SISMID Spatial Statistics in Epidemiology and Public Health
2016 R Notes: Clustering and Cluster Detection for Count Data

Jon Wakefield
Departments of Statistics and Biostatistics, University of Washington

2016-07-03

North Carolina SIDS Data

The `nc.sids` data frame has 100 rows and 21 columns and can be found in the `spdep` library.

It contains data given in Cressie (1991, pp. 386-9), Cressie and Read (1985) and Cressie and Chan (1989) on sudden infant deaths in North Carolina for 1974–78 and 1979–84.

The data set also contains the neighbour list given by Cressie and Chan (1989) omitting self-neighbours (`ncCC89.nb`), and the neighbour list given by Cressie and Read (1985) for contiguities (`ncCR85.nb`).

Data are available on the numbers of cases and on the number of births, both dichotomized by a binary indicator of race.

The data are ordered by county ID number, not alphabetically as in the source tables.

North Carolina SIDS Data

The code below plots the county boundaries along with the observed SMRs for 1974.

The expected numbers are based on internal standardization with a single stratum. So the single reference probability is the incidence of SIDS in 1974.

```
library(maptools)
library(spdep)
nc.sids <- readShapePoly(system.file("etc/shapes/sids.shp",
    package = "spdep")[1], ID = "FIPSNO",
    proj4string = CRS("+proj=longlat +ellps=clrk66"))
nc.sids2 <- nc.sids  # Create a copy, to add to
Y <- nc.sids$SID74
E <- nc.sids$BIR74 * sum(Y)/sum(nc.sids$BIR74)
nc.sids2$SMR74 <- Y/E
nc.sids2$EXP74 <- E
brks <- seq(0, 5, 1)
rm(nc.sids)  # We load another version of this later, so tidy up
```

SMR Plot

The map of the SMRs shows a number of counties with high relative risks (the risk relative to the state wide risk).

```
spplot(nc.sids2, "SMR74", at = brks,
    col.regions = grey.colors(5, start = 0.9,
        end = 0.1))
```

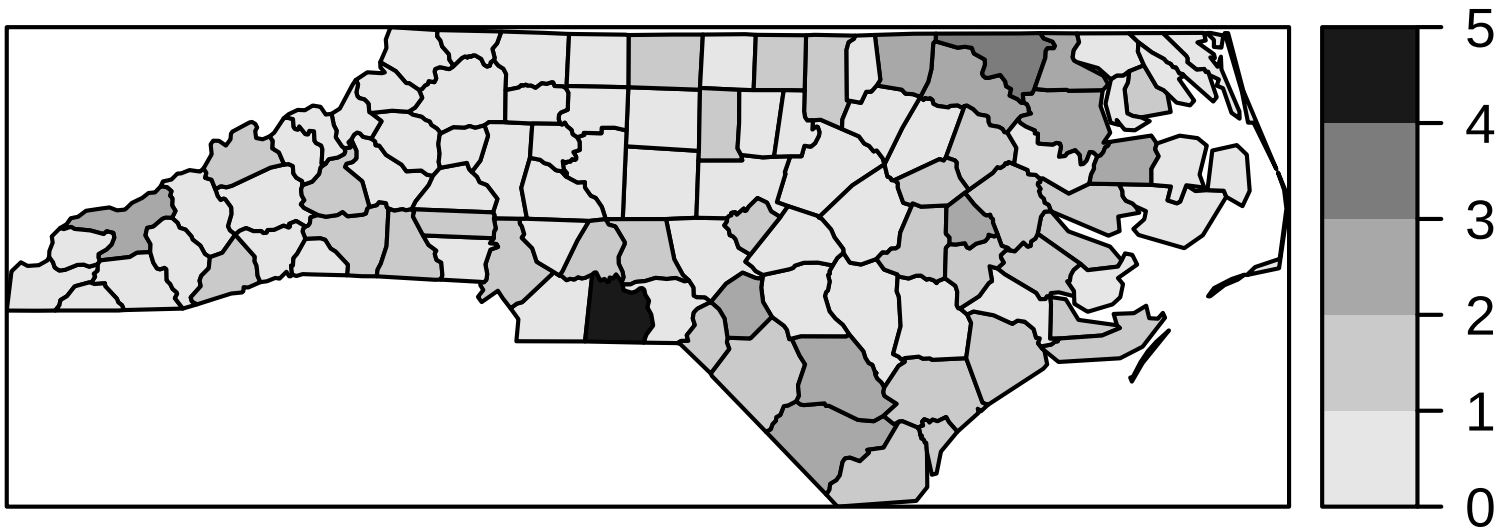


Figure 1: Map of SMRs for SIDS in 1974 in North Carolina

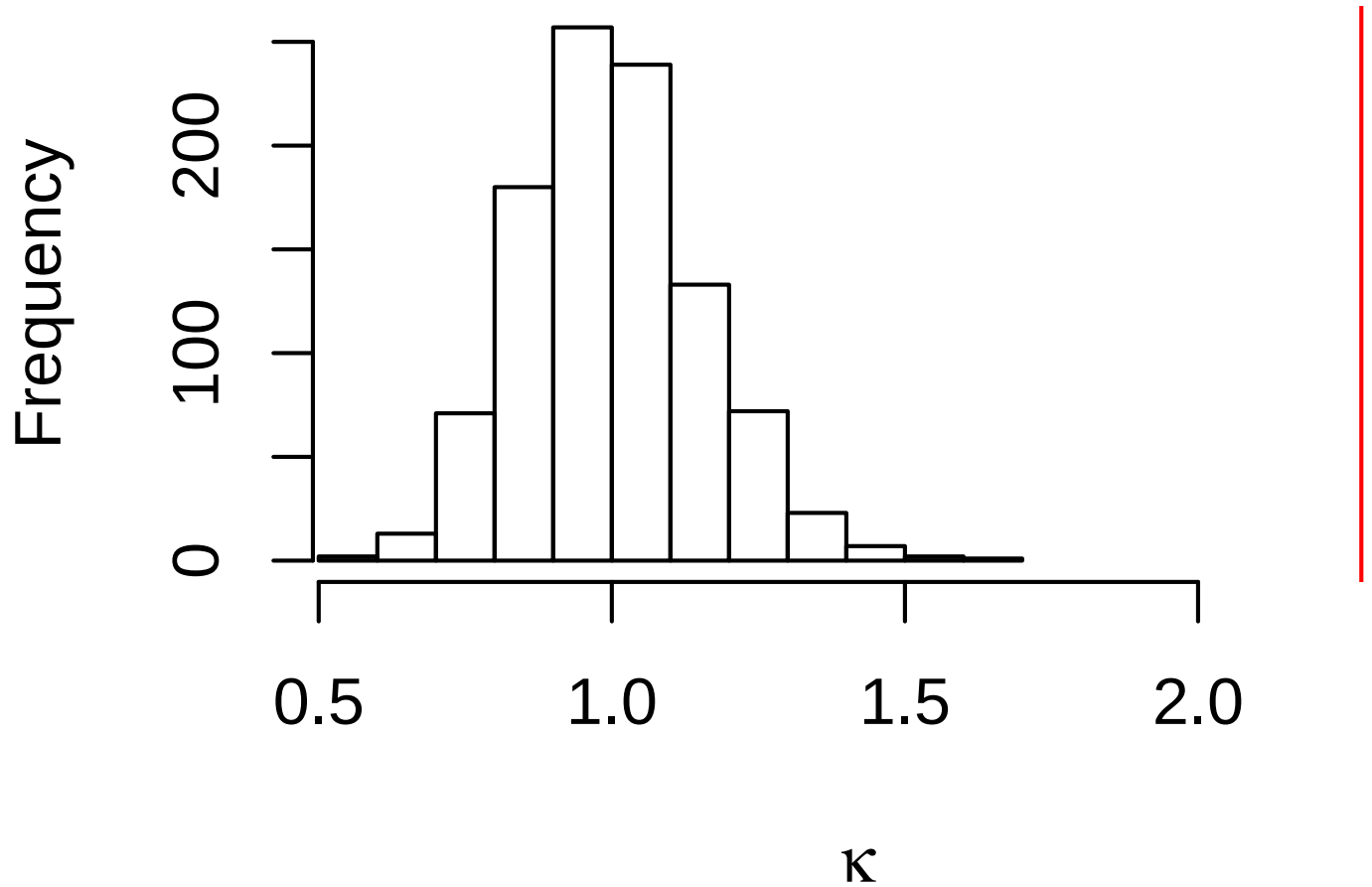
Overdispersion

Examine κ, the overdispersion statistic, and use a Monte Carlo test to examine significance.

```
library(spdep)
kappaval <- function(Y, fitted, df) {
    sum((Y - fitted)^2/fitted)/df
}
mod <- glm(Y ~ 1, offset = log(E), family = "quasipoisson")
kappaest <- kappaval(Y, mod$fitted, mod$df.resid)
nMC <- 1000
ncts <- length(E)
yMC <- matrix(rpois(n = nMC * ncts, lambda = E),
    nrow = ncts, ncol = nMC)
kappaMC <- NULL
for (i in 1:nMC) {
    modMC <- glm(yMC[, i] ~ 1, offset = log(E),
        family = "quasipoisson")
    kappaMC[i] <- kappaval(yMC[, i], modMC$fitted,
        modMC$df.resid)
}
```

Overdispersion: $\widehat{\kappa}$ is significantly different from 1

```
hist(kappaMC, xlim = c(min(kappaMC),
    max(kappaMC, kappaest)), main = "",
    xlab = expression(kappa))
abline(v = kappaest, col = "red")
```



Disease Mapping

We first fit a non-spatial random effects model:

$$\begin{aligned} Y_i|\beta_0, \epsilon_i &\sim_{iid} \text{Poisson}(E_i e^{\beta_0+\epsilon_i}), \\ \epsilon_i|\sigma_\epsilon^2 &\sim_{iid} N(0, \sigma_\epsilon^2) \end{aligned}$$

with the default priors on β_0 and σ_ϵ^2.

```
library(INLA)
nc.sids2$ID <- 1:100
m0 <- inla(SID74 ~ f(ID, model = "iid"),
    family = "poisson", E = EXP74, data = as.data.frame(nc.sids2
    control.predictor = list(compute = TRUE))
```

The `control.predictor` argument indicates we want fitted values.

