Analysing Spatial Data in R: Worked examples: disease mapping I

Roger Bivand

Department of Economics
Norwegian School of Economics and Business Administration
Bergen, Norway

31 August 2007

Disease mapping

- Disease mapping is one of the areas of applied statistics that is developing most rapidly and is in most demand
- It involved both spatial data and methods ranging from visualisation to advanced statistics
- Waller and Gotway discuss many approaches, using, among other data sets, the classic Scottish lip cancer set by county
- Banerjee, Carlin and Gelfand also discuss the same data set, providing code for a WinBUGS example

Scottish lip cancer data set

Reading the data as downloaded from Waller and Gotway's website, we need first to import the county boundaries and assign the correct CRS, then transform to the British National Grid:

```
> library(rgdal)
> scot_LL <- readOGR(".", "scot")

OGR data source with driver: ESRI Shapefile
Source: ".", layer: "scot"
with  56  rows and  2  columns

> proj4string(scot_LL) <- CRS("+proj=longlat ellps=WGS84")
> EPSG <- make_EPSG()
> EPSG[grep("British National Grid", EPSG$note), 1:2]

      code                                note
2418 27700 # OSGB 1936 / British National Grid

> scot_BNG0 <- spTransform(scot_LL, CRS("+init=epsg:27700"))
```

Scottish lip cancer data set

The data as given by Clayton and Kaldor (1987) use the same county ID values, but in a different order, so we need to merge the data frame so as to output the same order as the county boundary polygons:

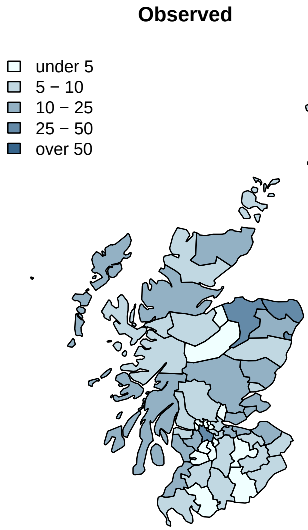
```
> library(maptools)
> scot_dat <- read.table("scotland.dat", skip = 1)
> names(scot_dat) <- c("District", "Observed", "Expected", "PcAFF",
+     "Latitude", "Longitude")
> row.names(scot_dat) <- formatC(scot_dat$District, width = 2,
+     flag = "0")
> ID <- formatC(scot_BNG0$ID, width = 2, flag = "0")
> scot_BNG1 <- spChFIDs(scot_BNG0, ID)
> scot_BNG <- spCbind(scot_BNG1, scot_dat[match(ID, row.names(scot_dat)),
+     ])
```

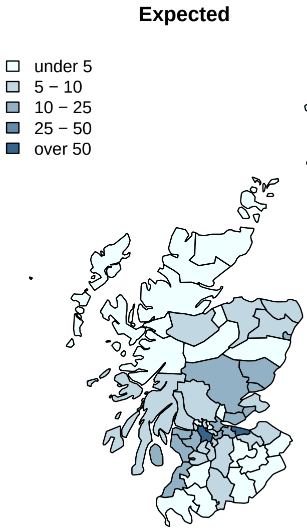
Scottish lip cancer data set

The original data table includes expected numbers of cases, which are based on age effects using the method of Mantel and Stark (1968), and are proportional to a “population at risk” after such effects have been taken into account. Setting up some palettes, we can look at maps of observed and expected counts:

```
> bluepal <- colorRampPalette(c("azure1", "steelblue4"))
> brks <- c(0, 5, 10, 25, 50, 100)
> library(classInt)
> O_CI <- classIntervals(scot_BNG$Observed, style = "fixed", fixedBreaks = brks)
> E_CI <- classIntervals(scot_BNG$Expected, style = "fixed", fixedBreaks = brks)
```

