086e-07
## PCTOWNHOME       0.138433   0.261916  0.5285  0.597125
## lag.(Intercept)  0.121965   0.055618  2.1929  0.028313
## lag.PEXPOSURE   -0.029996   0.015186 -1.9753  0.048238
## lag.PCTAGE65P    0.176055   0.214533  0.8206  0.411851
## lag.PCTOWNHOME  -0.152598   0.057907 -2.6352  0.008408
##
## Rho: 0.023834, LR test value: 2.0874, p-value: 0.14852
## Asymptotic standard error: 0.01635
##     z-value: 1.4578, p-value: 0.14491
## Wald statistic: 2.1251, p-value: 0.14491
##
## Log likelihood: -269.5561 for mixed model
## ML residual variance (sigma squared): 0.39743, (sigma: 0.63042)
## Number of observations: 281
## Number of parameters estimated: 10
## AIC: 559.11, (AIC for lm: 559.2)
## LM test for residual autocorrelation
## test value: 4.6067, p-value: 0.031847
```

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.

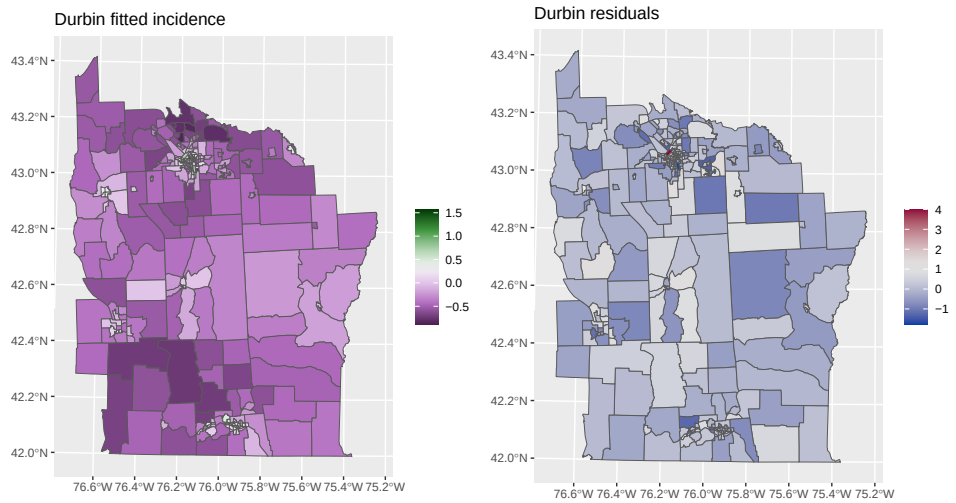
Q30 : *What was the strength of the autocorrelation parameter? Is this model better than one without the spatial lag term?* *Jump to A30* •

Q31 : *Which predictors are now significant? Compare to the SAR error model. Are any of the the neighbour ("lag") effects significant? How do these affect the other coefficients?* *Jump to A31* •

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

##             Estimate Pr(>|z|)
## (Intercept)  -0.5988   0.0006
## PEXPOSURE     0.0651   0.1177
## PCTAGE65P     3.7569   0.0000
## PCTOWNHOME   -0.4297   0.0239
```

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

##                 Estimate Pr(>|z|)
## (Intercept)      -1.0949   0.0000
## PEXPOSURE         0.1981   0.0162
## PCTAGE65P         3.3511   0.0000
## PCTOWNHOME        0.1384   0.5971
## lag.(Intercept)   0.1220   0.0283
## lag.PEXPOSURE    -0.0300   0.0482
## lag.PCTAGE65P     0.1761   0.4119
## lag.PCTOWNHOME   -0.1526   0.0084
```