Disease Mapping

Examine the first few "fitted values", summaries of the posterior distribution of $\exp(\beta_0 + \epsilon_i)$, $\quad i = 1, \ldots, n$.

```
head(m0$summary.fitted.values)
##                            mean        sd 0.025quant  0.5quant 0.975quant
## fitted.predictor.001 1.2515021 0.2930181  0.7548490 1.2250844   1.899824
## fitted.predictor.002 0.7665958 0.2700582  0.3481650 0.7299039   1.397177
## fitted.predictor.003 0.9149708 0.3494437  0.3989681 0.8598644   1.751025
## fitted.predictor.004 2.7309425 0.7626511  1.5074088 2.6400575   4.470065
## fitted.predictor.005 0.9027425 0.3177245  0.4165809 0.8575336   1.650257
## fitted.predictor.006 0.8544442 0.3152039  0.3789292 0.8076757   1.601193
##                            mode
## fitted.predictor.001 1.1747221
## fitted.predictor.002 0.6631748
## fitted.predictor.003 0.7637463
## fitted.predictor.004 2.4583333
## fitted.predictor.005 0.7763712
## fitted.predictor.006 0.7245109
```

Disease Mapping

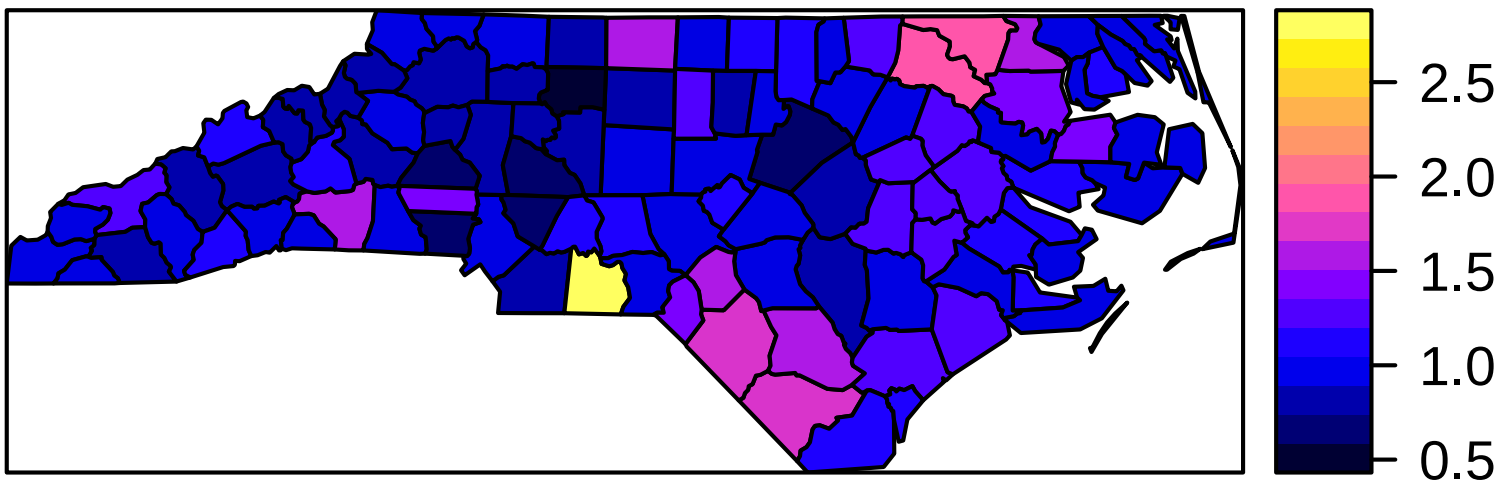
Create two interesting inferential summaries:

- the posterior mean of the relative risk
- a binary indicator of whether the posterior median is greater than 1.5 (which we assume is an epidemiologically significant value). This value can be changed, based on the context.

```
nc.sids2$RRpmean0 <- m0$summary.fitted.values[,
    1]
nc.sids2$RRind0 <- m0$summary.fitted.values[,
    4] > 1.5
```

Disease Mapping

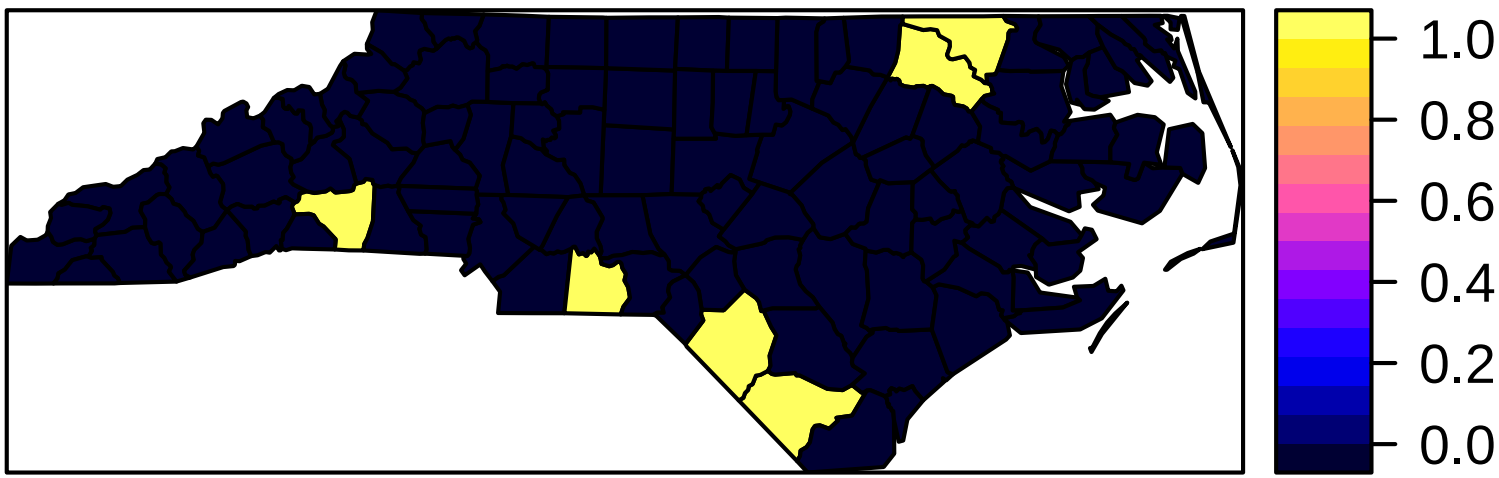
```
# Display posterior means of
# relative risks
spplot(nc.sids2, "RRpmean0")
```



A number of counties have high mean values.

Disease Mapping

```
# Display indicators of whether 0.5
# points above 1.5
spplot(nc.sids2, "RRind0")
```



Six counties have posterior median relative risk greater than 1.

Disease Mapping with IID and ICAR random effects

We now fit a model with non-spatial and ICAR spatial random effects.

```
nc.sids2$ID2 <- 1:100
m1 <- inla(SID74 ~ 1 + f(ID, model = "iid") + f(ID2,
    model = "besag", graph = "examples/NC.graph"),
    family = "poisson", E = EXP74, data = as.data.frame(nc.sids2
    control.predictor = list(compute = TRUE))
# Define summary quantities of interest as with iid
# model
nc.sids2$RRpmean1 <- m1$summary.fitted.values[, 1]
nc.sids2$RRind1 <- m1$summary.fitted.values[, 4] >
    1.5
```

Disease Mapping with IID and ICAR random effects

Summarize the IID+ICAR model

```
summary(m1)
##
## Call:
## c("inla(formula = SID74 ~ 1 + f(ID, model = \"iid\") + f(ID2, model = \"besag\", ",  "    graph = \"exa
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          0.9511          0.4588          0.0746          1.4845
##
## Fixed effects:
##                mean    sd 0.025quant 0.5quant 0.975quant    mode kld
## (Intercept) -0.0554 0.054    -0.1642  -0.0545     0.0479 -0.0524   0
##
## Random effects:
## Name   Model
##  ID   IID model
## ID2   Besags ICAR model
##
## Model hyperparameters:
##                          mean        sd 0.025quant  0.5quant 0.975quant
## Precision for ID  17946.355 1.751e+04   1204.782 12805.631   64499.90
## Precision for ID2     2.299 8.844e-01      1.092     2.125       4.50
##                         mode
## Precision for ID  3275.258
## Precision for ID2    1.823
##
## Expected number of effective parameters(std dev): 34.19(5.792)
## Number of equivalent replicates : 2.925
##
## Marginal log-Likelihood:  -314.28
## Posterior marginals for linear predictor and fitted values computed
```

Disease Mapping with IID and ICAR random effects

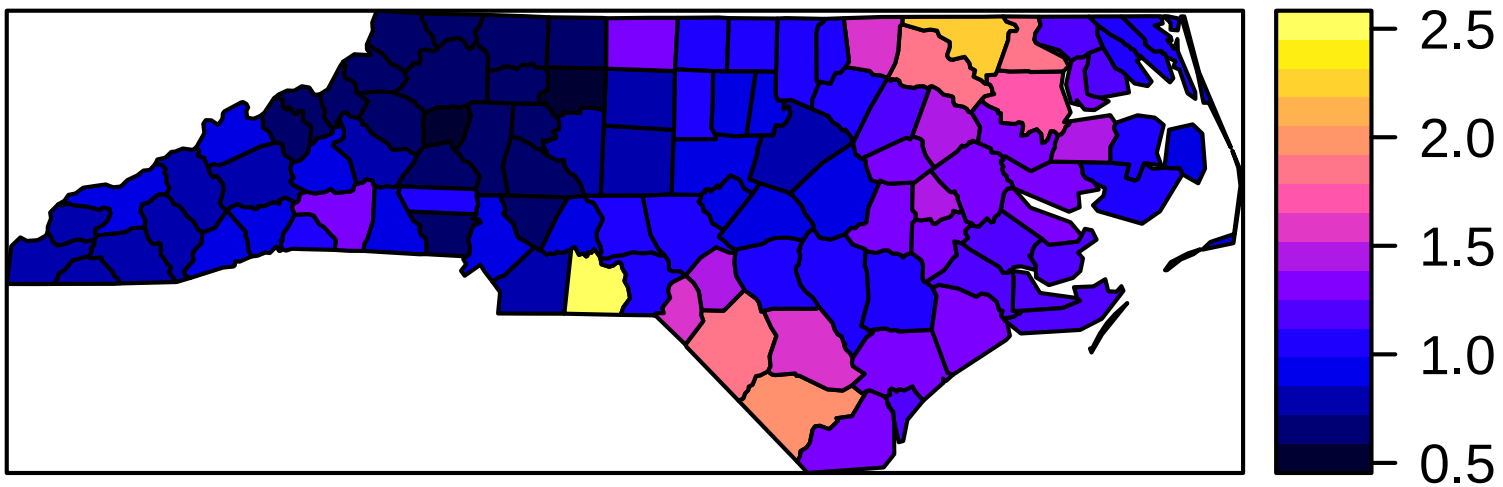
As an aside, if we wanted to create a neighbour list based on regions with contiguous boundaries we can use the `poly2nb` function in the `spdep` library.

```
nc.sids <- readShapePoly(system.file("etc/shapes/sids.shp",
    package = "spdep")[1])
# Create adjacency matrix
nc.nb <- poly2nb(nc.sids)
nb2INLA("inlanc.graph", nc.nb)  # Slighty different to NC.graph
rm(nc.sids)
```

Disease Mapping with IID and ICAR random effects

Display posterior means of relative risks.