Lip cancer 1975–1980: observed and expected





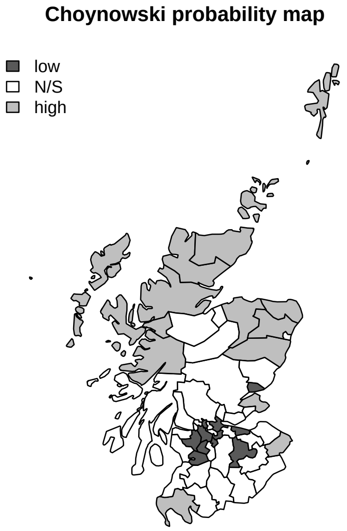
Probability maps and smoothing

- Choynowski proposed a method for making probability maps in 1959, still described by contemporary authors — it splits between upper and lower tails
- It is also possible to say just `ppois(Observed, Expected)` to get a similar result. If the observations are distributed as assumed, this will indicate which regions appear unusual
- Probability maps typically mirror the spatial pattern of relative risks; using Empirical Bayes smoothing will use the population at risk to adjust relative risks
- Empirical Bayes smoothing is implemented for the method-of-moments global and local cases in **spdep** and for the ML global case in **DCluster**

Choynowski probability map

The legacy Choynowski approach is implemented in spdep

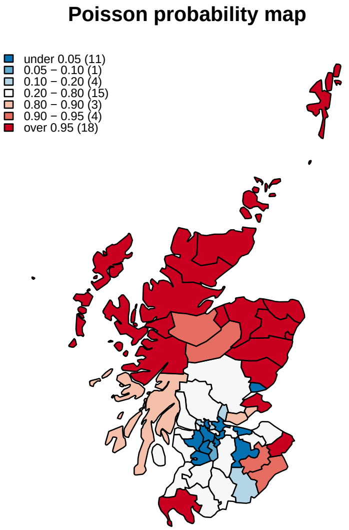
```
> library(spdep)
> ch <- choynowski(scot_BNG$Observed,
+     scot_BNG$Expected)
```



Poisson probability map

The Poisson probability map in one tail is identical with the Choynowski approach, but shows both tails rather than folding them together

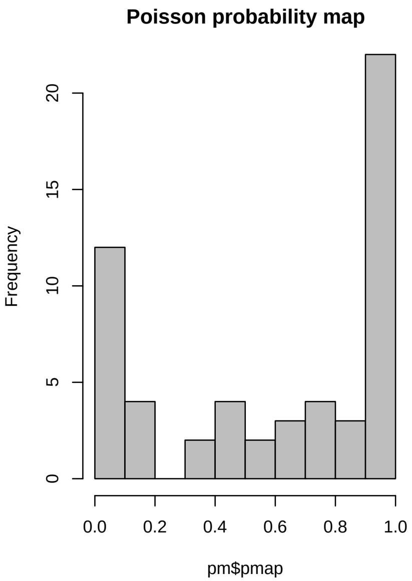
```
> pm <- probmap(scot_BNG$Observed,
+     scot_BNG$Expected)
> names(pm)

[1] "raw"      "expCount" "relRisk"
[4] "pmap"

> scot_BNG$SMR <- pm$relRisk
```



Weaknesses of probability maps

If the underlying distribution of the data does not agree with our assumption, we may get several possible processes mixed up, overdispersion with spatial dependence:

```
> table(findInterval(pm$pmap,
+     seq(0, 1, 1/10)))

 1  2  4  5  6  7  8  9 10
12  4  2  4  2  3  4  3 22
```



Empirical Bayes smoothing

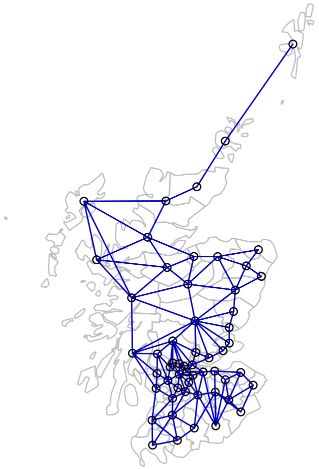
The method of moments approach is implemented in **spdep**, while the maximum likelihood approach is implemented in **DCluster**:

```
> eb1 <- EBest(scot_BNG$Observed, scot_BNG$Expected)
> unlist(attr(eb1, "parameters"))

        a         b
0.8027001 0.9996270

> scot_BNG$EB_mm <- eb1$estmm * 100
> library(DCluster)
> res <- empbaysmooth(scot_BNG$Observed, scot_BNG$Expected)
> unlist(res[2:3])

      nu    alpha
1.644016 1.148840

> scot_BNG$EB_ml <- res$smthrr * 100
```