```
NY8$durbin_fit <- m.z.ppp.durbin$fit
NY8$durbin_resid <- m.z.ppp.durbin$residuals
g1 <- ggplot(data = NY8) +
  geom_sf(aes(fill = durbin_fit)) +
  scale_fill_gradientn(colours = hcl.colors(12, "purple-green")) +
  labs(fill = "", title = "Durbin fitted incidence")
#
g2 <- ggplot(data = NY8) +
  geom_sf(aes(fill = durbin_resid)) +
  scale_fill_gradientn(colours = hcl.colors(12, "blue-red")) +
  labs(fill = "", title = "Durbin residuals")
gridExtra::grid.arrange(g1, g2, nrow = 1)
```



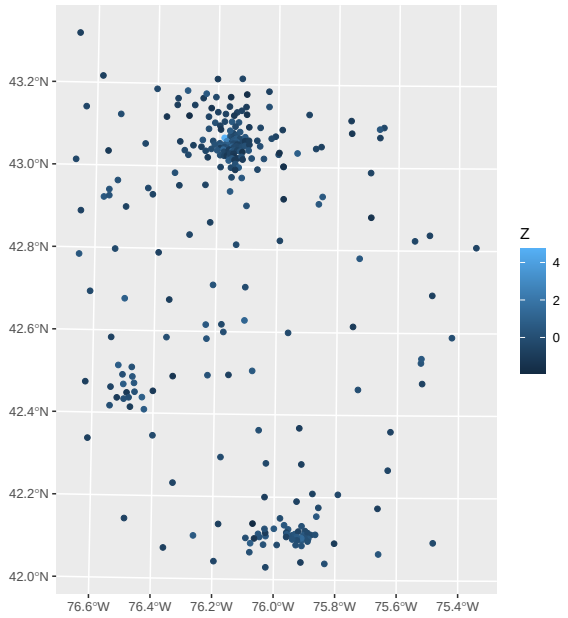
9.4 * Comparison with point-based modelling

Another way to model polygon data is to consider all attributes to be concentrated at the centroids, and use point-based geostatistical models. 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 an `sf` object from the centroids of the polygons and their attributes. •

For this we use the `st_centroid`unction.

```
NY8.pts <- st_centroid(NY8)
str(NY8.pts)

## Classes 'sf' and 'data.frame': 281 obs. of  27 variables:
##  $ AREANAME      : Factor w/ 64 levels "Auburn city",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ AREAKEY       : chr  "36007000100" "36007000200" "36007000300" "36007000400" ...
##  $ X             : num  4.07 4.64 5.71 7.61 7.32 ...
##  $ Y             : num  -67.4 -66.9 -67 -66 -67.3 ...
##  $ POP8          : num  3540 3560 3739 2784 2571 ...
##  $ TRACTCAS      : num  3.08 4.08 1.09 1.07 3.06 1.06 2.09 0.02 2.04 0.02 ...
##  $ PROPCAS       : num  0.00087 0.001146 0.000292 0.000384 0.00119 ...
##  $ PCTOWNHOME    : num  0.328 0.427 0.338 0.462 0.192 ...
##  $ PCTAGE65P     : num  0.147 0.235 0.138 0.119 0.142 ...
##  $ Z             : num  0.142 0.356 -0.582 -0.296 0.457 ...
##  $ AVGIDIST      : num  0.237 0.209 0.171 0.141 0.158 ...
##  $ PEXPOSURE     : num  3.17 3.04 2.84 2.64 2.76 ...
##  $ Cases         : num  3.08 4.08 1.09 1.07 3.06 ...
##  $ Xm            : num  4069 4639 5709 7614 7316 ...
##  $ Ym            : num  -67353 -66862 -66978 -65996 -67318 ...
##  $ Xshift        : num  423391 423961 425031 426935 426638 ...
##  $ Yshift        : num  4661502 4661993 4661878 4662859 4661537 ...
##  $ geometry      :sfc_POINT of length 281; first list element:  'XY' num  422178 4662168
##  $ lmresid       : num  0.1089 0.0346 -0.5591 -0.1194 0.3879 ...
##  $ sarresid      : num  0.1372 0.0729 -0.577 -0.1079 0.4427 ...
##  $ resid.change  : num  0.0284 0.0383 -0.018 0.0115 0.0549 ...
##  $ sar_trend     : num  0.0175 0.299 -0.0405 -0.1783 0.0301 ...
##  $ sar_stochastic: num  -0.0128 -0.0164 0.0359 -0.0102 -0.016 ...
##  $ lagsar_fit    : num  0.0738 0.3199 0.078 -0.1596 0.0833 ...
##  $ lagsar_resid  : num  0.0682 0.0356 -0.6597 -0.1368 0.3736 ...
##  $ durbin_fit    : num  0.0464 0.3758 0.2062 -0.0953 0.1871 ...
##  $ durbin_resid  : num  0.0956 -0.0203 -0.7878 -0.201 0.2698 ...
##  - attr(*, "sf_column")= chr "geometry"
##  - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...
##   ..- attr(*, "names")= chr [1:26] "AREANAME" "AREAKEY" "X" "Y" ...

