Introduction to spatial data analysis with R

Edgar Dobriban

June 29, 2014

This script presents an introduction to spatial data analysis with R. It is based on Chapters 2 and 9 of our book: **Applied Spatial Data Analysis with R**, Roger S. Bivand, Edzer Pebesma and V. Gomez-Rubio UseR! Series, Springer, 2nd ed. 2013.

Please consult the textbook for more extensive reference, it is a great resource.

Running this script requires that the code and data bundles from the book be unzipped in the same folder as the script. For instance the zip bundle for chapter 2 is available at: `http://www.asdar-book.org/data2ed.php?chapter=2`

Similarly, get the zip bundle for chapter 9.

In the first part of the lecture, we will work with existing spatial data in R. Second we will also understand how to create such data.

Okay, let's get started: clear up the workspace and change directory to where the data is!

```
#set up: clear all  & set wd
rm(list=ls())
setwd("C:/Dropbox/253/lat_bundle")
```

Let's load two required libraries:

rgdal: processes GDAL - Geospatial Data Analysis files spdep: Spatial dependence: weighting schemes, statistics and models.

```
library(rgdal)

## Loading required package:  sp
## rgdal:  version:  0.8-16, (SVN revision 498)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime:  GDAL 1.11.0, released 2014/04/16
## Path to GDAL shared files:  C:/Users/Edgar/Documents/R/win-library/3.1/rgdal/gdal
## GDAL does not use iconv for recoding strings.
## Loaded PROJ.4 runtime:  Rel.  4.8.0, 6 March 2012, [PJ_VERSION: 480]
## Path to PROJ.4 shared files:  C:/Users/Edgar/Documents/R/win-library/3.1/rgdal/proj

library(spdep)

## Loading required package:  Matrix
```

The data set consists of Leukemia incidence in 281 census tracts in 8 central NY state counties (and was collected in the 1980's)

This is how we read the data sets:

```
NY8 <- readOGR(".", "NY8_utm18")

## OGR data source with driver: ESRI Shapefile
## Source: ".", layer: "NY8_utm18"
## with 281 features and 17 fields
## Feature type: wkbPolygon with 2 dimensions
```

```
# read Shapefile (spdep package),
# arguments (1)"." - source: a directory; here current wd
# (2) layer: shapefile name
# Note: find shapefiles by googling "[location] shapefile"
```

Some additional data:

```
#city names
cities <- readOGR(".", "NY8cities")

## OGR data source with driver: ESRI Shapefile
## Source: ".", layer: "NY8cities"
## with 6 features and 1 fields
## Feature type: wkbPoint with 2 dimensions

#locations of 11 inactive hazardous waste sites;
# TCE: Trichloroethylene
TCE <- readOGR(".", "TCE")

## OGR data source with driver: ESRI Shapefile
## Source: ".", layer: "TCE"
## with 11 features and 5 fields
## Feature type: wkbPoint with 2 dimensions
```

Let's look at how the data is stored.

```
# How is the data stored?
class(NY8)

## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"

getClass("SpatialPolygonsDataFrame")

## Class "SpatialPolygonsDataFrame" [package "sp"]
##
## Slots:
##
## Name:         data    polygons   plotOrder
## Class:  data.frame        list     integer
##
## Name:         bbox proj4string
## Class:      matrix         CRS
##
## Extends:
## Class "SpatialPolygons", directly
## Class "Spatial", by class "SpatialPolygons", distance 2

#it's a SpatialPolygonsDataFrame - a data.frame on polygons

#spatial polygons contain polygons and Spatial* characteristics
getClass("SpatialPolygons")

## Class "SpatialPolygons" [package "sp"]
##
## Slots:
##
```

```
## Name:     polygons   plotOrder        bbox
## Class:        list     integer      matrix
##
## Name:  proj4string
## Class:         CRS
##
## Extends: "Spatial"
##
## Known Subclasses: "SpatialPolygonsDataFrame"
```

```
# a polygon is a sequence of closed lines;
# point coordinates where the first point equals the last
getClass("Polygon")
```

```
## Class "Polygon" [package "sp"]
##
## Slots:
##
## Name:    labpt    area    hole ringDir  coords
## Class: numeric numeric logical integer  matrix
##
## Extends: "Line"
```

```
#labpt - label point, centroid of polygon
# a line is an ordered list of coordinates
getClass("Line")
```

```
## Class "Line" [package "sp"]
##
## Slots:
##
## Name:  coords
## Class: matrix
##
## Known Subclasses: "Polygon"
```

```
#finally... Spatial is the mother class of all Spatial* classes
# used in in the sp package
getClass("Spatial")
```

```
## Class "Spatial" [package "sp"]
##
## Slots:
##
## Name:         bbox proj4string
## Class:      matrix         CRS
##
## Known Subclasses:
## Class "SpatialPoints", directly
## Class "SpatialGrid", directly
## Class "SpatialLines", directly
## Class "SpatialPolygons", directly
## Class "SpatialPointsDataFrame", by class "SpatialPoints", distance 2
## Class "SpatialPixels", by class "SpatialPoints", distance 2
## Class "SpatialGridDataFrame", by class "SpatialGrid", distance 2
## Class "SpatialLinesDataFrame", by class "SpatialLines", distance 2
## Class "SpatialPixelsDataFrame", by class "SpatialPoints", distance 3
## Class "SpatialPolygonsDataFrame", by class "SpatialPolygons", distance 2
```