Neighbours

In order to investigate spatial dependence, we need a list of neighbours. We take the list given by Clayton and Kaldor, re-ordering to suit out data order:

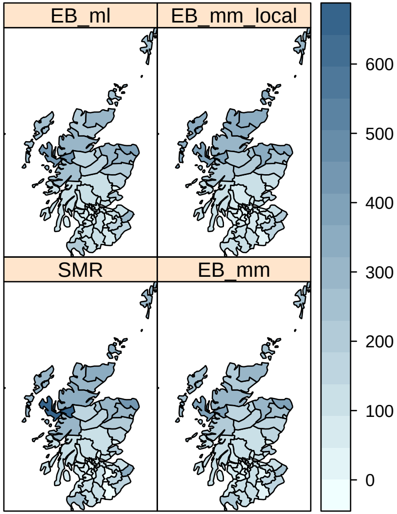
```
> CK_nb <- read.gal("CK.gal",
+     region.id = scot_BNG$District)
> CK_nb

Neighbour list object:
Number of regions: 56
Number of nonzero links: 264
Percentage nonzero weights: 8.418367
Average number of links: 4.714286
```



Local Empirical Bayes smoothing

If instead of shrinking to a global rate, we shrink to a local rate, we may be able to take unobserved heterogeneity into account; here we use the list of neighbours:

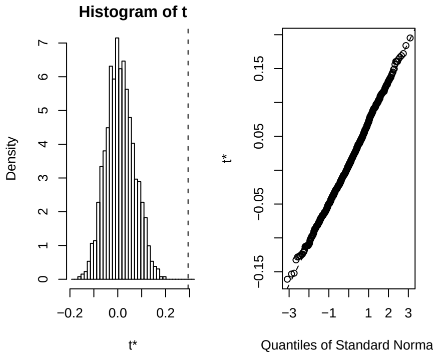
```
> eb2 <- EBlocal(scot_BNG$Observed,
+     scot_BNG$Expected, CK_nb)
> scot_BNG$EB_mm_local <- eb2$est *
+     100
```



Moran's *I*

DCluster provides a permutation bootstrap test for spatial autocorrelation of the difference between observed and expected counts:

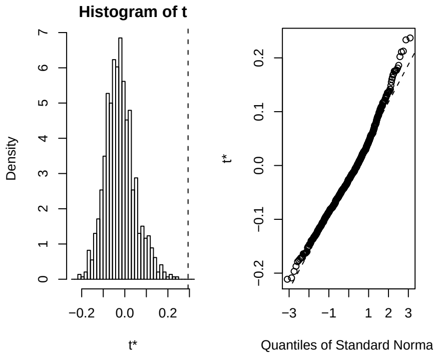
```
> lw <- nb2listw(CK_nb)
> set.seed(20060614)
> moran.boot <- boot(as(scot_BNG,
+     "data.frame"), statistic = moranI.boot,
+     R = 999, listw = lw, n = length(CK_nb),
+     S0 = Szero(lw))
```



Moran's *I*

It also provides parametric bootstraps for variants, including the Negative Binomial:

```
> moran.pgboot <- boot(as(scot_BNG,
+     "data.frame"), statistic = moranI.pboot,
+     sim = "parametric", ran.gen = negbin.sim,
+     R = 999, listw = lw, n = length(CK_nb),
+     S0 = Szero(lw))
```