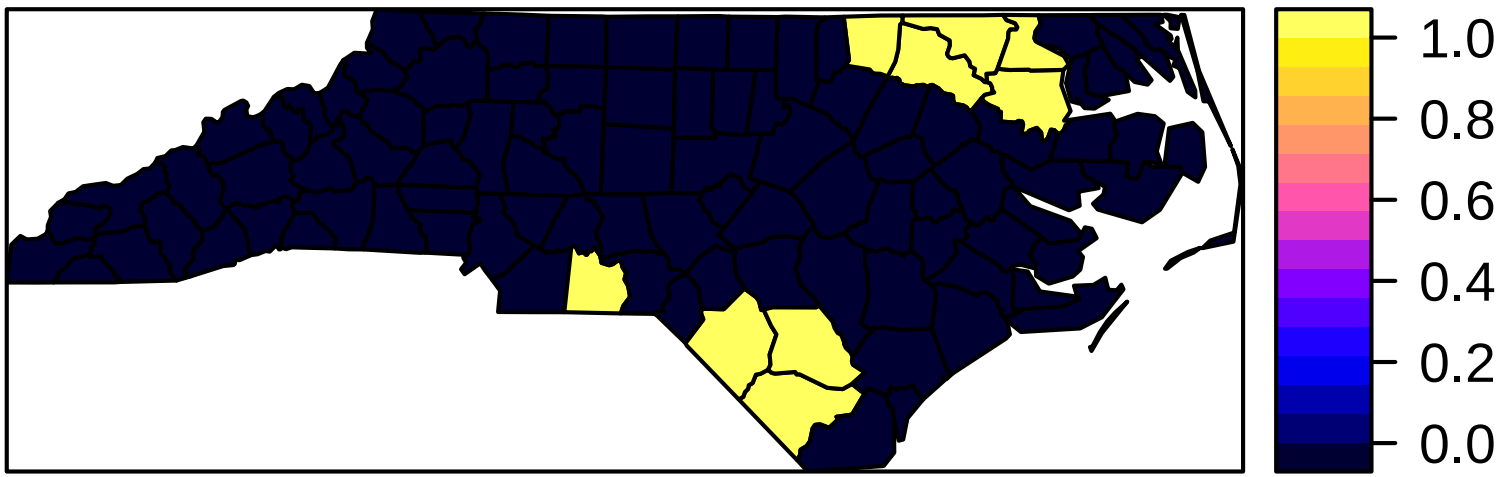
```
spplot(nc.sids2, "RRpmean1")
```



Disease Mapping with IID and ICAR random effects

Display areas with medians above 1.5, ie those areas with greater than 50% chance of exceedence of 1.5.

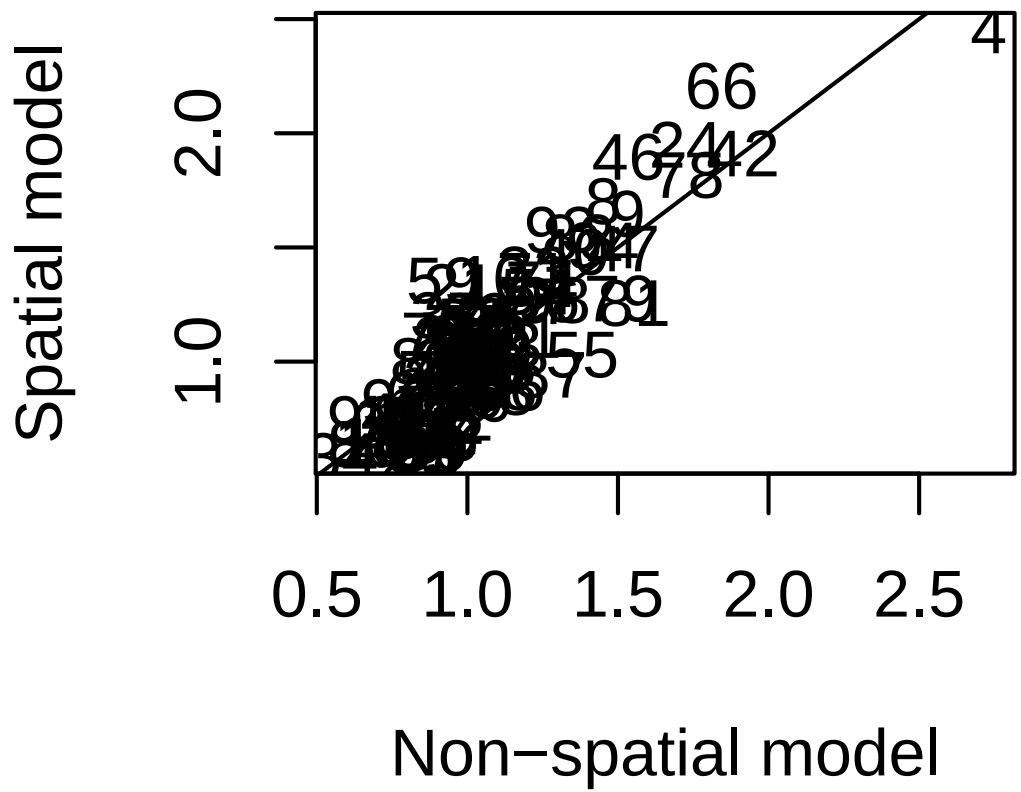
```
spplot(nc.sids2, "RRind1")
```



Both summaries show differences with the iid only model, with changes in the obvious direction. The spatial smoothing model gives a larger collection in the north-east, for example, and a single area in the west is not highlighted. As we will see later, the spatial model is supported by the data in this example.

Disease Mapping: Comparison of posterior mean RRs

```
plot(nc.sids2$RRpmean1 ~ nc.sids2$RRpmean0,
     type = "n", xlab = "Non-spatial model",
     ylab = "Spatial model")
text(nc.sids2$RRpmean1 ~ nc.sids2$RRpmean0)
abline(0, 1)
```



Disease mapping with IID and ICAR random effects

We now examine the variances of the spatial and non-spatial random effects.

Recall that the ICAR model variance has a conditional interpretation.

To obtain a rough estimate of the marginal variance we obtain the posterior median of the S_i's and evaluate their variance.

From the output below, we conclude that the spatial random effects dominate for the SIDS data so that we conclude there is clustering of cases in neighboring areas.

Proportion of variation that is spatial

```
nareas <- 100
mat.marg <- matrix(NA, nrow = nareas, ncol = 1000)
m <- m1$marginals.random$ID2
for (i in 1:nareas) {
    Sre <- m[[i]]
    mat.marg[i, ] <- inla.rmarginal(1000, Sre)
}
var.Sre <- apply(mat.marg, 2, var)
var.eps <- inla.rmarginal(1000, inla.tmarginal(function(x)
    m1$marginals.hyper$"Precision for ID"))
mean(var.Sre)
## [1] 0.1989025
mean(var.eps)
## [1] 0.0001426175
perc.var.Sre <- mean(var.Sre/(var.Sre + var.eps))
perc.var.Sre
## [1] 0.9992725
```

Clustering via Moran's *I*

We evaluate Moran's test for spatial autocorrelation using the "W" style weight function: this standardizes the weights so that for each area the weights sum to 1. Also define the "B" style for later.

To obtain a variable with approximately constant variance we form residuals from an intercept only model.

```
library(spdep)
# Note the nc.sids loaded from the data() command
# is in a different order to that obtained from the
# shapefile
data(nc.sids)
col.W <- nb2listw(ncCR85.nb, style = "W", zero.policy = TRUE)
col.B <- nb2listw(ncCR85.nb, style = "B", zero.policy = TRUE)
rm(nc.sids)
quasipmod <- glm(SID74 ~ 1, offset = log(EXP74), data = nc.sids2,
    family = quasipoisson())
sidsres <- residuals(quasipmod, type = "pearson")
```

Clustering via Moran's *I*

```
moran.test(sidsres, col.W)
##
##  Moran I test under randomisation
##
## data:  sidsres
## weights: col.W
##
## Moran I statistic standard deviate = 2.4351, p-value = 0.0074
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      0.147531140      -0.010101010       0.004190361
```

This analysis suggests significant clustering.

Clustering via Moran's *I*

Moran's test may suggest spatial autocorrelation if there exists a non-constant mean function.

Hence, we should endeavor to remove the large-scale trends.

Below we fit a model with Eastings and Northings (of the County seat) as covariates – both show some at least some association.

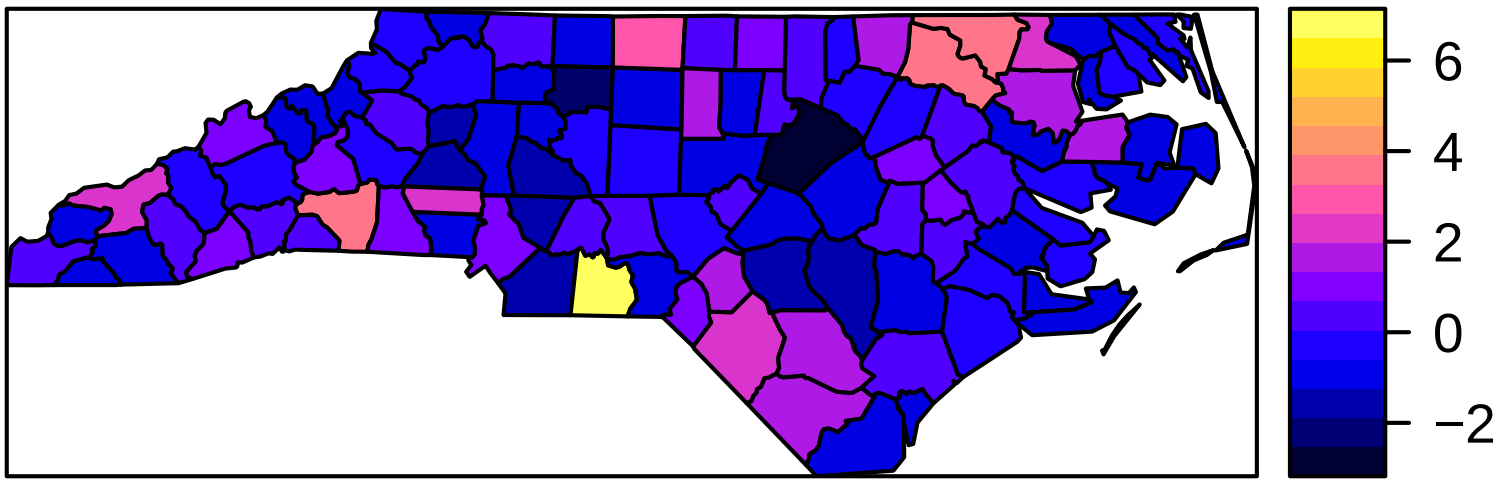
Clustering via Moran's *I*

```
quasipmod2 <- glm(SID74 ~ east + north, offset = log(EXP74),
    data = nc.sids2, family = quasipoisson())
summary(quasipmod2)
##
## Call:
## glm(formula = SID74 ~ east + north, family = quasipoisson(),
##     data = nc.sids2, offset = log(EXP74))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7961  -1.0249  -0.3475   0.6043   4.7261
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.2465437  0.2680159  -0.920  0.35992
## east         0.0020105  0.0006469   3.108  0.00247 **
## north       -0.0028032  0.0014545  -1.927  0.05687 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 2.039456)
##
##     Null deviance: 203.34  on 99  degrees of freedom
## Residual deviance: 171.80  on 97  degrees of freedom
## AIC: NA
##
```

Clustering via Moran's *I*

We map the residuals to get a visual on the clustering.

```
sidsres2 <- residuals(quasipmod2, type = "pearson")
nc.sids2$res <- sidsres2
par(mar = c(0.1, 0.1, 0.1, 0.1))
spplot(nc.sids2, "res")
```



Clustering via Moran's *I*

The significance of the Moran statistic is reduced, though still significant if judged by conventional levels.

```
moran.test(sidsres2, col.W)
##
##  Moran I test under randomisation
##
## data:  sidsres2
## weights: col.W
##
## Moran I statistic standard deviate = 2.1328, p-value = 0.0164
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      0.127428361      -0.010101010       0.004157993
```

Neighborhood options

There are various coding schemes for the weights.