What does the data contain?

```
summary(NY8)

## Object of class SpatialPolygonsDataFrame
## Coordinates:
##       min     max
## x  358242  480393
## y 4649755 4808545
## Is projected: TRUE
## proj4string :
## [+proj=utm +zone=18 +ellps=WGS84 +units=m
## +no_defs]
## Data attributes:
##                    AREANAME          AREAKEY
##  NA                    : 83   36007000100:  1
##  Syracuse city         : 63   36007000200:  1
##  Binghamton city       : 18   36007000300:  1
##  Remainder of Clay tow:  6   36007000400:  1
##  Johnson City village :  5   36007000500:  1
##  Onondaga town         :  5   36007000600:  1
##  (Other)               :101   (Other)    :275
##        X                 Y
##  Min.   :-55.48   Min.   :-75.29
##  1st Qu.:-19.46   1st Qu.:-30.60
##  Median :-12.47   Median : 31.97
##  Mean   :-11.31   Mean   :  4.98
##  3rd Qu.: -1.21   3rd Qu.: 39.12
##  Max.   : 53.51   Max.   : 56.41
##
##       POP8          TRACTCAS
##  Min.   :    9   Min.   :0.00
##  1st Qu.: 2510   1st Qu.:0.31
##  Median : 3433   Median :1.89
##  Mean   : 3764   Mean   :2.11
##  3rd Qu.: 4889   3rd Qu.:3.08
##  Max.   :13015   Max.   :9.29
##
##     PROPCAS           PCTOWNHOME
##  Min.   :0.000000   Min.   :0.0008
##  1st Qu.:0.000093   1st Qu.:0.4589
##  Median :0.000413   Median :0.6509
##  Mean   :0.000595   Mean   :0.5873
##  3rd Qu.:0.000917   3rd Qu.:0.7561
##  Max.   :0.006993   Max.   :1.0000
##
##    PCTAGE65P            Z
##  Min.   :0.0040   Min.   :-1.921
##  1st Qu.:0.0999   1st Qu.:-0.717
##  Median :0.1264   Median :-0.288
##  Mean   :0.1373   Mean   :-0.216
##  3rd Qu.:0.1610   3rd Qu.: 0.250
##  Max.   :0.5051   Max.   : 4.711
##
```

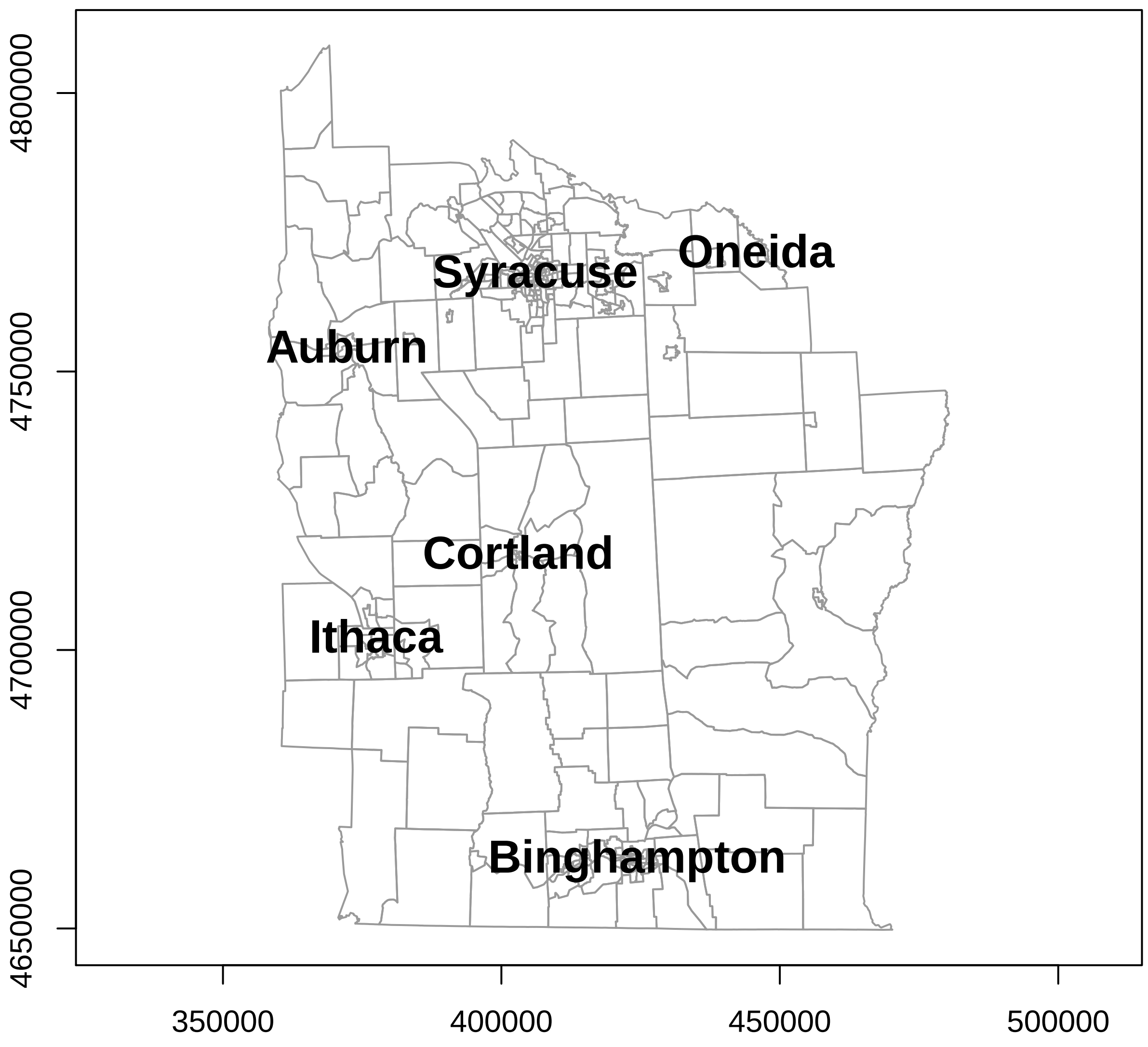
```
##     AVGIDIST       PEXPOSURE         Cases
##  Min.   :0.018   Min.   :0.613   Min.   :0.000
##  1st Qu.:0.027   1st Qu.:0.994   1st Qu.:0.309
##  Median :0.032   Median :1.175   Median :1.889
##  Mean   :0.149   Mean   :1.804   Mean   :2.107
##  3rd Qu.:0.130   3rd Qu.:2.566   3rd Qu.:3.083
##  Max.   :3.526   Max.   :5.865   Max.   :9.286
##
##        Xm               Ym
##  Min.   :-55482   Min.   :-75291
##  1st Qu.:-19460   1st Qu.:-30601
##  Median :-12469   Median : 31970
##  Mean   :-11309   Mean   :  4980
##  3rd Qu.: -1213   3rd Qu.: 39123
##  Max.   : 53509   Max.   : 56410
##
##      Xshift           Yshift
##  Min.   :363839   Min.   :4653564
##  1st Qu.:399862   1st Qu.:4698254
##  Median :406852   Median :4760825
##  Mean   :408013   Mean   :4733835
##  3rd Qu.:418108   3rd Qu.:4767978
##  Max.   :472830   Max.   :4785265
##

#Note the special slots of this class
#Coordinates
#proj4string
#Spatial data.frame - with coordinates X,Y
```

What about the 'cities' data?