ggplot(data = NY8.pts) +
  geom_sf(aes(color=Z))
```



Task 42 : Model the 8-county leukemia incidences by an additive model 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, as using the polygons with no spatial structure..

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

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

coefficients(m.z.ppp)

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

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. •

There is no built-in bubble plot function for `sf` point geometries, so we build a small **user-defined function** for this. It will also be used later in the tutorial.

user-defined function

> **Note:** This code uses the ability of R to build a command string using the `paste` function, parse it into R internal format with the `parse` functions, and then evaluate it in the current environment with the `eval` function.

Arguments:

.oint.obj.name : Name of the point object, not the object itself

.field.name : Name of the data column ("field") in the point object to be plotted

.field.label : A text label for the field

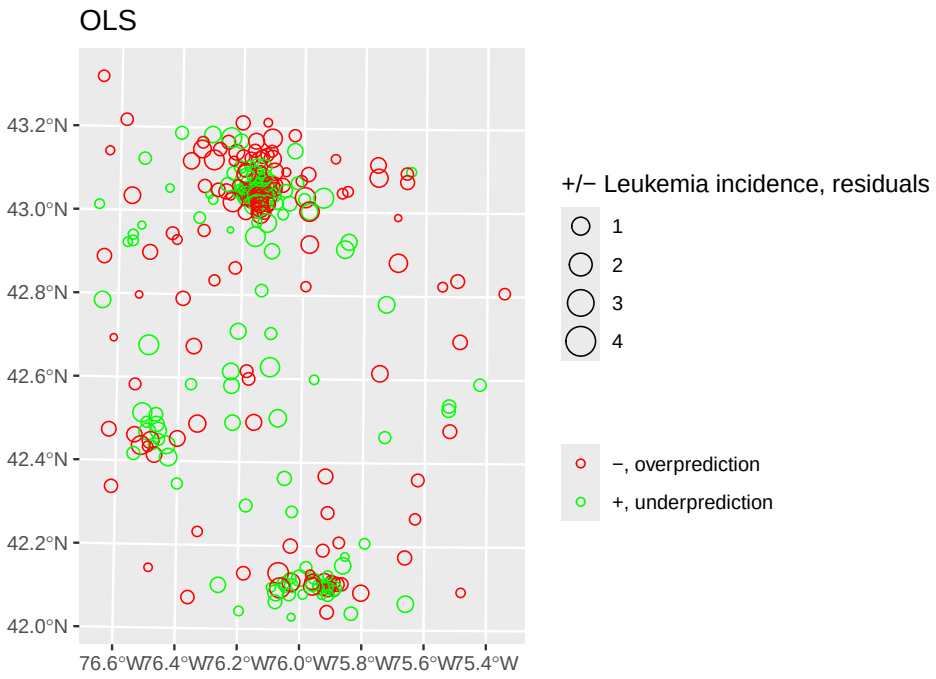
.title : Plot title, default none.

```
bubble.sf <- function(.point.obj.name, .field.name, .field.label, .title="") {
    # make a plus/minus indicator
    eval(parse(
        text = paste0("pm <- factor(",
                      .point.obj.name, "$",
                      .field.name, "> 0)")
    ))
    # rename them
    levels(pm) <- c("-, overprediction", "+, underprediction")
    # plot
    eval(parse(
      text = paste0("ggplot(",
                    .point.obj.name,
                    ") + geom_sf(aes(colour = pm, size = abs(",
                    .field.name,
                    ")), shape = 1) + labs(size = paste('+/-', .field.label),
                    colour = '', title = '",
```

```
                    .title, "') +
                    scale_colour_manual(values = c('red', 'green'))")
    ))
}
```

Now use this function on the OLS residuals:

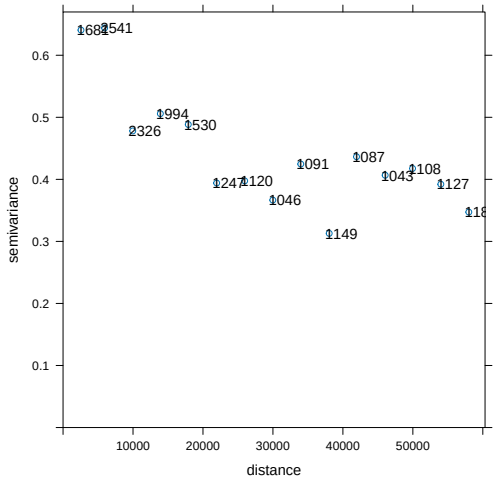
```
NY8.pts$lm.resid <- residuals(m.z.ppp.pts)
bubble.sf("NY8.pts", "lm.resid",
       "Leukemia incidence, residuals", "OLS")
```



Q32 : *Does there appear to be spatial dependence among the residuals?*
Jump to A32 •

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

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



Q33 : *Does the variogram support the assumption of spatial dependence among the residuals?* *Jump to A33* •

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

10 Answers

A1 : *The CRS of the imported objects is +proj=utm +zone=18 +ellps=WGS84 +units=m +no_defs. This refers to the PROJ4 format*[10]*. This one is easy enough to read: UTM projection from the WGS84 ellipsoid, coordinate system UTM in zone 18N.* *Return to Q1* •

A2 : *The number of links is reduced from 1624 "Queen" links to 1528 "Rook" links, i.e., a loss of 96 or 6.28%.*

There is no "correct" answer to the second part of this question, it depends on the process being modelled. *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 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).*

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

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 : *The most common number of links is 6; there are 61 census districts with this number of neighbours.* *Return to Q4* •

A5 : *There are 6 polygons with only one neighbour:*