B has 0/1 corresponding to non-neighbor/neighbor – this means areas with many neighbors are more influential.

W has rows standardized by the number of neighbors so that the sum for each row (area) is unity.

Weights can be more complex, depending on inverse distance, for example. See Bivand et al. (2013, Section 9.2).

Neighborhood options

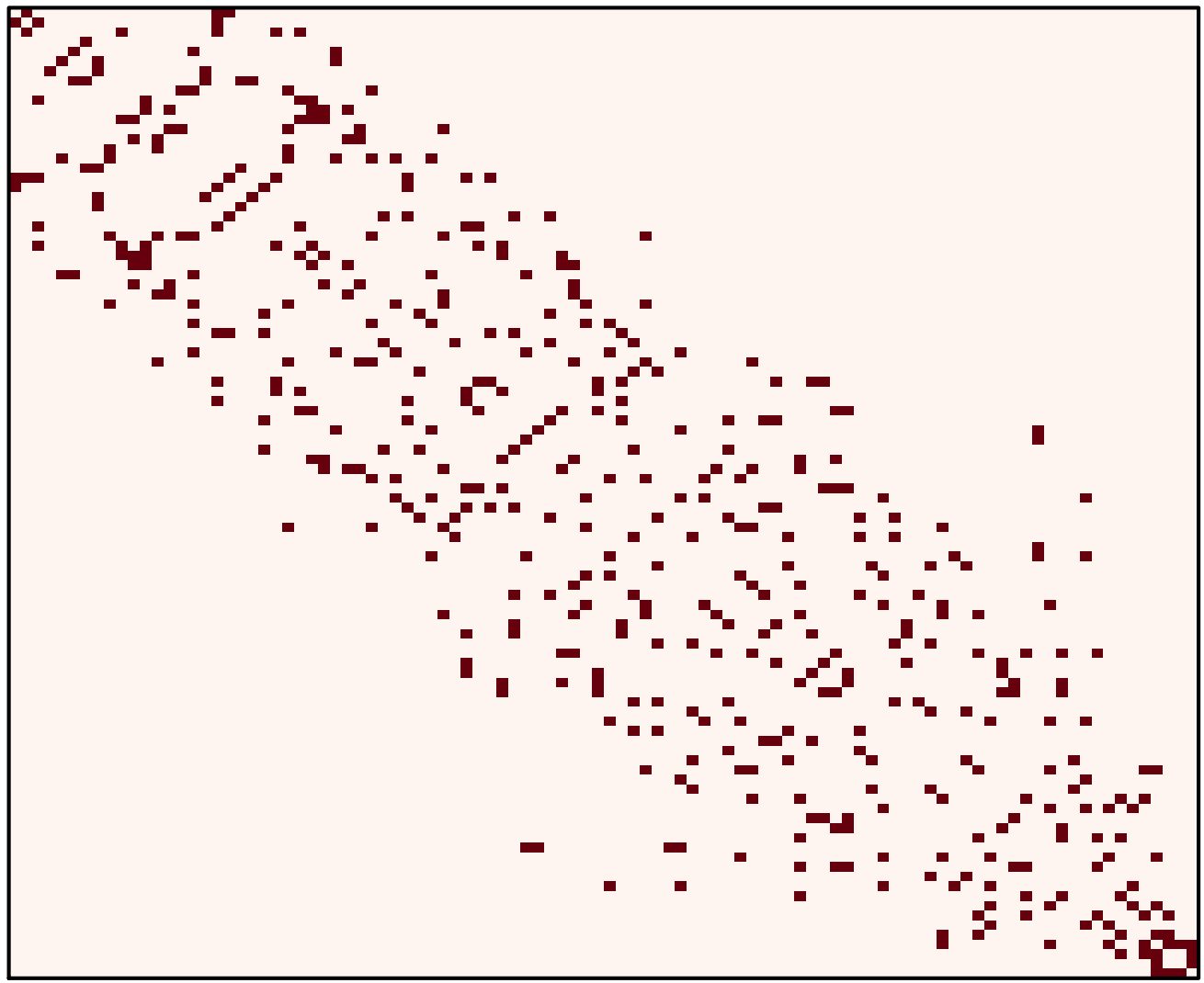
```
library(RColorBrewer)
pal <- brewer.pal(9, "Reds")
z <- t(listw2mat(col.W))
brks <- c(0, 0.1, 0.143, 0.167, 0.2, 0.5, 1)
nbr3 <- length(brks) - 3
image(1:100, 1:100, z[, ncol(z):1], breaks = brks,
    col = pal[c(1, (9 - nbr3):9)], main = "W style",
    axes = FALSE)
box()
z <- t(listw2mat(col.B))
brks <- c(0, 0.1, 0.143, 0.167, 0.2, 0.5, 1)
nbr3 <- length(brks) - 3
image(1:100, 1:100, z[, ncol(z):1], breaks = brks,
    col = pal[c(1, (9 - nbr3):9)], main = "B style",
    axes = FALSE)
box()
```

Neighborhood Options

Neighborhood Options

B style

1:100



Moran's *I* with a different neighborhood structure

We now use Moran's statistic on the detrended residuals, but with the binary "B" weight option. This option has unstandardized weights.

The conclusion, evidence of spatial autocorrelation, is the same as with the standardized weights option.

```
moran.test(sidsres2, col.B)
##
##  Moran I test under randomisation
##
## data:  sidsres2
## weights: col.B
##
## Moran I statistic standard deviate = 2.2357, p-value = 0.0126
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      0.125344196      -0.010101010       0.003670354
```

Clustering via Geary's c

We now use Geary's statistic on the detrended residuals, and come to the same conclusion

```
geary.test(sidsres2, col.W)
##
##  Geary C test under randomisation
##
## data:  sidsres2
## weights: col.W
##
## Geary C statistic standard deviate = 2.3479, p-value = 0.0094
## alternative hypothesis: Expectation greater than statistic
## sample estimates:
## Geary C statistic       Expectation          Variance
##         0.8195420         1.0000000         0.0059072
```

North Carolina SIDS Data: Clustering Conclusions

The disease mapping model shows that almost all of the residual variation is spatial.

Both of the Moran's I and Geary's c methods suggest that there is evidence of clustering in these data.

So all methods in agreement!

Cluster detection

We now turn to cluster detection: detecting areas or contiguous collections of areas that appear to be at elevated risk.

We concentrate on the `SatScan` method but for illustration show the methods of Openshaw and Besag and Newell in action.

The results from these two methods are difficult to interpret formally (Openshaw's in particular) because of the multiple testing problem.

Cluster detection: Openshaw

We implement Openshaw's method using the centroids of the areas in data.

Circles of radius 30 are used and the centers are placed on a grid of size 10.

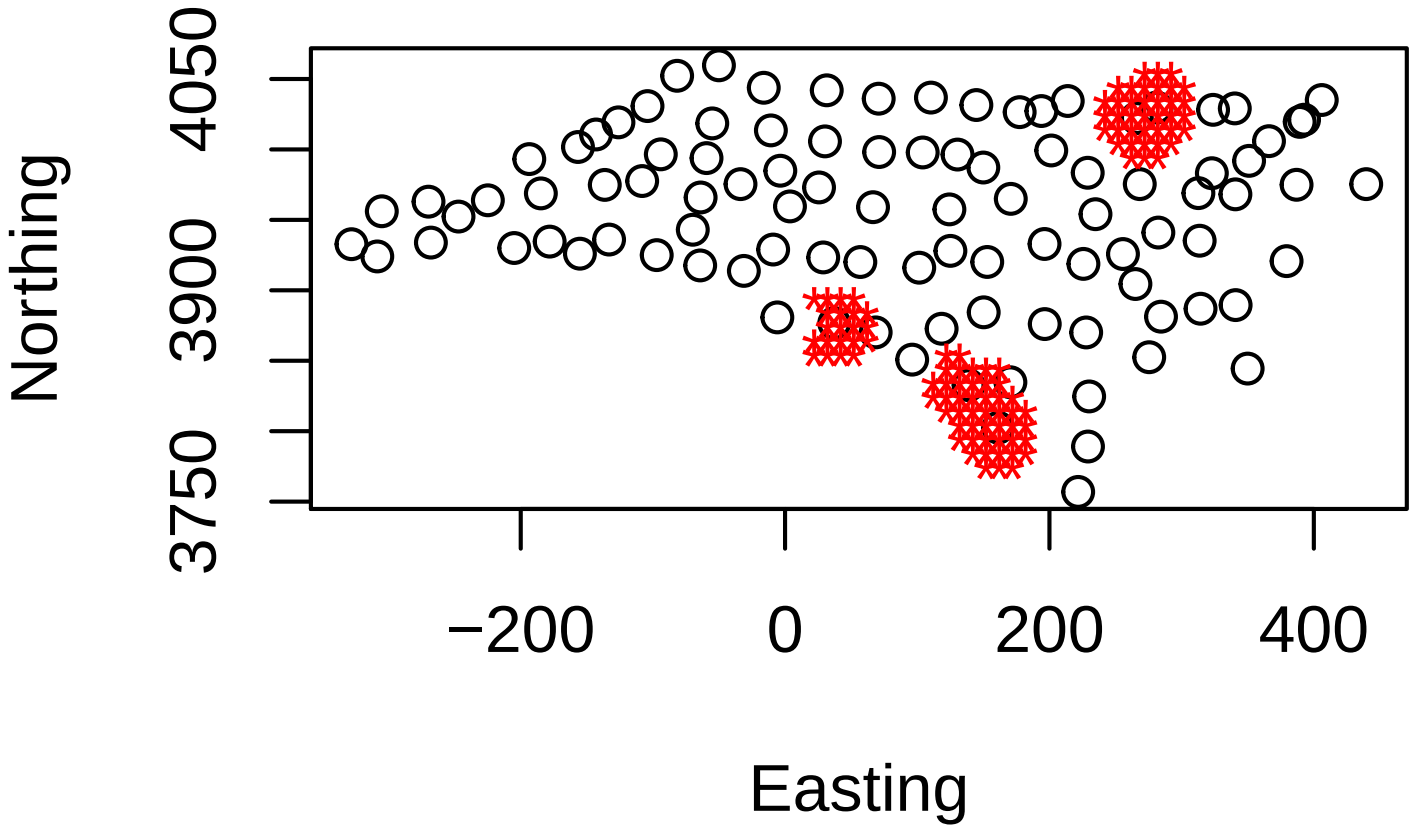
For multiple radii, multiple calls are required.