Assunção and Reis' correction

The Assunção and Reis' correction to Moran's I is implemented in **spdep**:

```
> EBImoran.mc(scot_BNG$Observed, scot_BNG$Expected, lw, nsim = 999)

Monte-Carlo simulation of Empirical Bayes Index

data:  cases: scot_BNG$Observed, risk population: scot_BNG$Expected
weights: lw
number of simulations + 1: 1000

statistic = 0.7634, observed rank = 1000, p-value = 0.001
alternative hypothesis: greater
```

Fitting base GLM models

We can fit GLMs for the base model with only the intercept, for the Poisson, quasi-Poisson, and Negative Binomial, to give a starting point with respect to overdispersion:

```
> base.glm <- glm(Observed ~ 1 + offset(log(Expected)), data = scot_BNG,
+     family = poisson())
> base.glmQ <- glm(Observed ~ 1 + offset(log(Expected)), data = scot_BNG,
+     family = quasipoisson())
> library(MASS)
> base.nb <- glm.nb(Observed ~ 1 + offset(log(Expected)), data = scot_BNG)
> unlist(summary(base.nb)[20:21])

NULL
```

Tests for overdispersion

Tests for overdispersion, based in part on work by Dean, are provided in **DCluster**:

```
> test.nb.pois(base.nb, base.glm)

Likelihood ratio test for overdispersion

data:  base.nb : base.glm
LR = 225.5509, = 1, p-value < 2.2e-16
sample estimates:
      zscore p.mayor.modZ
-2.751195704  0.005937816

> DeanB(base.glm)

Dean's P_B test for overdispersion

data:  base.glm
P_B = 40.1049, p-value < 2.2e-16
alternative hypothesis: greater
```

Fitting GLMs

We can augment the base model with the percentage occupied in agriculture, forestry and fisheries, as a measure of exposure to sunlight, but this does not seem to alleviate overdispersion much:

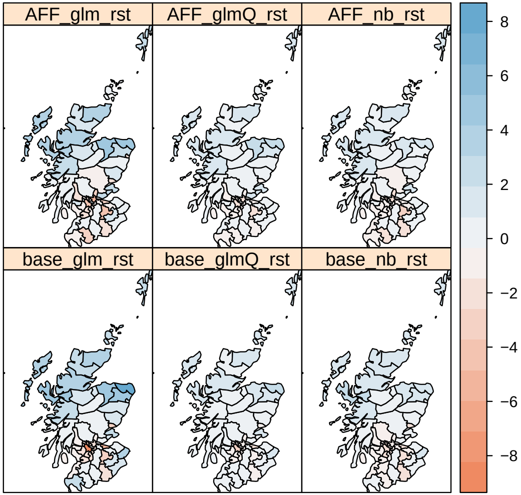
```
> AFF.glm <- glm(Observed ~ PcAFF + offset(log(Expected)), data = scot_BNG,
+     family = poisson())
> AFF.glmQ <- glm(Observed ~ PcAFF + offset(log(Expected)), data = scot_BNG,
+     family = quasipoisson())
> AFF.nb <- glm.nb(Observed ~ PcAFF + offset(log(Expected)), data = scot_BNG)
> unlist(summary(AFF.nb)[20:21])

NULL

> anova(base.nb, AFF.nb)

Likelihood ratio tests of Negative Binomial Models

Response: Observed
                          Model   theta Resid. df    2 x log-lik.   Test    df
1     1 + offset(log(Expected)) 1.87949        55       -363.1521
2 PcAFF + offset(log(Expected)) 2.98428        54       -342.9405 1 vs 2     1
  LR stat.      Pr(Chi)
1
2 20.21164 6.932991e-06
```

Residuals of the GLMs

In the same way that we "stacked up" smoothed rates, we can add the standardised residulats of the GLM fits to our `Spatial` object, to examine them visually for patterning:

```
> scot_BNG$base_glm_rst <- rstandard(base.glm)
> scot_BNG$base_glmQ_rst <- rstandard(base.glmQ)
> scot_BNG$base_nb_rst <- rstandard(base.nb)
> scot_BNG$AFF_glm_rst <- rstandard(AFF.glm)
> scot_BNG$AFF_glmQ_rst <- rstandard(AFF.glmQ)
> scot_BNG$AFF_nb_rst <- rstandard(AFF.nb)
```

Residuals of the GLMs



What next?