```
NY8[ix.min.nb, ]
```

```
## Simple feature collection with 6 features and 26 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 360387.1 ymin: 4752012 xmax: 440217.7 ymax: 4808545
## Projected CRS: WGS 84 / UTM zone 18N
##                AREANAME     AREAKEY         X        Y  POP8 TRACTCAS
## 56                   NA 36011990100 -50.24266 36.81422 10494     3.85
## 98    Canastota village 36053030300  18.89833 41.54373  4773     0.44
## 101                  NA 36053030403   8.49700 38.27917  4295     1.39
## 102   Cazenovia village 36053030501  10.82350 24.17720  2599     3.24
## 245   Marcellus village 36067016502 -28.68800 30.85230  1870     1.03
## 246 Skaneateles village 36067016600 -35.98330 26.86590  2786     1.04
##      PROPCAS PCTOWNHOME  PCTAGE65P        Z  AVGIDIST PEXPOSURE
## 56  0.000367  0.5365118 0.03220888 -0.77182 0.0774664  2.047259
## 98  0.000092  0.5683716 0.15482925 -1.19833 0.0206926  0.727191
## 101 0.000324  0.7482353 0.11105937 -0.58616 0.0226724  0.818563
## 102 0.001247  0.5618091 0.14736437  0.48944 0.0332935  1.202777
## 245 0.000551  0.4950860 0.15187166  0.08210 0.0525526  1.659229
## 246 0.000373  0.6508585 0.19059584 -0.31166 0.0918058  2.217090
##       Cases        Xm       Ym   Xshift  Yshift
## 56  3.83889 -50242.66 36814.22 369078.9 4765669
## 98  0.43958  18898.33 41543.73 438219.9 4770399
## 101 1.39555   8497.00 38279.17 427818.6 4767134
## 102 3.23935  10823.50 24177.20 430145.1 4753032
## 245 1.02822 -28688.00 30852.30 390633.6 4759707
## 246 1.04204 -35983.30 26865.90 383338.3 4755721
##                           geometry    lmresid   sarresid resid.change
## 56  POLYGON ((369143 4807833, 3... -0.1813091 -0.1869162 -0.005607100
## 98  POLYGON ((438903.1 4768869,... -1.0099686 -0.9624871  0.047481482
## 101 POLYGON ((430465.2 4767073,... -0.1285998 -0.1309291 -0.002329248
## 102 POLYGON ((429880.3 4754495,...  0.6803931  0.6657846 -0.014608524
## 245 POLYGON ((391274.5 4760874,...  0.1955863  0.2247398  0.029153565
## 246 POLYGON ((382461.1 4755744,... -0.2911735 -0.2897520  0.001421545
##       sar_trend sar_stochastic   lagsar_fit lagsar_resid  durbin_fit
## 56  -0.57494379   -0.009959993 -0.549902361   -0.2219176 -0.53984254
## 98  -0.21396515   -0.021877705 -0.186078362   -1.0122516 -0.35960042
## 101 -0.44973987   -0.005491047 -0.402521807   -0.1836382 -0.48976089
## 102 -0.20820610    0.031861485 -0.132817256    0.6222573 -0.28808108
## 245 -0.13286400   -0.009775819 -0.113706224    0.1958062 -0.24730518
## 246 -0.01797552   -0.003932528 -0.001379646   -0.3102804  0.02397284
##     durbin_resid
## 56   -0.23197746
## 98   -0.83872958
## 101  -0.09639911
## 102   0.77752108
## 245   0.32940518
## 246  -0.33563284
```

There are 2 polygons with 11 neighbours:

```
NY8[ix.max.nb, ]