The significance level for calling a cluster is 0.002.

```
library(spdep)
data(nc.sids)
sids <- data.frame(Observed = nc.sids$SID74)
sids <- cbind(sids, Expected = nc.sids$BIR74 * sum(nc.sids$SID74)/sum(nc.sids$B
sids <- cbind(sids, x = nc.sids$x, y = nc.sids$y)
# GAM
library(DCluster)
sidsgam <- opgam(data = sids, radius = 30, step = 10,
    alpha = 0.002)
names(sidsgam)
## [1] "x"         "y"         "statistic" "cluster"   "pvalue"    "size"
dim(sidsgam)
## [1] 106   6
```

Cluster detection: Openshaw

```
plot(sids$x, sids$y, xlab = "Easting", ylab = "Northing")
# Plot centroids of locations flagged as clusters
points(sidsgam$x, sidsgam$y, col = "red", pch = "*")
```




```
rm(nc.sids)
```

Clustering via Openshaw

A subset of the 106 signinificant Openshaw results.

```
head(sidsgam, n = 15)
##          x       y statistic cluster       pvalue size
## 1   151.96 3776.92        15       1 1.743356e-03    1
## 2   161.96 3776.92        15       1 1.743356e-03    1
## 3   171.96 3776.92        15       1 1.743356e-03    1
## 4   141.96 3786.92        15       1 1.743356e-03    1
## 5   151.96 3786.92        15       1 1.743356e-03    1
## 6   161.96 3786.92        15       1 1.743356e-03    1
## 7   171.96 3786.92        15       1 1.743356e-03    1
## 8   181.96 3786.92        15       1 1.743356e-03    1
## 9   131.96 3796.92        15       1 1.743356e-03    1
## 10 141.96 3796.92        15       1 1.743356e-03    1
## 11 151.96 3796.92        15       1 1.743356e-03    1
## 12 161.96 3796.92        15       1 1.743356e-03    1
## 13 171.96 3796.92        15       1 1.743356e-03    1
## 14 181.96 3796.92        15       1 1.743356e-03    1
## 15 131.96 3806.92        46       1 5.531787e-06    2
```

Cluster detection: Besag and Newell

We illustrate the Besag and Newell method using $k = 20$ as the cluster size.

```
library(maptools)
library(maps)
library(ggplot2)
library(sp)
nc.sids <- readShapePoly(system.file("etc/shapes/sids.shp",
    package = "spdep")[1], ID = "FIPSNO",
    proj4string = CRS("+proj=longlat +ellps=clrk66"))
referencep <- sum(nc.sids$SID74)/sum(nc.sids$BIR74)
population <- nc.sids$BIR74
cases <- nc.sids$SID74
E <- nc.sids$BIR74 * referencep
SMR <- cases/E
n <- length(cases)
```

Cluster detection: Besag and Newell

We need to form a matrix containing the centroids.

```
getLabelPoint <- function(county) {
    Polygon(county[c("long", "lat")])@labpt
}
df <- map_data("county", "north carolina")  # NC region county d
centNC <- by(df, df$subregion, getLabelPoint)  # Returns list
centNC <- do.call("rbind.data.frame", centNC)  # Convert to Data
names(centNC) <- c("long", "lat")  # Appropriate Header
centroids <- matrix(0, nrow = n, ncol = 2)
for (i in 1:n) {
    centroids[i, ] <- c(centNC$lat[i], centNC$long[i])
}
colnames(centroids) <- c("x", "y")
rownames(centroids) <- 1:n
```

SpatialEpi package

The most recent version of `SpatialEpi` can be installed from github (see below for the relevant command).

Otherwise the usual `install.packages("SpatialEpi")` should be used.

Cluster detection: Besag and Newell

```
# devtools::install_github('rudeboybert/SpatialEpi')
library(SpatialEpi)
k <- 20
alpha.level <- 0.01
geo <- centroids
BNresults <- besag_newell(geo, population, cases, expected.cases = NULL,
    k, alpha.level)
BNsig <- length(BNresults$p.values[BNresults$p.values <
    alpha.level])
cat("No of sig results = ", BNsig, "\n")
## No of sig results =  11
resmat <- matrix(NA, nrow = BNsig, ncol = 100)
reslen <- NULL
for (i in 1:length(BNresults$clusters)) {
    reslen[i] <- length(BNresults$clusters[[i]]$location.IDs.included)
    resmat[i, 1:reslen[i]] <- BNresults$clusters[[i]]$location.IDs.included
}
```

Cluster detection: Besag and Newell

Now we set up the polygons for plotting the results.

```
NCTemp <- map("county", "north carolina",
    fill = TRUE, plot = FALSE)
NCIDs <- substr(NCTemp$names, 1 + nchar("north carolina,"),
    nchar(NCTemp$names))
NC <- map2SpatialPolygons(NCTemp, IDs = NCIDs,
    proj4string = CRS("+proj=longlat"))
# Fix currituck county which is 3 islands
index <- match(c("currituck:knotts", "currituck:main",
    "currituck:spit"), NCIDs)
currituck <- list()
for (i in c(27:29)) currituck <- c(currituck,
    list(Polygon(NC@polygons[[i]]@Polygons[[1]]@coords)))
currituck <- Polygons(currituck, ID = "currituck")
```

Cluster detection: Besag and Newell

```
# make new spatial polygons object
NC.new <- NC@polygons[1:(index[1] - 1)]
NC.new <- c(NC.new, currituck)
NC.new <- c(NC.new, NC@polygons[(index[3] +
    1):length(NC@polygons)])
NC.new <- SpatialPolygons(NC.new, proj4string = CRS("+proj=longlat"))
NCIDs <- c(NCIDs[1:(index[1] - 1)], "currituck",
    NCIDs[(index[3] + 1):length(NC@polygons)])
NC <- NC.new
```

Cluster detection: Besag and Newell

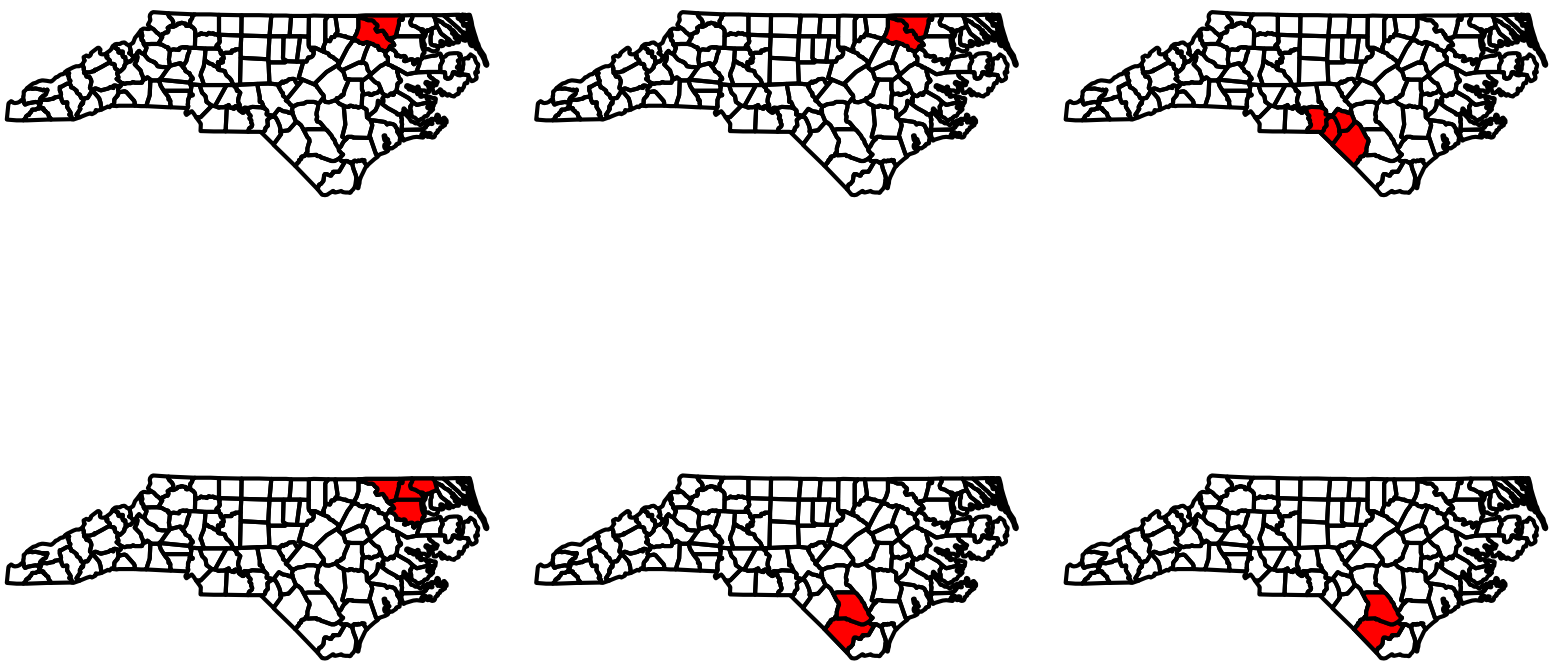
```
# SANITY CHECK: Reorder Spatial Polygons of list to
# match order of county
names <- rep("", 100)
for (i in 1:length(NC@polygons)) names[i] <- NC@polygons[[i]]@ID
identical(names, NCIDs)
## [1] FALSE

index <- match(NCIDs, names)
NC@polygons <- NC@polygons[index]
rm(index)

names <- rep("", 100)
for (i in 1:length(NC@polygons)) names[i] <- NC@polygons[[i]]@ID
identical(names, NCIDs)
## [1] TRUE
```