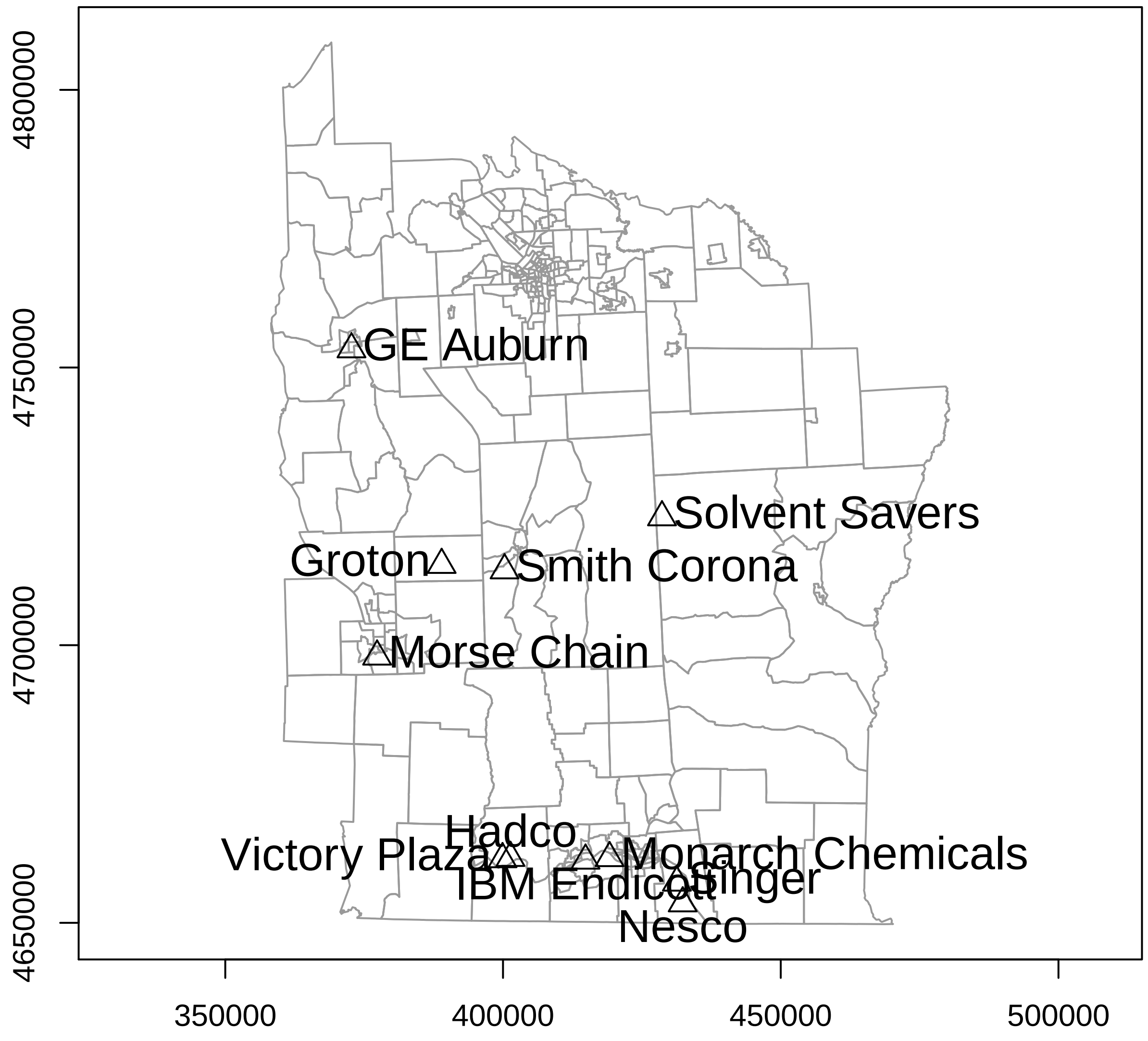
```
summary(cities)

## Object of class SpatialPointsDataFrame
## Coordinates:
##                min     max
## coords.x1  372237  445728
## coords.x2 4662141 4771698
## Is projected: TRUE
## proj4string :
## [+proj=utm +zone=18 +ellps=WGS84 +units=m
## +no_defs]
## Number of points: 6
## Data attributes:
##      Auburn Binghampton    Cortland      Ithaca
##           1           1           1           1
##      Oneida    Syracuse
##           1           1
```

Let's make some plots:

```
#plot cities & TCE locations
plot(NY8, border="grey60", axes=TRUE)
text(coordinates(cities), labels=as.character(cities$names), font=2, cex=1.5)
```

