Introduction to the North Carolina SIDS data set

Roger Bivand

December 15, 2004

1 Introduction

This data set was presented first in Symons et al. (1983), analysed with reference to the spatial nature of the data in Cressie and Read (1985), expanded in Cressie and Chan (1989), and used in detail in Cressie (1991). It is for the 100 counties of North Carolina, and includes counts of numbers of live births (also non-white live births) and numbers of sudden infant deaths, for the 1974–1978 and 1979–1984 periods. In Cressie and Read (1985), a listing of county neighbours based on shared boundaries (contiguity) is given, and in Cressie and Chan (1989), and in Cressie (1991, pp. 386–389), a different listing based on the criterion of distance between county seats, with a cutoff at 30 miles. The county seat location coordinates are given in miles in a local (unknown) coordinate reference system. The data are also used to exemplify a range of functions in the S-PLUS spatial statistics module user's manual (Kaluzny et al., 1996).

2 Getting the data into R

We will be using the **spdep** package, here version: spdep, version 0.3-01, 2004-11-20, and the **maptools** package. The data from the sources refered to above is collected in the `nc.sids` data set in **spdep**. But to map it, we also need access to data for the county boundaries for North Carolina; this has been made available in the **maptools** package in shapefile format[1]. These data are known to be geographical coordinates (longitude-latitude in decimal degrees) and are assumed to use the NAD83 datum.

```
> library(spdep)
```

The shapefile format presupposes that you have three files with extensions `*.shp`, `*.shx`, and `*.dbf`, where the first contains the geometry data, the second the spatial index, and the third the attribute data. They are required to have the same name apart from the extension, and are read using `read.shape()`. By default, this function reads in the data in all three files, although it is only given the name of the file with the geometry. The imported object in R has class `Map`, and is a list with two components, `"Shapes"`, which is a list of shapes, and `"att.data"`, which is a data frame with tabular data, one row for each shape in `"Shapes"`. Here we will be using data previously read from the shapefile, and stored in the `nc.sids` data set:

```
> data(nc.sids)
```

```
> plot(sidspolys, forcefill = FALSE)
> points(sidscents)
```

[1] These data are taken with permission from: `http://sal.agecon.uiuc.edu/datasets/sids.zip`.

We can examine the names of the columns of the data frame to see what it contains — in fact some of the same columns that we will be examining below, and some others which will be useful in cleaning the data set. We will similarly convert the geometry format of the `Map` object to that of a `polylist` object, which will be easier to handle. Finally, we retreive the centroids of the county polygons to use as label points. Using the `plot()` function for `"polylist"` objects from **maptools**, we can display the polygon boundaries and centroids, shown in Figure 1.