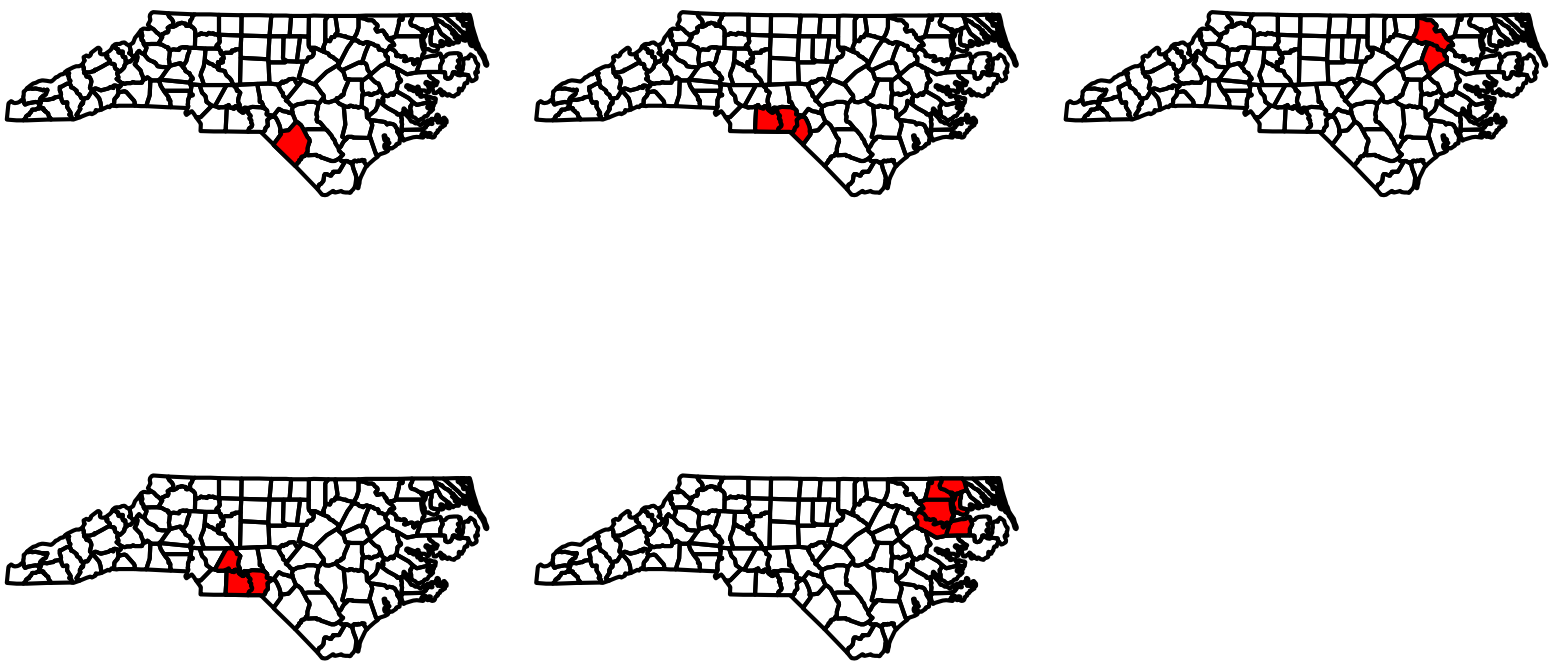
Cluster detection: Besag and Newell

```
par(mfrow = c(3, 3), mar = c(0.1, 0.1, 0.1, 0.1))
for (i in 1:6) {
    plot(NC.new)
    plot(NC.new[resmat[i, c(1:reslen[i])]], col = "red",
        add = T)
}
```



Cluster detection: Besag and Newell

```
par(mfrow = c(3, 3), mar = c(0.1, 0.1, 0.1, 0.1))
for (i in 7:11) {
    plot(NC.new)
    plot(NC.new[resmat[i, c(1:reslen[i])]], col = "red",
        add = T)
}
```



Cluster detection: SatScan method

We now turn to SatScan and set 20% as the upper bound on the proportion of the population to be contained in any one potential cluster.

```
# SpatialEpi implementation of SatScan
pop.upper.bound <- 0.2
n.simulations <- 999
alpha.level <- 0.05
Kpoisson <- kulldorff(geo, cases, population, expected.cases = N
    pop.upper.bound, n.simulations, alpha.level, plot = T)
Kcluster <- Kpoisson$most.likely.cluster$location.IDs.included
```

Cluster detection: SatScan method

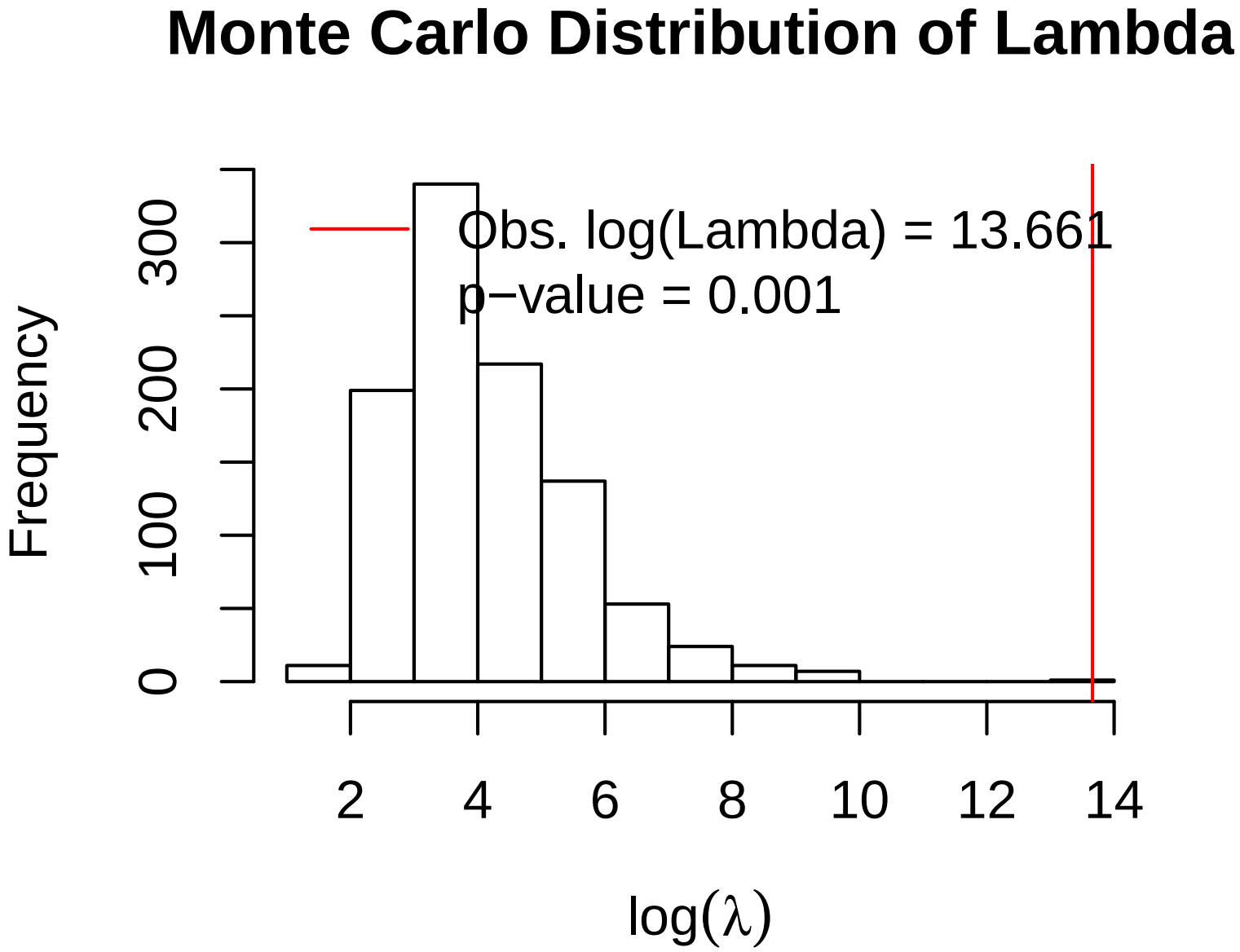
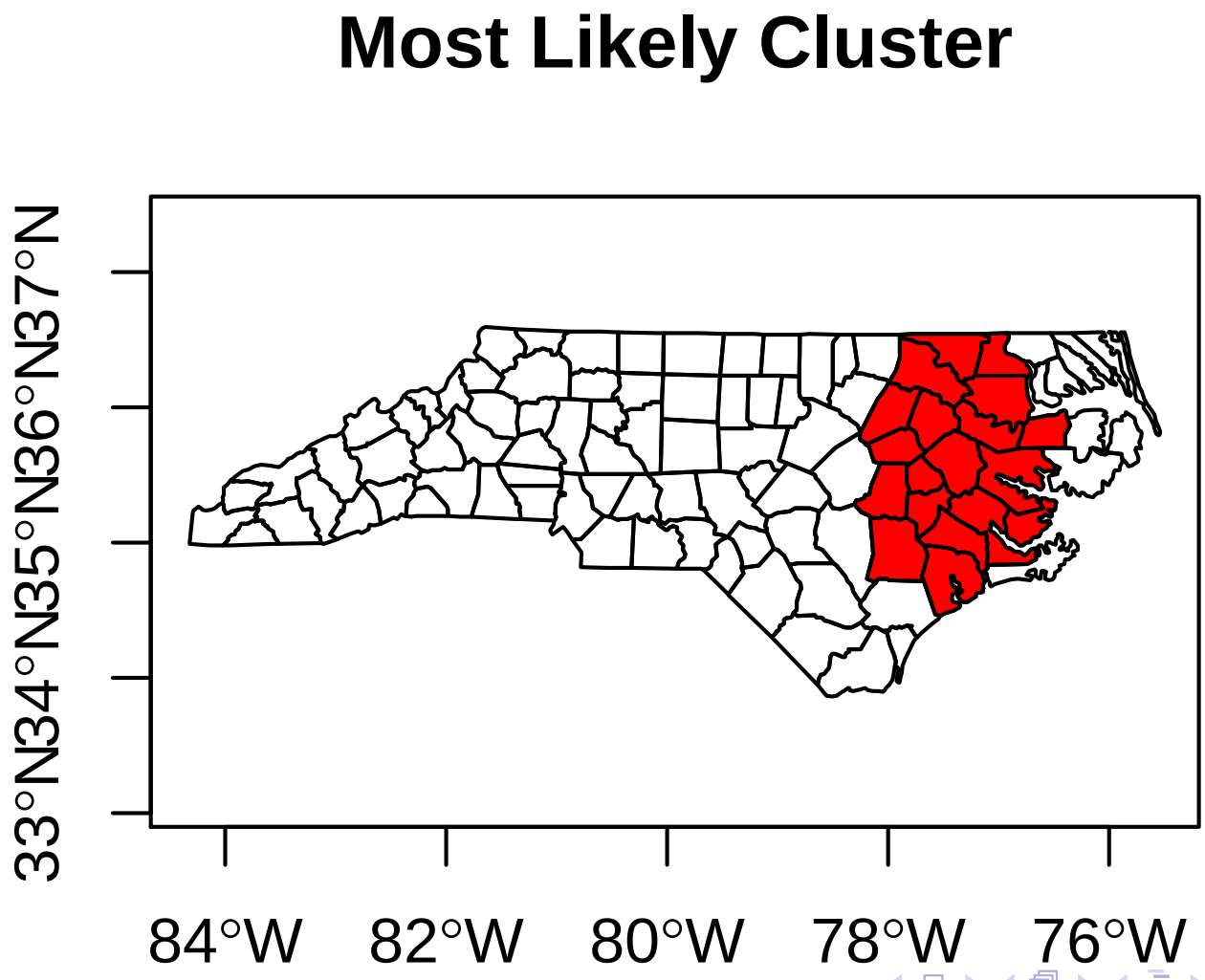