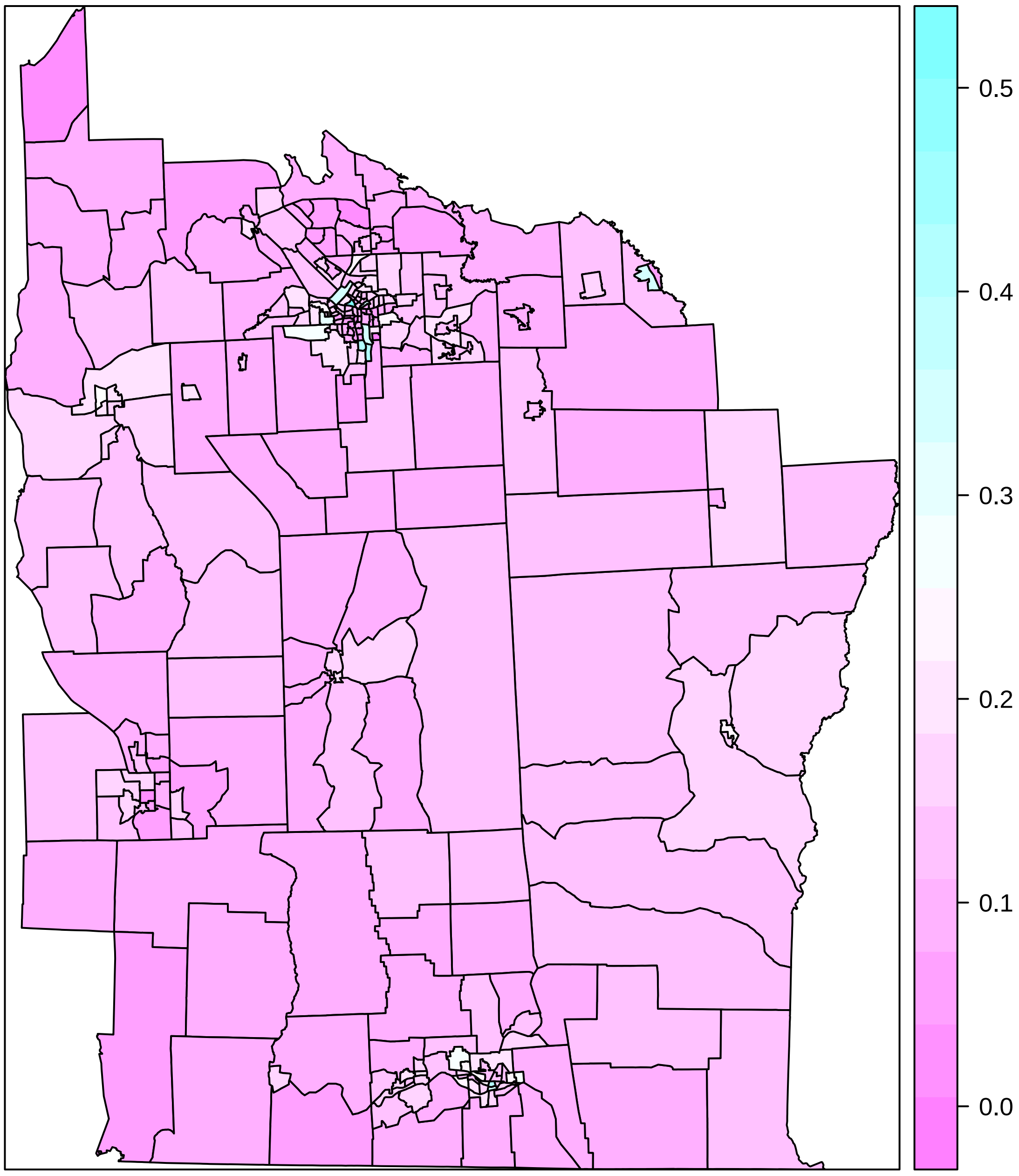


```
plot(NY8, border="grey60", axes=TRUE)
points(TCE, pch=2, cex=1.5)
text(coordinates(TCE), labels=as.character(TCE$name), cex=1.5,
     font=1, pos=c(4,1,4,1,4,4,4,2,3,4,2), offset=0.3)
```