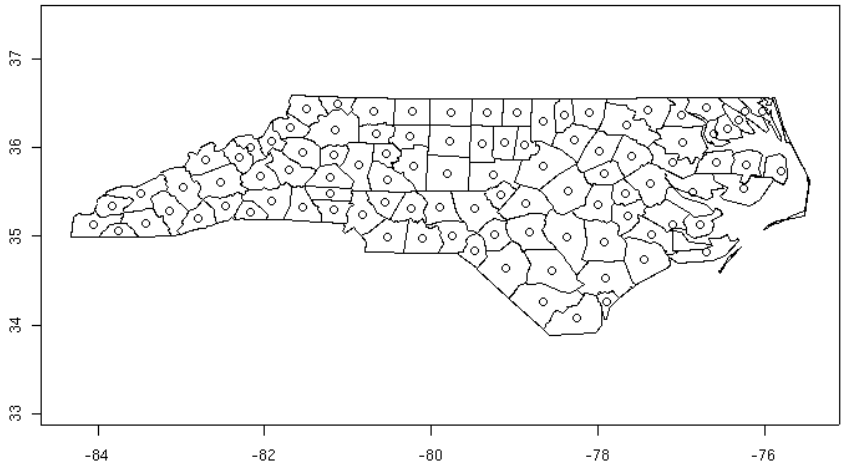

Figure 1: County boundaries and polygon centroids, North Carolina

It may be of interest to look at the structure of a polygon list member. This is made up of a two-column matrix with polygon coordinates. In general, each sub-polygon will have equal first and last coordinates to ensure closure, but this is not absolutely required. Rows in the coordinate matrix set to `NA` represent breaks between sub-polygons, and are respected by the underlying R graphics functions. The attributes contain further information about the polygon: `pstart` is a list with `from` and `to` components, which are vectors of first and last rows in the matrix for each sub-polygon in the object — there are `nParts` elements in both `from` and `to`. `RingDir` and `ringDir` should be the same (but are not here, `ringDir` is correct, and `RingDir` is wrong!), and are computed in two different ways to determine whether each of the `nParts` sub-polygons runs clockwise or counter-clockwise. Counter-clockwise sub-polygons are "holes" in the surrounding sub-polygon. Finally, `bbox` contains the bounding box of this object. Its appearance is shown in Figure 2.

```
> round(t(sidspolys[[56]]), 3)
```

```
       [,1]    [,2]    [,3]    [,4]    [,5]    [,6] [,7]    [,8]
[1,] -75.783 -75.773 -75.545 -75.703 -75.741 -75.783   NA -75.891
[2,]  36.225  36.229  35.788  36.050  36.050  36.225   NA  35.631
       [,9]   [,10]   [,11]   [,12]   [,13]   [,14]   [,15]   [,16]
[1,] -75.908 -76.021 -75.988 -75.818 -75.749 -75.729 -75.779 -75.891
[2,]  35.666  35.669  35.893  35.924  35.869  35.665  35.579  35.631
     [,17]   [,18]   [,19]   [,20]   [,21]   [,22]   [,23]   [,24]
[1,]    NA -75.491 -75.475 -75.521 -75.692 -75.749 -75.526 -75.457
[2,]    NA  35.670  35.564  35.281  35.235  35.190  35.228  35.617
       [,25]   [,26]
```

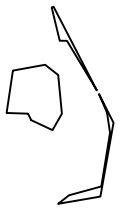

Figure 2: Plot of polygon 56 from the list of polygons.

```
[1,] -75.534 -75.491
[2,]  35.769  35.670
attr(,"after")
[1] NA NA NA
attr(,"plotOrder")
[1] 1 2 3
attr(,"pstart")
attr(,"pstart")$from
[1]  1  8 18

attr(,"pstart")$to
[1]  6 16 26

attr(,"bbox")
[1] -76.02121  35.18983 -75.45698  36.22926
attr(,"RingDir")
[1] 1 1 1
attr(,"nParts")
[1] 3
attr(,"ringDir")
[1] 1 1 1
```

```
> sidscents[56, ]
```

```
[1] -75.80982  35.73548
```

```
> plot(sidspolys[[56]], type = "l", asp = 1, axes = FALSE,
+     xlab = "", ylab = "")
```

We will now examine the data set reproduced from Cressie and collaborators, included in **spdep**, and add the neighbour relationships used in Cressie and Chan (1989) to the background map as a graph shown in Figure 3:

```
> plot(sidspolys, border = "grey", forcefill = FALSE)
> plot(ncCC89.nb, sidscents, add = TRUE, col = "blue")
```

Printing the neighbour object shows that it is a neighbour list object, with a very sparse structure — if displayed as a matrix, only 3.94% of cells would be filled. Objects of class `nb` contain a list as long as the number of counties; each component of the list is a vector with the index numbers of the neighbours of the county in question, so that the neighbours of the county with `region.id` of "1825" can be retreived by matching against the indices. More information can be obtained by using `summary()` on an `nb`