Figure 4:

Cluster detection: SatScan method

```
plot(NC.new, axes = TRUE)
plot(NC.new[Kcluster], add = TRUE, col = "red")
title("Most Likely Cluster")
```



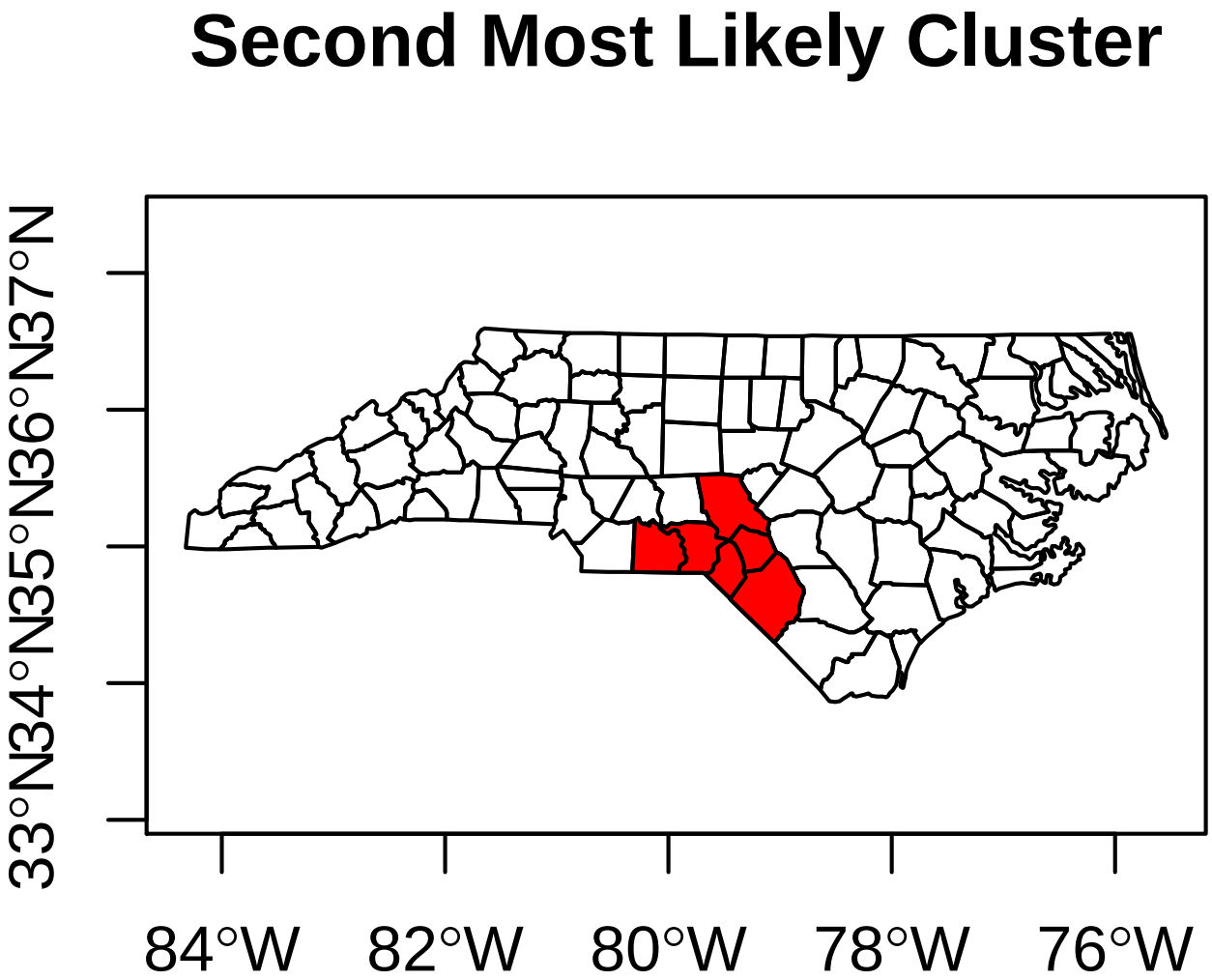
Cluster detection: SatScan method

Now look at secondary clusters.

Two are significant, and indicated in the figures below,

```
K2cluster <- Kpoisson$secondary.clusters[[1]]$location.IDs.inclu
plot(NC.new, axes = TRUE)
plot(NC.new[K2cluster], add = TRUE, col = "red")
title("Second Most Likely Cluster")
```

Cluster detection: SatScan method



Bayes cluster model

```
# Load NC map and obtain geographic
# centroids
library(maptools)
sp.obj <- readShapePoly(system.file("etc/shapes/sids.shp",
    package = "spdep")[1], ID = "FIPSNO",
    proj4string = CRS("+proj=longlat +ellps=clrk66"))
centroids <- latlong2grid(coordinates(sp.obj))
```

Bayes cluster model: running the MCMC algorithm

```
y <- sp.obj$SID74
population <- sp.obj$BIR74
E <- expected(population, y, 1)
max.prop <- 0.2
k <- 5e-05
shape <- c(2976.3, 2.31)
rate <- c(2977.3, 1.31)
J <- 7
pi0 <- 0.95
n.sim.lambda <- 0.5 * 10^4
n.sim.prior <- 0.5 * 10^4
n.sim.post <- 0.5 * 10^5
output <- bayes_cluster(y, E, population, sp.obj, centroids,
    max.prop, shape, rate, J, pi0, n.sim.lambda, n.sim.prior,
    n.sim.post)
## [1] "Algorithm started on: Sun Jul  3 10:42:37 2016"
## [1] "Importance sampling of lambda complete on: Sun Jul  3 10:42:41 2016"
## [1] "Prior map MCMC complete on: Sun Jul  3 10:42:43 2016"
## [1] "Posterior estimation complete on: Sun Jul  3 10:44:25 2016"
```

Bayes cluster model

```
SMR <- y/E
plotmap(SMR, sp.obj, nclr = 6, location = "bottomleft",
    leg.cex = 0.5)
plotmap(output$prior.map$high.area, sp.obj, nclr = 6,
    location = "bottomleft", leg.cex = 0.5)
plotmap(output$post.map$high.area, sp.obj, nclr = 6,
    location = "bottomleft", leg.cex = 0.5)
barplot(output$pk.y, names.arg = 0:J, xlab = "k", ylab = "P(k|y)
plotmap(output$post.map$RR.est.area, sp.obj, log = TRUE,
    nclr = 6, location = "bottomleft", leg.cex = 0.5)
```