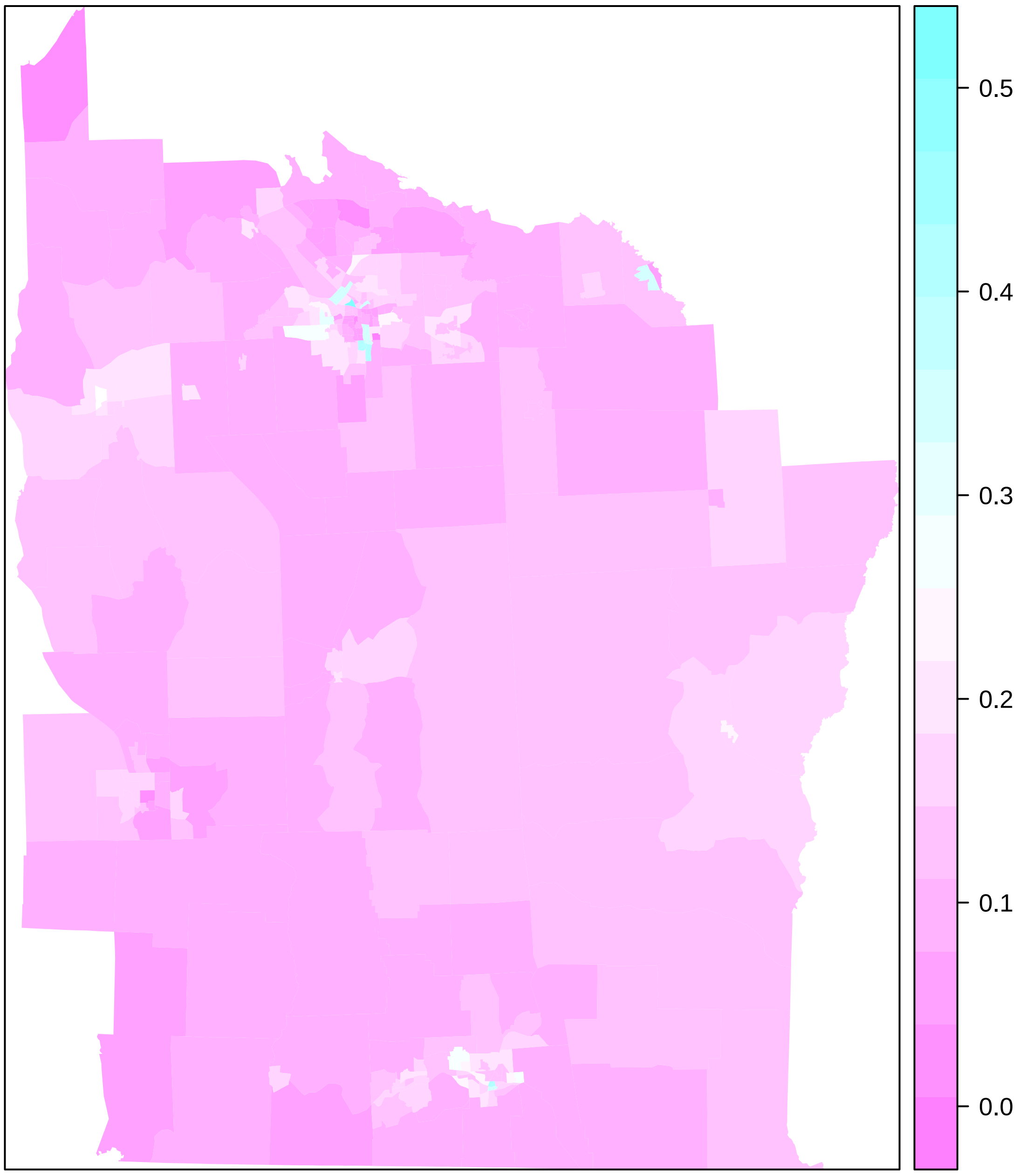


Let's plot one of the features - percent age > 65.

```
#plot one of the features - percent age > 65
spplot(NY8, c("PCTAGE65P"))#, col="transparent"
```



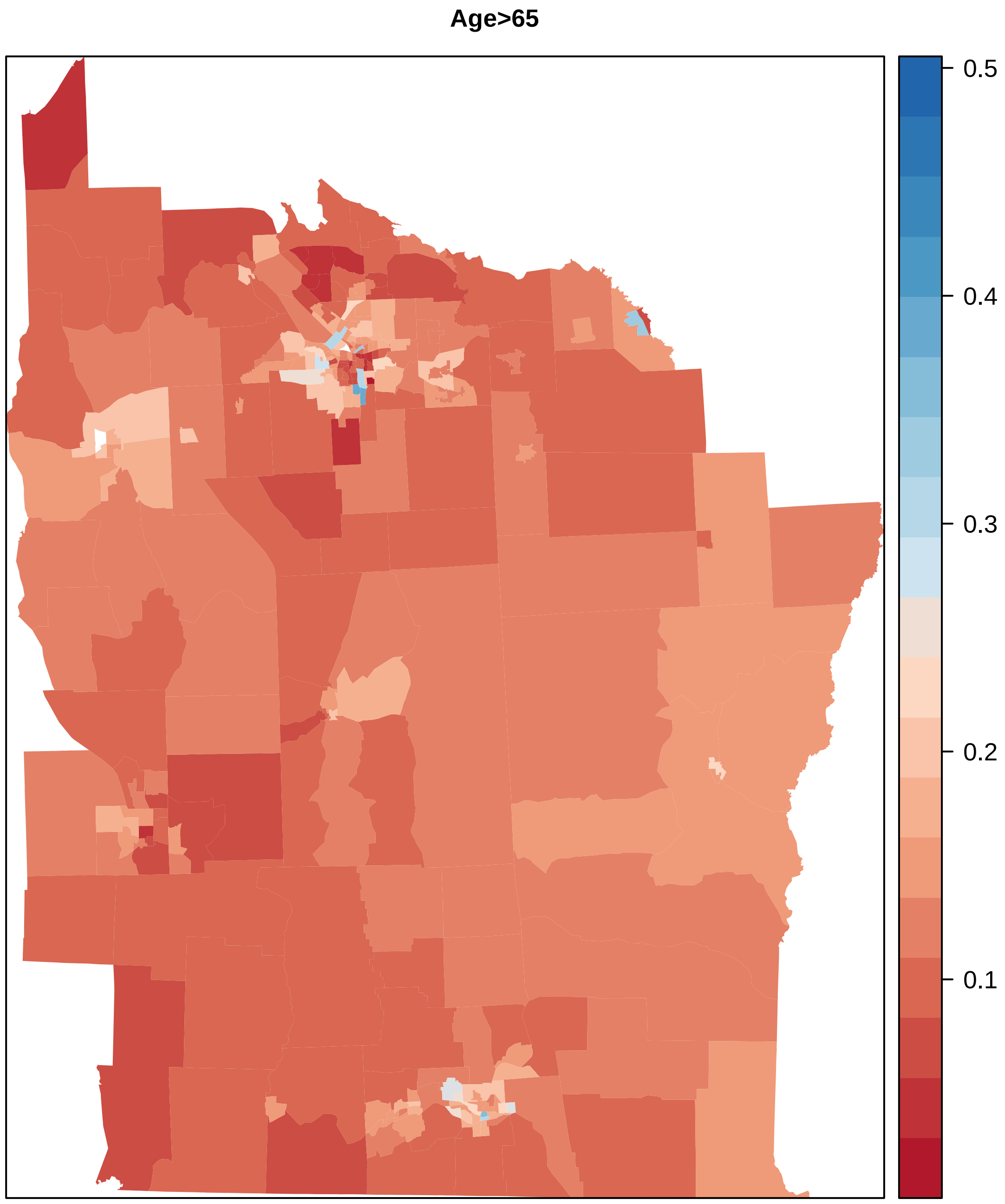
```r
spplot(NY8, c("PCTAGE65P"), col="transparent")
```



Let's make a different plot, with a new color palette:

```
#different plot: new color palette
#load package
library("RColorBrewer")

#color palette creator function
rds <- colorRampPalette(brewer.pal(8, "RdBu"))
#get a range for the values
tr_at <- seq(min(NY8$PCTAGE65P), max(NY8$PCTAGE65P), length.out=20)
#create a color interpolating function taking the required
#number of shades as argument
tr_rds <- rds(20)
#parameters
# at - at which values colors change
```