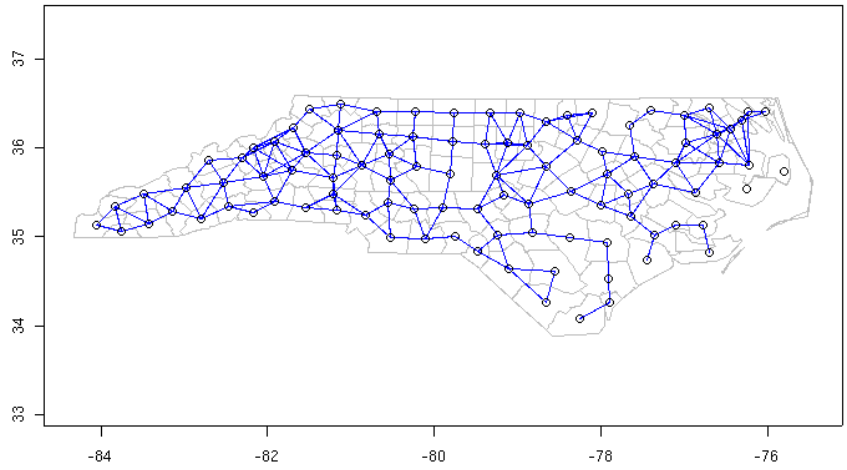

Figure 3: Overplotting shapefile boundaries with 30 mile neighbour relations as a graph.

object. Finally, we associate a vector of names with the neighbour list, through the `row.names` argument. The names should be unique, as with data frame row names.

```
> ncCC89.nb

Neighbour list object:
Number of regions: 100
Number of nonzero links: 394
Percentage nonzero weights: 3.94
Average number of links: 3.94
2 regions with no links:
2000 2099

> r.id <- attr(ncCC89.nb, "region.id")
> ncCC89.nb[[match("1825", r.id)]]

[1]  2 18 19

> r.id[ncCC89.nb[[match("1825", r.id)]]]

[1] 1827 1874 1880
```

The neighbour list object records neighbours by their order in relation to the list itself, so the neighbours list for the county with `region.id` "1825" are the second, eighteenth, and nineteenth in the list. We can retreive their codes by looking them up in the `region.id` attribute.

```
> nc.sids[card(ncCC89.nb) == 0, ]

     CNTY.ID BIR74 SID74 NWBIR74 BIR79 SID79 NWBIR79 east north
Dare    2000   521     0      43  1059     1      73  482   145
Hyde    2099   338     0     134   427     0     169  446   110
          x       y       lon      lat L.id M.id
Dare 439.65 3975.36 -75.66893 35.92070    2    4
Hyde 379.42 3920.72 -76.32823 35.42260    2    4
```

We should also note that this neighbour criterion generates two counties with no neighbours, Dare and Hyde, whose county seats were more than 30 miles from their nearest neighbours. The `card()` function returns the cardinality of the neighbour set. We need

to return to methods for handling no-neighbour objects later on. We will also show how new neighbours lists may be constructed in R, and compare these with those from the literature.

2.1 Probability mapping

Rather than review functions for measuring and modelling spatial dependence in the **spdep** package, we will focus on probability mapping for disease rates data. Typically, we have counts of the incidence of some disease by spatial unit, associated with counts of populations at risk. The task is then to try to establish whether any spatial units seem to be characterised by higher or lower counts of cases than might have been expected in general terms (Bailey and Gatrell, 1995).

An early approach by Choynowski (1959), described by Cressie and Read (1985) and Bailey and Gatrell (1995), assumes, given that the true rate for the spatial units is small, that as the population at risk increases to infinity, the spatial unit case counts are Poisson with mean value equal to the population at risk times the rate for the study area as a whole. Choynowski's approach folds the two tails of the measured probabilities together, so that small values, for a chosen α, occur for spatial units with either unusually high or low rates. For this reason, the high and low counties are plotted separately in Figure 4.

```
> ch <- choynowski(nc.sids$SID74, nc.sids$BIR74)
```

```
> plot(sidspolys, forcefill = FALSE)
> legend(c(-84, -81), c(33.9, 34.5), fill = grey(c(2, 5)/7),
+     legend = c("high", "low"), bty = "n", ncol = 2)
> plot(subset(sidspolys, ((ch$pmap < 0.05) & (ch$type))),
+     col = grey(5/7), add = TRUE, forcefill = FALSE)
> plot(subset(sidspolys, ((ch$pmap < 0.05) & (!ch$type))),
+     col = grey(2/7), add = TRUE, forcefill = FALSE)
```