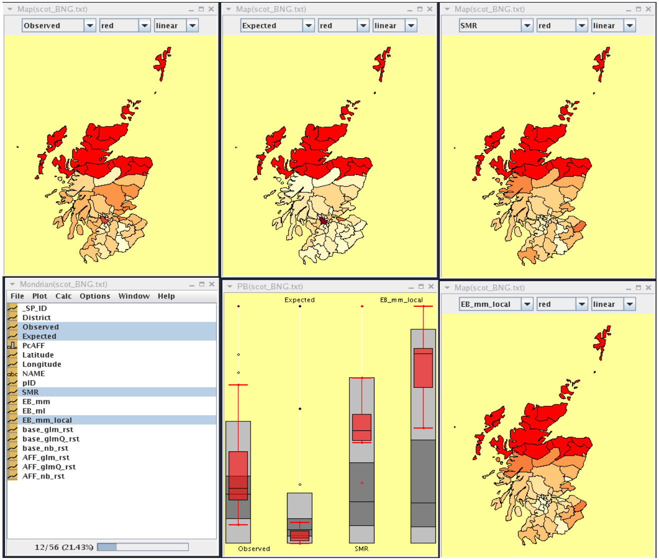
- We are pretty badly misspecified, but can the spatial dependence be separated from the distributional assumptions?
- There is a forthcoming paper in *Geographical Analysis* using permutation bootstrap on Moran's I of the deviance residuals of GLM fits, but this doesn't help with overdispersion
- Virgilio Gómez-Rubio has been working on exporting neighbour lists to Brugs/Openbugs and/or WinBUGS, and we are close to having something that can be released
- We can already export `SpatialPolygons` to WinBugs, but here this would require further manual intervention to set links to islands

Exporting the results

Finally, we write the results so far out as a shapefile, and as a text file to be read by Mondrian, since dynamic graphics may offer further insight:

```
> writePolyShape(scot_BNG, "scot_BNG")
> crs <- showWKT(proj4string(scot_BNG), "scot_BNG.prj")
> cat(strwrap(gsub(",", ", ", crs)), sep = "\n")

PROJCS["OSGB 1936 / British National Grid", GEOGCS["OSGB 1936",
DATUM["D_OSGB_1936", SPHEROID["Airy_1830", 6377563.396, 299.3249646]],
PRIMEM["Greenwich", 0], UNIT["Degree", 0.017453292519943295]],
PROJECTION["Transverse_Mercator"], PARAMETER["latitude_of_origin", 49],
PARAMETER["central_meridian", -2], PARAMETER["scale_factor",
0.9996012717], PARAMETER["false_easting", 400000],
PARAMETER["false_northing", -100000], UNIT["Meter", 1]]

> sp2Mondrian(scot_BNG, "scot_BNG.txt")
```

Exploring the results in Mondrian


Map(scot_BNG.txt): Observed, red, linear

Map(scot_BNG.txt): Expected, red, linear

Map(scot_BNG.txt): SMR, red, linear

Map(scot_BNG.txt): EB_mm_local, red, linear

Mondrian(scot_BNG.txt)

_SP_ID
District
Observed
Expected
PcAFF
Latitude
Longitude
NAME
pID
SMR
EB_mm
EB_ml
EB_mm_local
base_glm_rst
base_glmQ_rst
base_nb_rst
AFF_glm_rst
AFF_glmQ_rst
AFF_nb_rst

12/56 (21.43%)

Summing up

- Using spatial classes with shared methods for visualisation, import and export should let researchers get on with what you are good at
- We still need feedback on things that need improving, and contributions of richer objects to suit the different research domains
- Writing small functions to output spatial objects for other software turns out to be much easier, since we now know how the spatial objects are constructed
- Please feel free to use the mailing list to follow up — it is pretty likely that somebody else will have seen your problem already (the archives are good too)