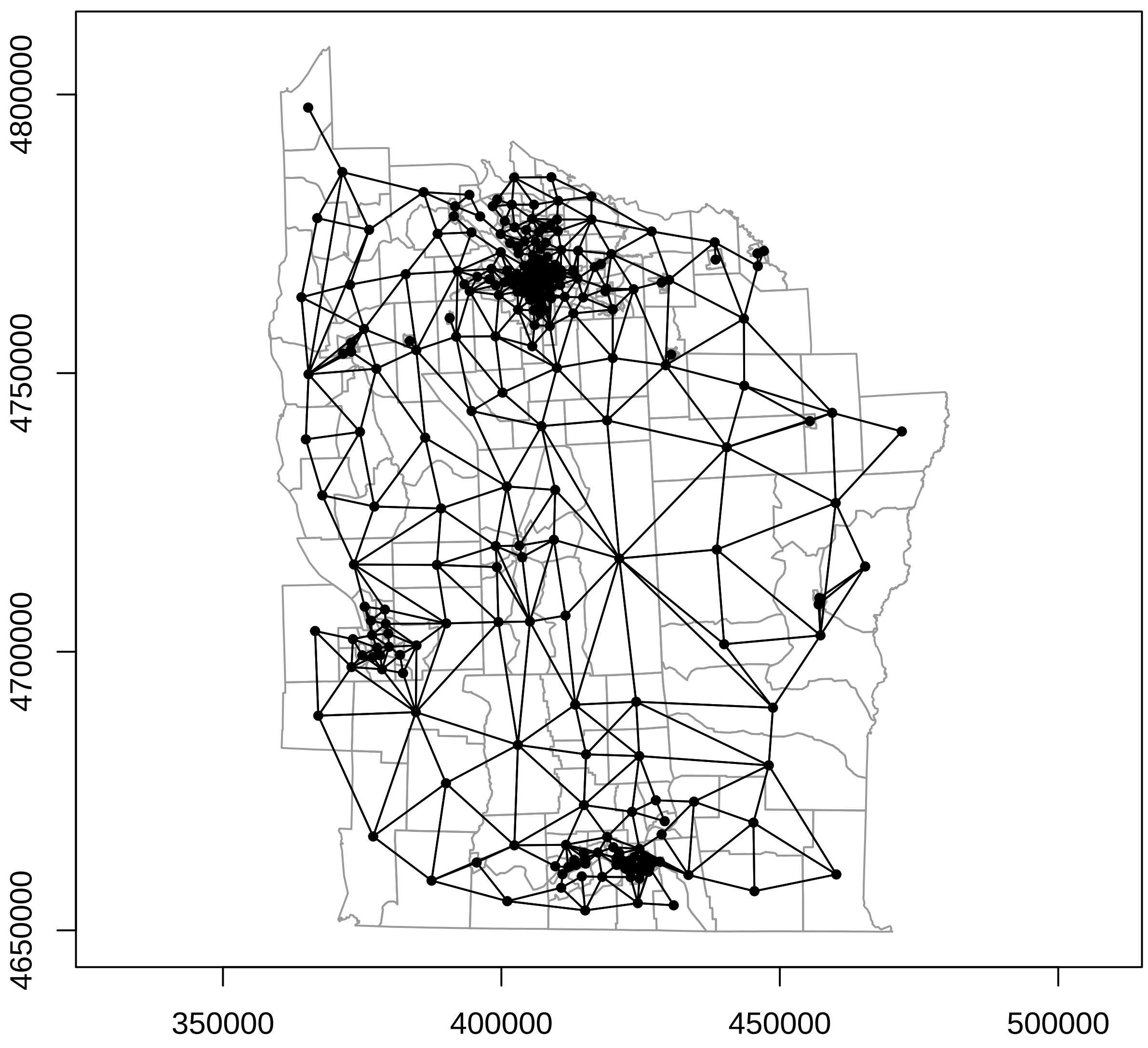
```r
# col.regions - specify fill colors
tr_pl <- spplot(NY8, c("PCTAGE65P"), at=tr_at, col="transparent",
                col.regions=tr_rds, main=list(label="Age>65", cex=0.8))
plot(tr_pl)
```



Finally, let's read the last piece of data - a list of neighbors that specifies the lattice structure.

```r
# reads a GAL lattice file into a neighbors list
NY_nb <- read.gal("NY_nb.gal", region.id=row.names(NY8))

summary(NY_nb) #which states are neighbors?

## Neighbour list object:
## Number of regions: 281
## Number of nonzero links: 1522
## Percentage nonzero weights: 1.928
## Average number of links: 5.416
## Link number distribution:
##
##  1  2  3  4  5  6  7  8  9 10 11
##  6 11 28 45 59 49 45 23 10  3  2
## 6 least connected regions:
## 55 97 100 101 244 245 with 1 link
## 2 most connected regions:
## 34 82 with 11 links
```

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



We finally get to the data analysis part!
Let's first fit a simple linear model, where we regress the leukemia incidence on covariates of interest.
The leukemia incidence is transformed the following way:

$$z = log(1000(Y_i + 1)/n_i)$$

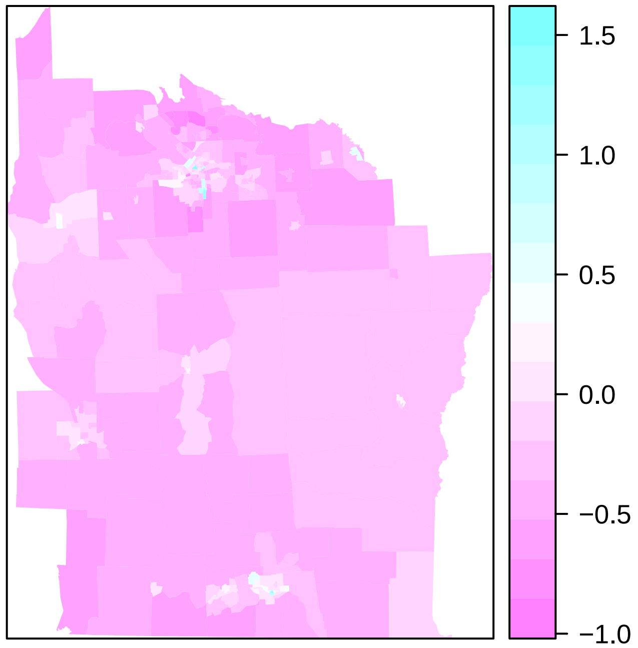
The covariates are:

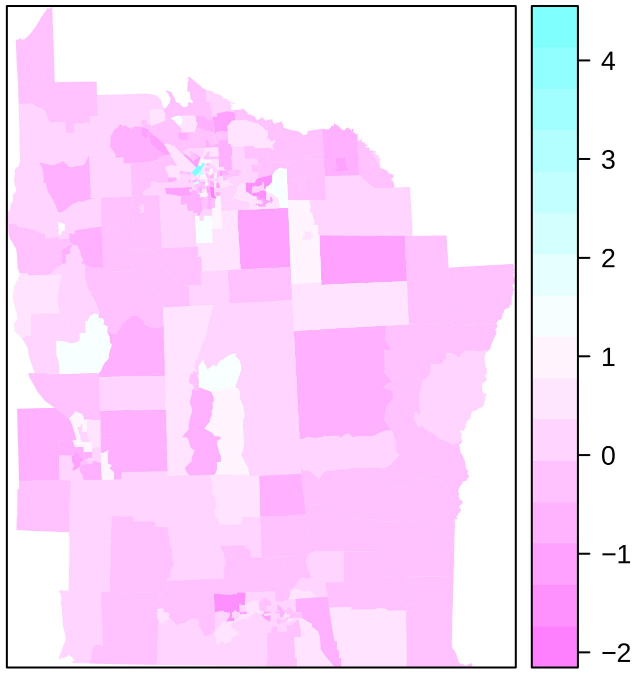
- exposure to TCE- log inverse distance from nearest site
- percent aged >65
- percent owning home

```
nylm <- lm(Z~PEXPOSURE+PCTAGE65P+PCTOWNHOME, data=NY8)
summary(nylm)

##
## Call:
## lm(formula = Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data = NY8)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.742 -0.396 -0.033  0.335  4.140
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.5173     0.1586   -3.26   0.0012
## PEXPOSURE     0.0488     0.0351    1.39   0.1648
## PCTAGE65P     3.9509     0.6055    6.53  3.2e-10
## PCTOWNHOME   -0.5600     0.1703   -3.29   0.0011
##
## (Intercept) **
## PEXPOSURE
## PCTAGE65P   ***
## PCTOWNHOME  **
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.657 on 277 degrees of freedom
## Multiple R-squared:  0.193,Adjusted R-squared:  0.184
## F-statistic: 22.1 on 3 and 277 DF,  p-value: 7.31e-13
```

Let's plot the fit and residual.

```
NY8$lm_fit <- nylm$fit
NY8$lm_residual <- nylm$residuals
rds <- colorRampPalette(brewer.pal(8, "RdBu"))
fit_pl <- spplot(NY8, c("lm_fit"), col="transparent", cex=0.8)
res_pl <- spplot(NY8, c("lm_residual"), col="transparent", cex=0.8)
plot(fit_pl, split=c(1,1,2,1), more=TRUE)
plot(res_pl, split=c(2,1,2,1), more=FALSE)
```





Let's move to a more sophisticated autoregressive model:

$$(I - \lambda W)(Y - X\beta) = \varepsilon$$

In order to specify such a model, we need to create the adjacency matrix W. We generate this as a 'spatial weight object' from the neighbor list object we loaded.

```
# generate weight object from neighbor list object
# "B" - generates binary weights
NYlistw<-nb2listw(NY_nb, style = "B")

# fit model (I - lambda* W)(Y- X* beta) = epsilon
nysar<-spautolm(Z~PEXPOSURE+PCTAGE65P+PCTOWNHOME, data=NY8, listw=NYlistw)

summary(nysar)
```

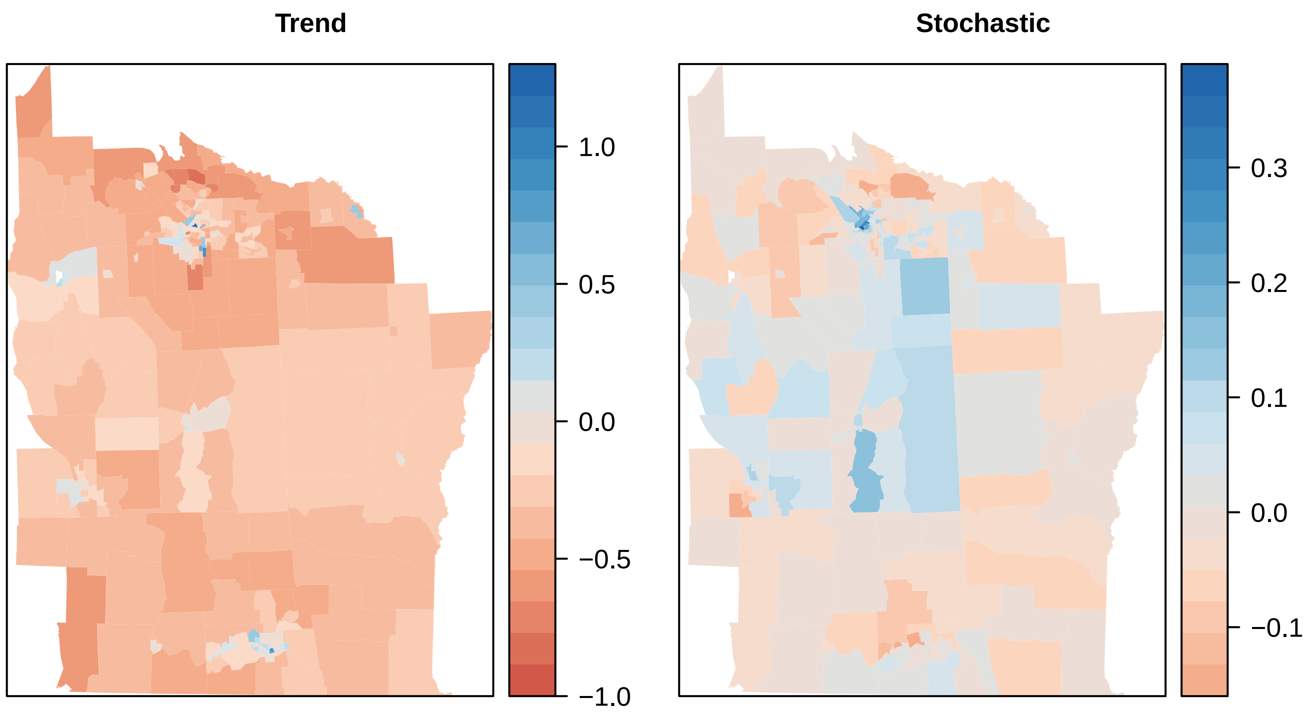
```
##
## Call:
## spautolm(formula = Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data = NY8,
##     listw = NYlistw)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.56754 -0.38239 -0.02643  0.33109  4.01219
##
## Coefficients:
##              Estimate Std. Error z value
## (Intercept) -0.618193   0.176784 -3.4969
## PEXPOSURE    0.071014   0.042051  1.6888
## PCTAGE65P    3.754200   0.624722  6.0094
## PCTOWNHOME  -0.419890   0.191329 -2.1946
##              Pr(>|z|)
## (Intercept) 0.0004707
## PEXPOSURE   0.0912635
## PCTAGE65P   1.862e-09
## PCTOWNHOME  0.0281930
##
## Lambda: 0.04049 LR test value: 5.244 p-value: 0.022026
## Numerical Hessian standard error of lambda: 0.01718
##
## Log likelihood: -276.1
## ML residual variance (sigma squared): 0.4139, (sigma: 0.6433)
## Number of observations: 281
## Number of parameters estimated: 6
## AIC: 564.2
```