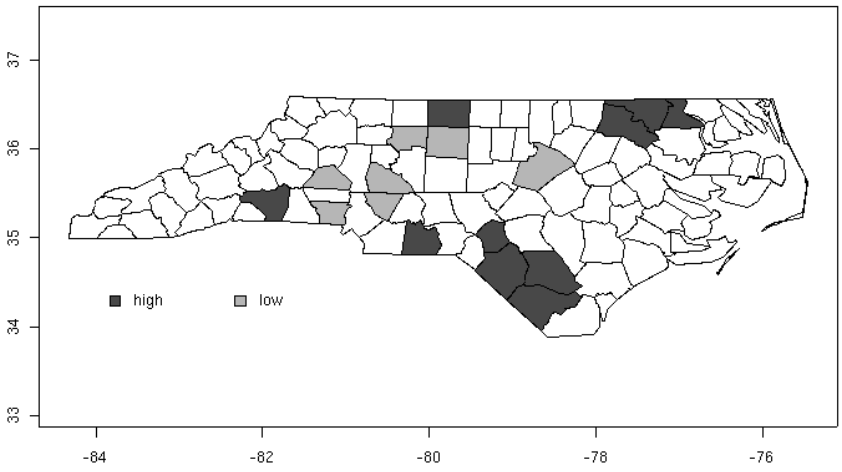

Figure 4: Probability map of North Carolina counties, SIDS cases 1974–78, $\alpha = 0.05$, reproducing Cressie and Read (1985), Figure 1.

For more complicated thematic maps, it may be helpful to use ColorBrewer (`http:`

//colorbrewer.org) colour palettes. Here we will only use the grey sequential palette, available in R in the **RColorBrewer** package (the colours are copied here to avoid loading the package).

While the choynowski() function only provides the probability map valyes required, the probmap() function returns raw (crude) rates, expected counts (assuming a constant rate across the study area), relative risks, and Poisson probability map values calculated using the standard cumulative distribution function ppois(). This does not fold the tails together, so that counties with lower observed counts than expected, based on population size, have values in the lower tail, and those with higher observed counts than expected have values in the upper tail, as Figure 5 shows.

```
> pmap <- probmap(nc.sids$SID74, nc.sids$BIR74)
> brks <- c(0, 0.001, 0.01, 0.025, 0.05, 0.95, 0.975, 0.99,
+     0.999, 1)
> cols <- c("#FFFFFF", "#F0F0F0", "#D9D9D9", "#BDBDBD",
+     "#969696", "#737373", "#525252", "#252525", "#000000")
> plot(sidspolys, col = cols[findInterval(pmap$pmap, brks)],
+     forcefill = FALSE)
> legend(c(-84, -81), c(33.9, 34.5), fill = cols, legend = leglabs(brks),
+     bty = "n", ncol = 2)
```

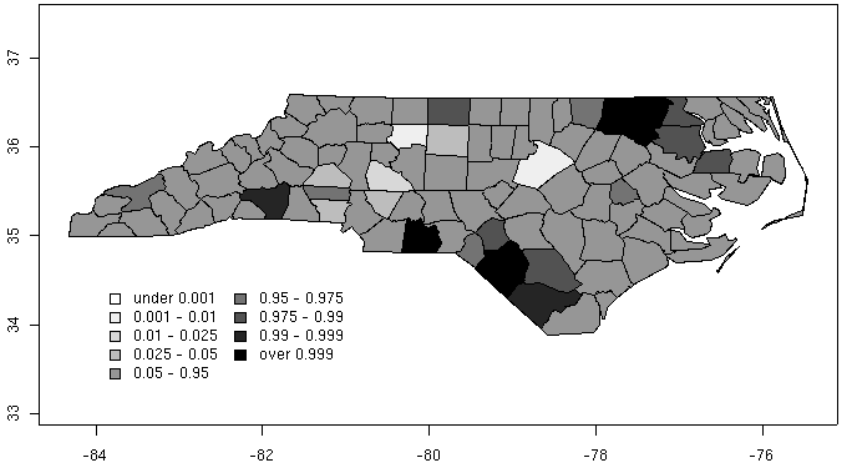

Figure 5: Probability map of North Carolina counties, SIDS cases 1974–78, reproducing Kaluzny et al. (1996), p. 67, Figure 3.28.