## Simple feature collection with 2 features and 26 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 411636.9 ymin: 4662952 xmax: 428909.5 ymax: 4737992
## Projected CRS: WGS 84 / UTM zone 18N
##    AREANAME     AREAKEY        X        Y POP8 TRACTCAS  PROPCAS
## 35       NA 36007012800 6.219888 -64.6361 5594     5.13 0.000917
## 83       NA 36023990100 1.861683 -16.6013 5532     3.35 0.000606
##    PCTOWNHOME PCTAGE65P        Z  AVGIDIST PEXPOSURE   Cases       Xm
## 35  0.6905334 0.1973543  0.09150 0.1459107  2.680410 5.13091 6219.888
## 83  0.6787440 0.1231020 -0.24037 0.0763459  2.032689 3.33995 1861.683
##          Ym   Xshift  Yshift                       geometry
## 35 -64636.1 425541.5 4664219 POLYGON ((421936.5 4665583,...
## 83 -16601.3 421183.3 4712254 POLYGON ((411675.8 4736791,...
##       lmresid    sarresid resid.change   sar_trend sar_stochastic
## 35 0.08487727  0.12009268   0.03521540  0.02055104    -0.04914372
## 83 0.07139986 -0.02619639  -0.09759625 -0.29553780     0.08136419
##      lagsar_fit lagsar_resid durbin_fit durbin_resid
## 35 -0.007850536   0.09935054 -0.1567066    0.2482066
## 83 -0.298550829   0.05818083 -0.3077049    0.0673349
```

These have indices 103 and 113, respectively, and in the 8-county dataset they have row names 103 and 113, respectively. Return to Q5 •

A6 : *Field* `AREANAME`*, with 64 names;* `"Syracuse city"` *is the factor name for Syracuse.* Return to Q6 •

A7 : *The weights for the minimum and maximum number of adjacent polgygons, in style W, are 1, 0.0909091* Return to Q7 •

A8 : *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 1, we see the Lakeside neighbourhood (with high incidence) appears to be mostly an industrial area fronting Onondaga Lake.* Return to Q8 •

A9 : *Yes, several high incidences are in the NW (Lakefront neighbourhood and near Westside) of Syracuse City, and some low incidences are clustered near the suburban areas. The ranking map shows this a bit better.* 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 0.* Return to Q10 •

A11 : *The probability increased, i.e., the evidence is less strong to reject the null hypothesis. For leukemia incidence, the probability increased from 0 to 0.001257. However, with both weightings the evidence is strong, and we reach the same conclusion.* Return to Q11
•

A12 : *The highest-leverage area is marked on the graph as original row 110; it has the highest incidence (4.71053) and a moderately-high weighted spatially-lagged proportion.*

This supports the hypothesis of autocorrelation. This area is adjacent to areas with row numbers 114, 120, 121, 131 and 196. Return to Q12 •

A13 : *Yes, clear evidence; there are 2 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 Q13 •

A14 : *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 Q14 •

A15 : *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 Q15 •

A16 : 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 Q16 •

A17 : *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 Q17 •

A18 : *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 Q18 •

A19 : *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 Q19 •

A20 : *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 Q20 •

A21 : *Tract 110 has a very high log-incidence (field Z), because although it has few cases, the population is only 9 people – look at Figure 1: 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 Q21 •

A22 : *Yes, the probability that we would be wrong to reject the null hypothesis of no spatial correlation is only 0.0086.* Return to Q22 •

A23 : *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 lakefront 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 Q23* •

A24 : *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 Q24* •

A25 : *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 Q25* •

A26 : *The trend is much more influential, ranging from about -1 to +1 in normalized incidence (variable* `Z`*).* *Return to Q26* •

A27 : *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 Q27* •

A28 : *The strength of spatial association among predictors is $\rho = 0.037$. 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 Q28* •

A29 : *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 Q29* •

A30 : *The strength of spatial association among predictors is $\rho = 0.024$. 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 Q30* •

A31 : *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 Q31* •

A32 : *No, the red and green circles seem to be intermixed.* *Return to Q32* •

A33 : *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 Q33* •

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, 21, 24, 29, 30, 44, 50, 52, 55, 56

[2] James Durbin. Estimation of parameters in time-series regression-models. *Journal of the Royal Statistical Society Series B*, 22(1):139–153, 1960. 51

[3] P. A. P. Moran. Notes on continuous stochastic phenomena. *Biometrika*, 37(1/2):17–23, 1950. doi: 10.2307/2332142. 29

[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. 40, 41

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

[6] Hadley Wickham, Danielle Navarro, and Thomas Lin Pedersen. Ggplot2: Elegant Graphics for Data Analysis (3e). https://ggplot2-book.org/, 2024. 2

[7] Leland Wilkinson, D. Wills, D. Rope, A. Norton, and R. Dubbs. *The Grammar of Graphics*. Springer, New York, 2nd edition edition, July 2005. ISBN 978-0-387-24544-7. 2

[8] 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 Commands