Note: lambda significantly different from 0 indicates that there is significant reduction in RSS by modelling the spatial correlations.

For more understanding, let's plot the trend and stochastic component, which are the first and second element on the RHS of the following equation:

$$Y = X * \beta + \lambda W)(Y - X\beta) + \varepsilon$$

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



Ok this is great, but how do I transform my data into spatial format? Let's examine how to create spatial data. There are two important structures, `"SpatialPointsDataFrame"` and `listw`.

```
#Key data type #1: "SpatialPointsDataFrame"
getClass("SpatialPointsDataFrame")

## Class "SpatialPointsDataFrame" [package "sp"]
##
## Slots:
##
## Name:         data  coords.nrs      coords
## Class:  data.frame     numeric      matrix
##
## Name:         bbox proj4string
## Class:      matrix         CRS
```

```
##
## Extends:
## Class "SpatialPoints", directly
## Class "Spatial", by class "SpatialPoints", distance 2
##
## Known Subclasses:
## Class "SpatialPixelsDataFrame", directly, with explicit coerce
```

```
#read data matrix
CRAN_df <- read.table("CRAN051001a.txt", header=TRUE)
CRAN_mat <- cbind(CRAN_df$long, CRAN_df$lat)
row.names(CRAN_mat) <- 1:nrow(CRAN_mat)
str(CRAN_mat)
```

```
##  num [1:54, 1:2] 153 145 16.3 -49.3 -42.9 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:54] "1" "2" "3" "4" ...
##   ..$ : NULL
```

```
#set CRS
llCRS <- CRS("+proj=longlat +ellps=WGS84")
CRAN_sp <- SpatialPoints(CRAN_mat, proj4string=llCRS)
summary(CRAN_sp)
```

```
## Object of class SpatialPoints
## Coordinates:
##                min    max
## coords.x1 -122.95 153.03
## coords.x2  -37.82  57.05
## Is projected: FALSE
## proj4string : [+proj=longlat +ellps=WGS84]
## Number of points: 54
```

```
#if you don't need CRS
llCRS <- CRS(as.character(NA))

CRAN_spdf <- SpatialPointsDataFrame(CRAN_sp, CRAN_df)
summary(CRAN_spdf)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##                min    max
## coords.x1 -122.95 153.03
## coords.x2  -37.82  57.05
## Is projected: FALSE
## proj4string : [+proj=longlat +ellps=WGS84]
## Number of points: 54
## Data attributes:
##              place        north          east
##  Bern           : 2   40d26'N: 2   13d22'E : 2
##  Pittsburgh, PA: 2   46d57'N: 2   7d26'E  : 2
##  Aalborg        : 1   20d45'S: 1   80d0'W  : 2
##  Aizu           : 1   22d43'S: 1   0d10'W  : 1
##  Ames, IA       : 1   22d54'S: 1   118d15'W: 1
##  Arezzo         : 1   23d32'S: 1   118d47'E: 1
##  (Other)        :46   (Other):46   (Other) :45
##           loc          long
##  Brazil     : 5   Min.   :-122.95
##  Germany    : 5   1st Qu.: -47.38
##  Italy      : 4   Median :   7.85
##  France     : 3   Mean   :  -0.66
##  Switzerland: 3   3rd Qu.:  16.83
##  Australia  : 2   Max.   : 153.03
##  (Other)    :32
##       lat
##  Min.   :-37.8
##  1st Qu.: 34.5
##  Median : 42.7
##  Mean   : 31.7
##  3rd Qu.: 47.6
##  Max.   : 57.0
##
```

The second key data type is the class of Spatial Weights - listw.

It is based on the Spatial neighbors `("nb")` class. This class is a list of length n, where n is the number of our nodes, with the index numbers of neighbors of each component stored as an integer vector. Run the following: `vignette("nb", package = "spdep")` to read more on neighbors.

Recall that we already have `NY_nb` as an object of nb class.

On top of it we can add nonzero weights for each pair of neighbors, getting the listw class.

```
NY_w <-nb2listw(NY_nb)
summary(NY_w)

## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 281
## Number of nonzero links: 1522
## Percentage nonzero weights: 1.928
## Average number of links: 5.416
## Link number distribution:
##
##  1  2  3  4  5  6  7  8  9 10 11
##  6 11 28 45 59 49 45 23 10  3  2
## 6 least connected regions:
## 55 97 100 101 244 245 with 1 link
## 2 most connected regions:
## 34 82 with 11 links
##
## Weights style: W
## Weights constants summary:
##     n    nn  S0    S1   S2
## W 281 78961 281 115.4 1169
```

By default each node is normalized to have total sum of out-weights equal to 1. We can also choose to use binary weights.

```
NY_w <-nb2listw(NY_nb, style = "B")
summary(NY_w)

## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 281
## Number of nonzero links: 1522
## Percentage nonzero weights: 1.928
## Average number of links: 5.416
## Link number distribution:
##
##  1  2  3  4  5  6  7  8  9 10 11
##  6 11 28 45 59 49 45 23 10  3  2
## 6 least connected regions:
## 55 97 100 101 244 245 with 1 link
## 2 most connected regions:
## 34 82 with 11 links
##
## Weights style: B
## Weights constants summary:
##     n    nn   S0   S1    S2
## B 281 78961 1522 3044 37160
```