Marilia Carvalho (personal communication) and Virgilio Gómez Rubio (Gómez Rubio, Ferrándiz and López, 2003) have pointed to the unusual shape of the distribution of the Poisson probability values (Figure 6), repeating the doubts about probability mapping voiced by Cressie (1991, p. 392): "an extreme value . . . may be more due to its lack of fit to the Poisson model than to its deviation from the constant rate assumption". There are many more high values than one would have expected, suggesting perhaps overdispersion, that is that the ratio of the mean and variance is larger than unity.

```
> hist(pmap$pmap, main = "")
```

One ad-hoc way to assess the impact of the possible failure of our assumption that the counts follow the Poisson distribution is to estimate the dispersion by fitting a gen-

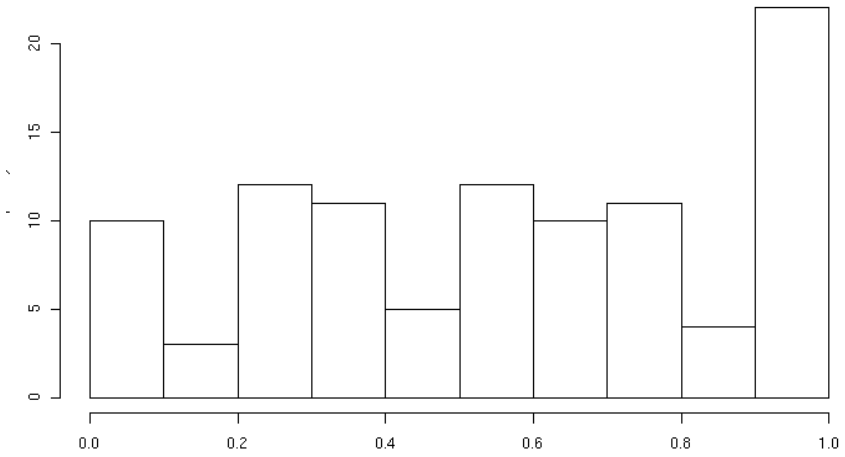

Figure 6: Histogram of Poisson probability values.

eral linear model of the observed counts including only the intercept (null model) and offset by the observed population at risk (suggested by Marilia Carvalho and associates):

```
> res <- glm(nc.sids$SID74 ~ offset(log(BIR74)), data = nc.sids,
+     family = "quasipoisson")
> stdres <- rstandard(res)
> brks <- c(-Inf, -2, -1.5, -1, 1, 1.5, 2, +Inf)
> cols <- c("#F7F7F7", "#D9D9D9", "#BDBDBD", "#969696",
+     "#737373", "#525252", "#252525")
> plot(sidspolys, col = cols[findInterval(stdres, brks)],
+     forcefill = FALSE)
> legend(c(-84, -81), c(33.9, 34.5), fill = cols, legend = leglabs(brks),
+     bty = "n", ncol = 2)
```

The dispersion is equal to 2.2784, much greater than unity; we calculate the corrected probability map values by taking the standardised residuals of the model, taking the size of the dispersion into account; the results are shown in Figure 7. Many fewer counties appear now to have unexpectedly large or small numbers of cases. This is an ad-hoc adjustment made because R provides access to a wide range of model-fitting functions that can be used to help check our assumptions. Gómez Rubio, Ferrándiz and López (2003) chose rather to construct a probability map under the hypothesis that data are drawn from a Negative Binomial.

So far, none of the maps presented have made use of the spatial dependence possibly present in the data. A further elementary step that can be taken is to map Empirical Bayes estimates of the rates, which are smoothed in relation to the raw rates. The underlying question here is linked to the larger variance associated with rate estimates for counties with small populations at risk compared with counties with large populations at risk. Empirical Bayes estimates place more credence on the raw rates of counties with large populations at risk, and modify them much less than they modify rates for small counties. In the case of small populations at risk, more confidence is placed in either the global rate for the study area as a whole, or for local Empirical Bayes estimates, in rates for a larger moving window including the neighbours of the

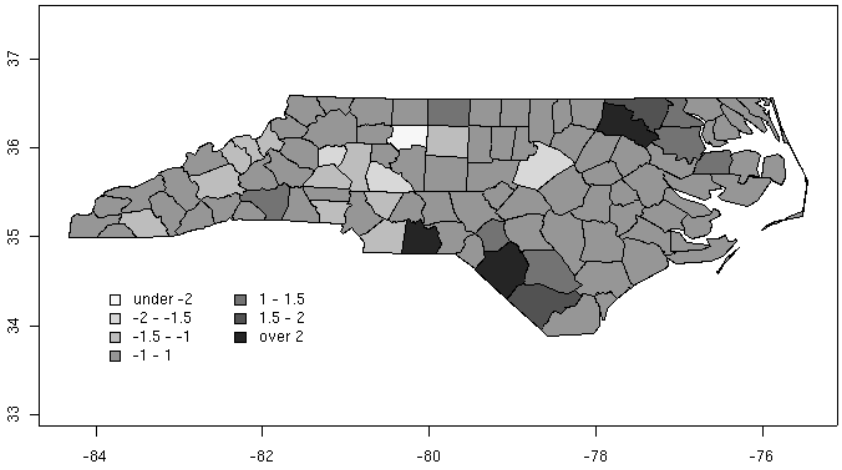

Figure 7: Standardised residual values from the fit of a quasi-Poisson fit of the null model for SIDS rates 1974-78, North Carolina counties.

county being estimated. The function used for this in **spdep** is `EBlocal()`, initially contributed by Marilia Carvalho. It parallels a similar function in GeoDa, but uses the Bailey and Gatrell (1995) interpretation of Marshall (1991), rather than that in GeoDa (Anselin, Syabri and Smirnov, 2002).

```
> res <- EBlocal(nc.sids$SID74, nc.sids$BIR74, ncCC89.nb,
+     zero.policy = TRUE)
> brks <- c(-Inf, 2, 2.5, 3, 3.5, Inf)
> cols <- c("#F7F7F7", "#CCCCCC", "#969696", "#636363",
+     "#252525")
> sub <- is.finite(res$est)
```

```
> plot(sidspolys, forcefill = FALSE)
> legend(c(-84, -81), c(33.9, 34.5), fill = cols, legend = leglabs(brks),
+     bty = "n", ncol = 2)
> plot(subset(sidspolys, sub), col = cols[findInterval(res$est[sub] *
+     1000, brks)], add = TRUE, forcefill = FALSE)
> points(sidscents[!sub, ], pch = 8)
```

The results are shown in Figure 8. Like other relevant functions in **spdep**, `EBlocal()` takes a `zero.policy` argument to allow missing values to be passed through. In this case, no local estimate is available for the two counties with no neighbours, marked by stars.

3 Preliminary exploration of the data (incomplete)

One of the first steps taken by Cressie and Read (1985) is to try to bring out spatial trends by dividing North Carolina up into 4×4 rough rectangles. Just to see how this works, let us map these rough rectangles before proceeding further (see Figure 9). We need to recall that the `nc.sids` data frame is not in the same order as the polygons.

```
> both <- factor(paste(nc.sids$L.id, nc.sids$M.id, sep = ":"))
> cols <- sample(rainbow(length(table(unclass(both)))))
```

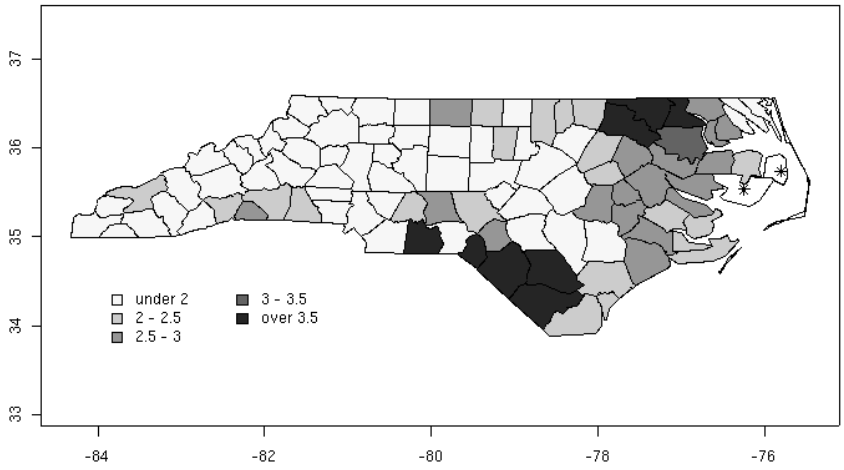

Figure 8: Local Empirical Bayes estimates for SIDS rates per 1000 using the 30 mile county seat neighbours list.

```
> plot(sidspolys, col = cols[both[order(nc.sids$CNTY.ID)]],
+     forcefill = FALSE)
> legend(c(-84, -81), c(33.5, 34.6), legend = levels(both),
+     fill = cols, bty = "n", cex = 0.9, y.intersp = 0.9,
+     ncol = 2)
```

(document to be extended in next release — terminates here to at least show how **maptools** and **spdep** can be used together).

References

Anselin, L., Syabri, I., Smirnov, O., 2002. Visualizing Multivariate Spatial Correlation with Dynamically Linked Windows. In Anselin, L., Rey, S. (Eds.), Proceedings, CSISS Workshop on New Tools for Spatial Data Analysis, Santa Barbara, CA, May 10-11, 2002. Center for Spatially Integrated Social Science, 20 pp., `http://sal.agecon.uiuc.edu/csiss/pdf/multilisa.pdf`.

Bailey, T. C., Gatrell, A. C., 1995. Interactive Spatial Data Analysis. Harlow: Longman, 413 pp.

Choynowski, M., 1959 Maps based on probabilities. Journal of the American Statistical Association, 54 (286), 385–388.

Cressie, N., 1991. Statistics for spatial data. New York: Wiley, 900 pp.

Cressie, N., Chan N. H., 1989. Spatial modelling of regional variables. Journal of the American Statistical Association, 84 (406), 393–401.

Cressie, N., Read, T. R. C., 1985. Do sudden infant deaths come in clusters?. Statistics and Decisions, Supplement Issue 2, 333–349.

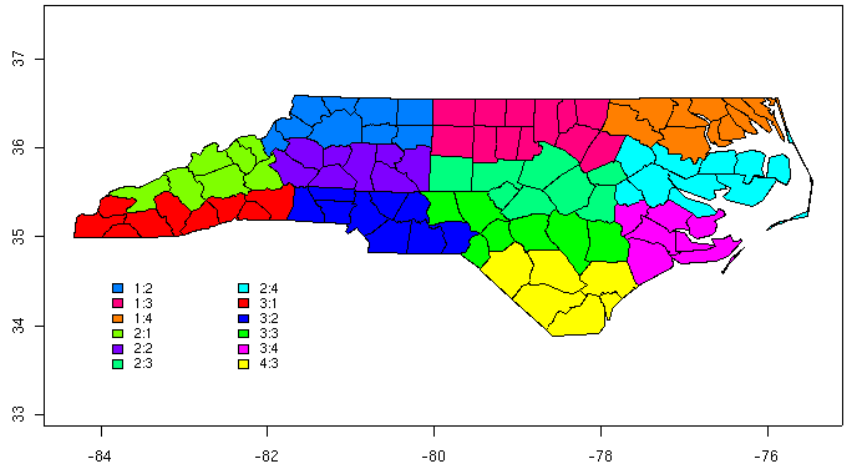

Figure 9: Rough rectangles used by Cressie and Read (1985) to bring out spatial trends.

Cressie, N., Read, T. R. C., 1989. Spatial data-analysis of regional counts. Biometrical Journal, 31 (6), 699–719.

Gómez Rubio, V., Ferrándiz, J., López, A., 2003 Detecting Disease Clusters with R. In: Hornik, K., Leisch, F., Zeilis, A. (Eds), Proceedings of the 3rd International Workshop on Distributed Statistical Computing, Vienna, Austria, 15 pp., (`http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/GomezRubioEtAl.pdf`).

Kaluzny, S. P., Vega, S. C., Cardoso, T. P., Shelly, A. A., 1996. `S-PLUS` SPATIAL-STATS user's manual version 1.0. Seattle: MathSoft Inc., 226 pp.

Marshall, R. M., 1991. Mapping disease and mortality rates using Empirical Bayes Estimators. Applied Statistics, 40 (2), 283–294.

Symons, M. J., Grimson, R. C., Yuan, Y. C., 1983. Clustering of rare events. Biometrics, 39 (1), 193–205.