Bayes cluster model

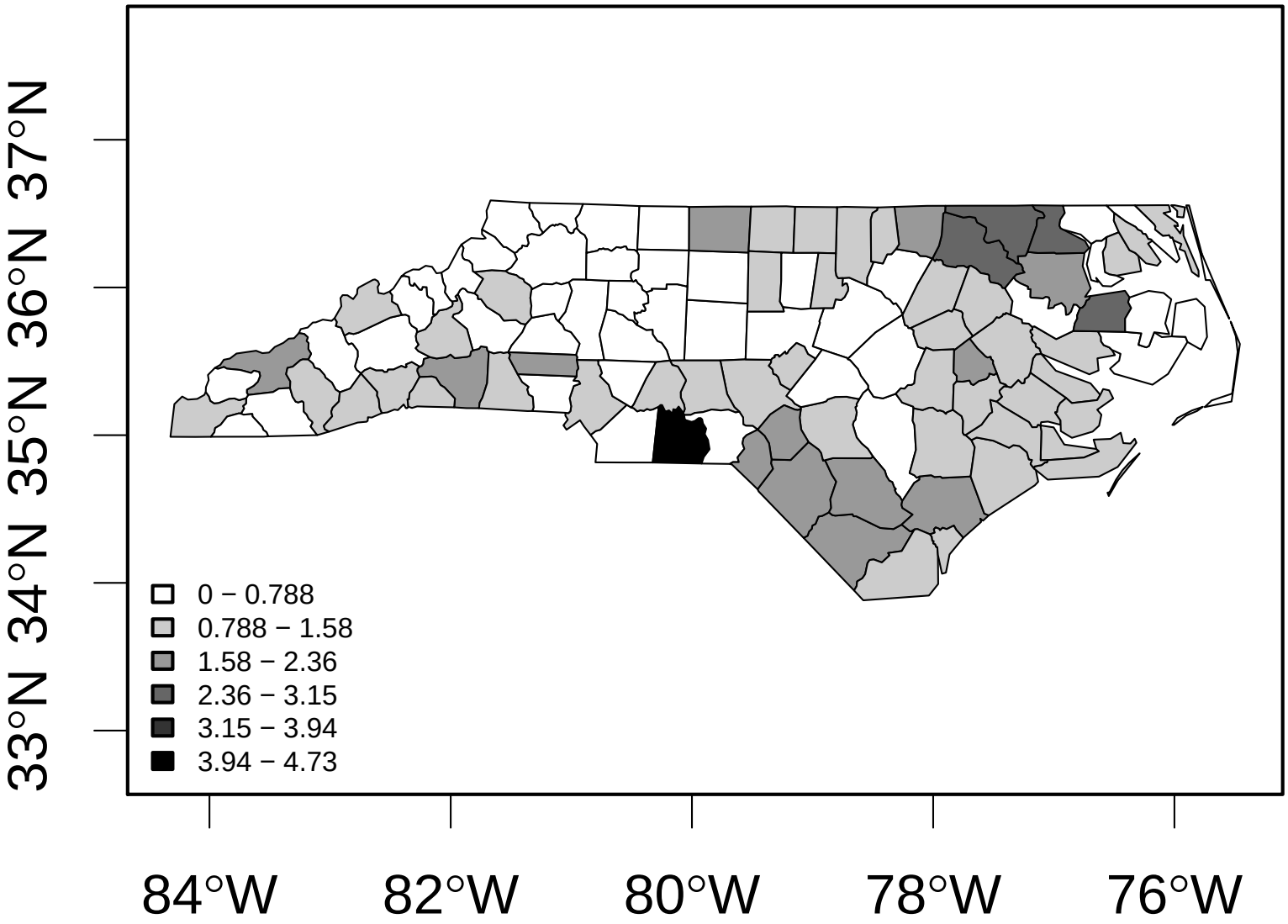


Figure 6: SMRs

Bayes cluster model

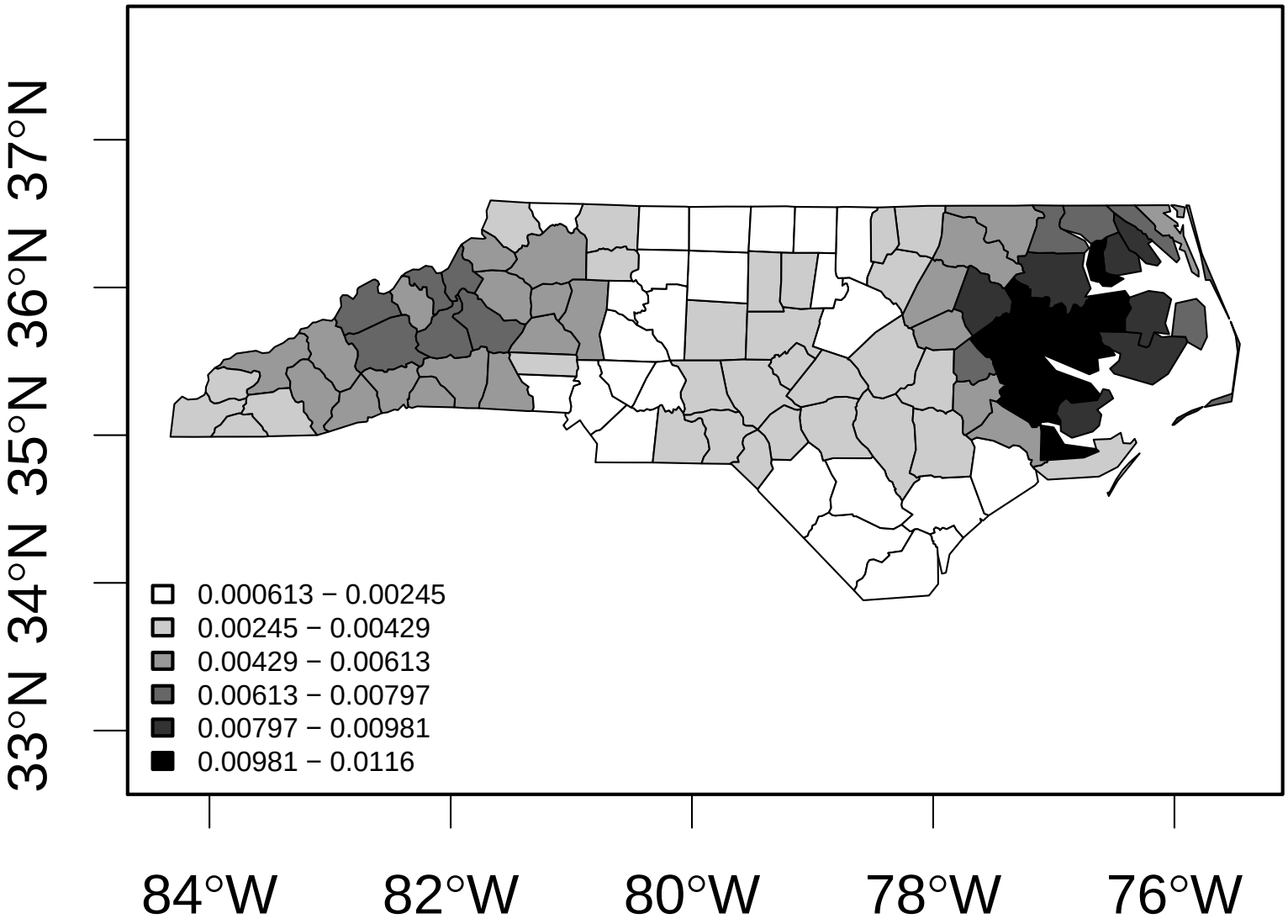


Figure 7: Prior probabilities of lying in a cluster

Bayes cluster model

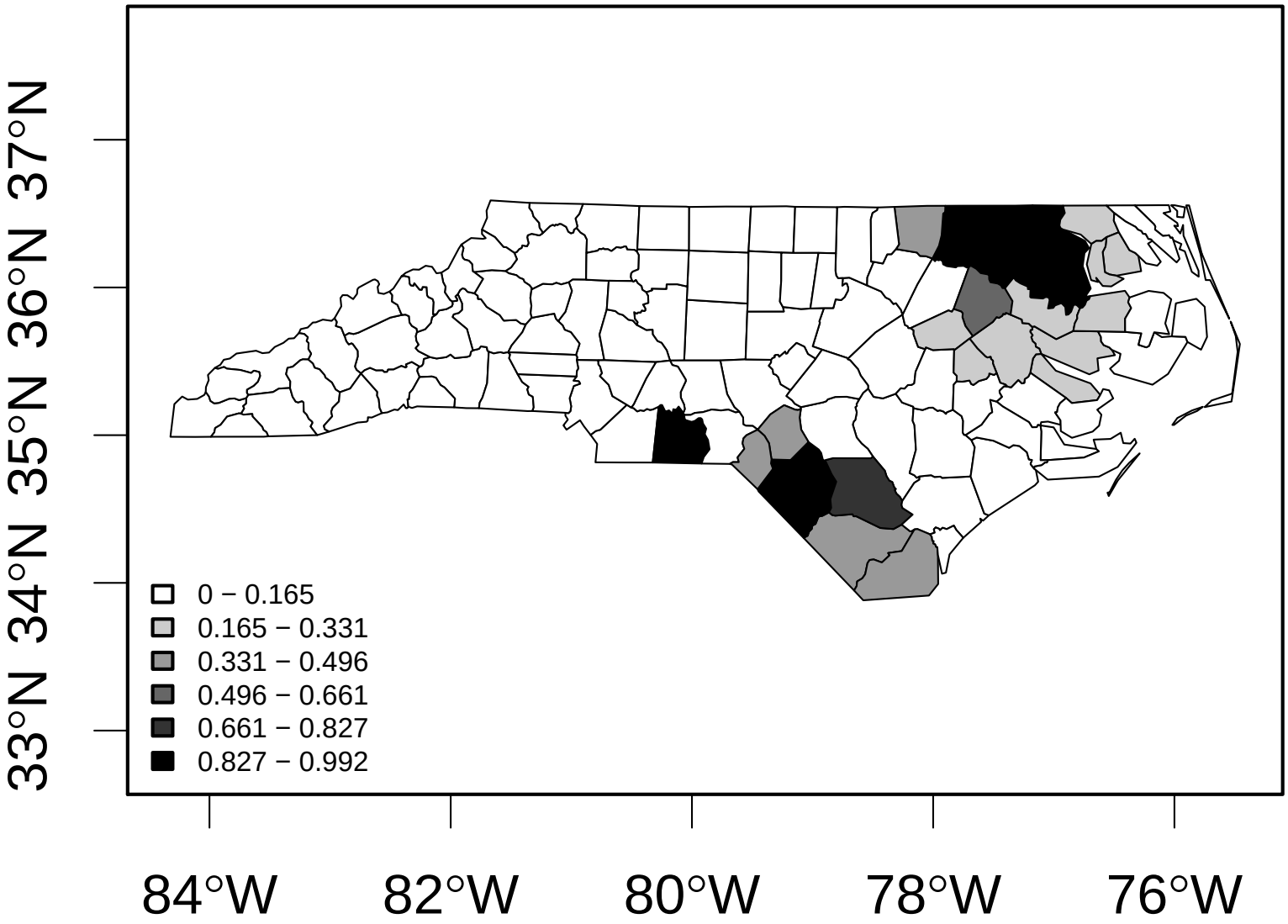


Figure 8: Posterior probability of a cluster

Bayes cluster model

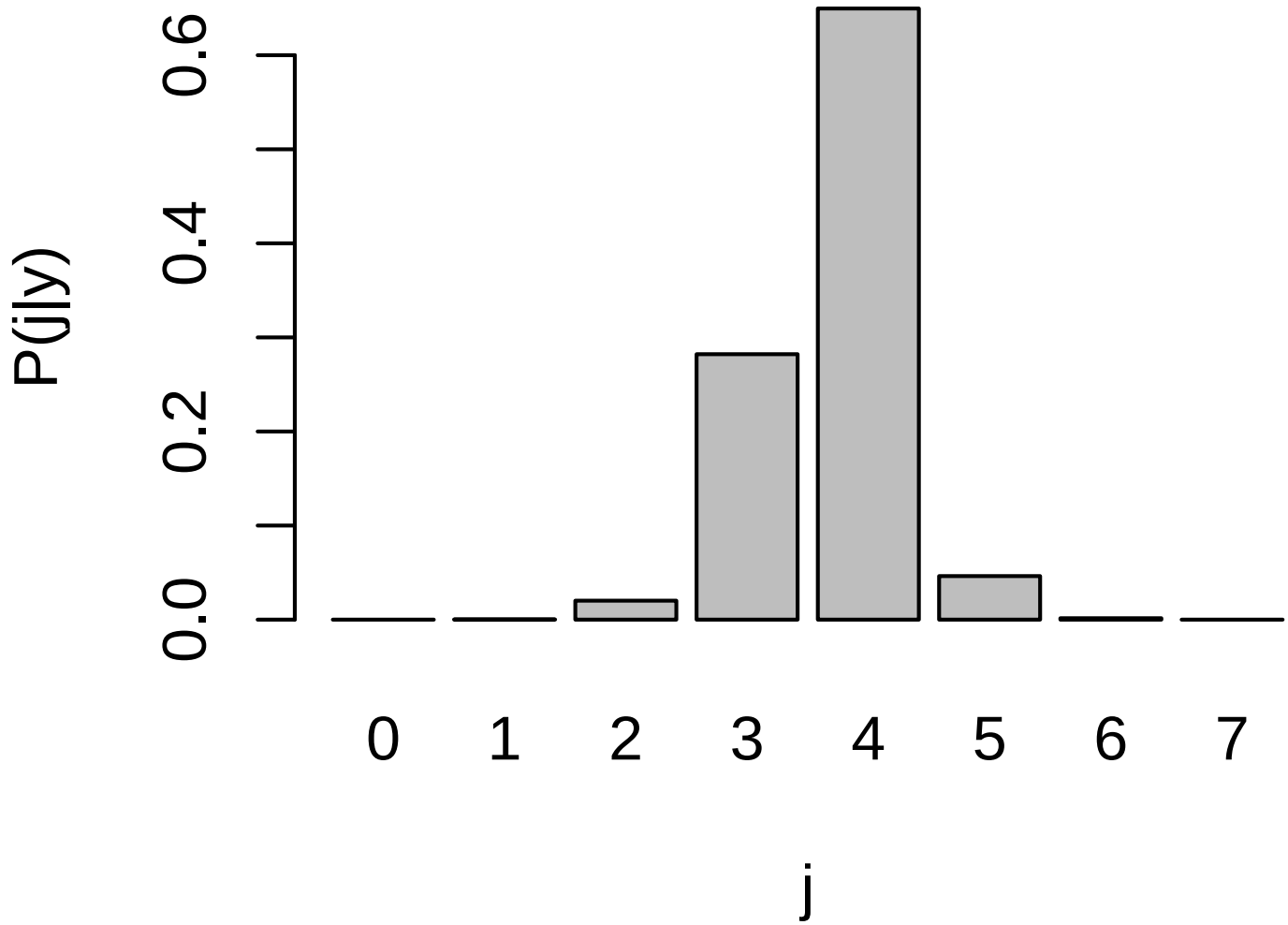


Figure 9: Posterior on the number of clusters

Bayes cluster model

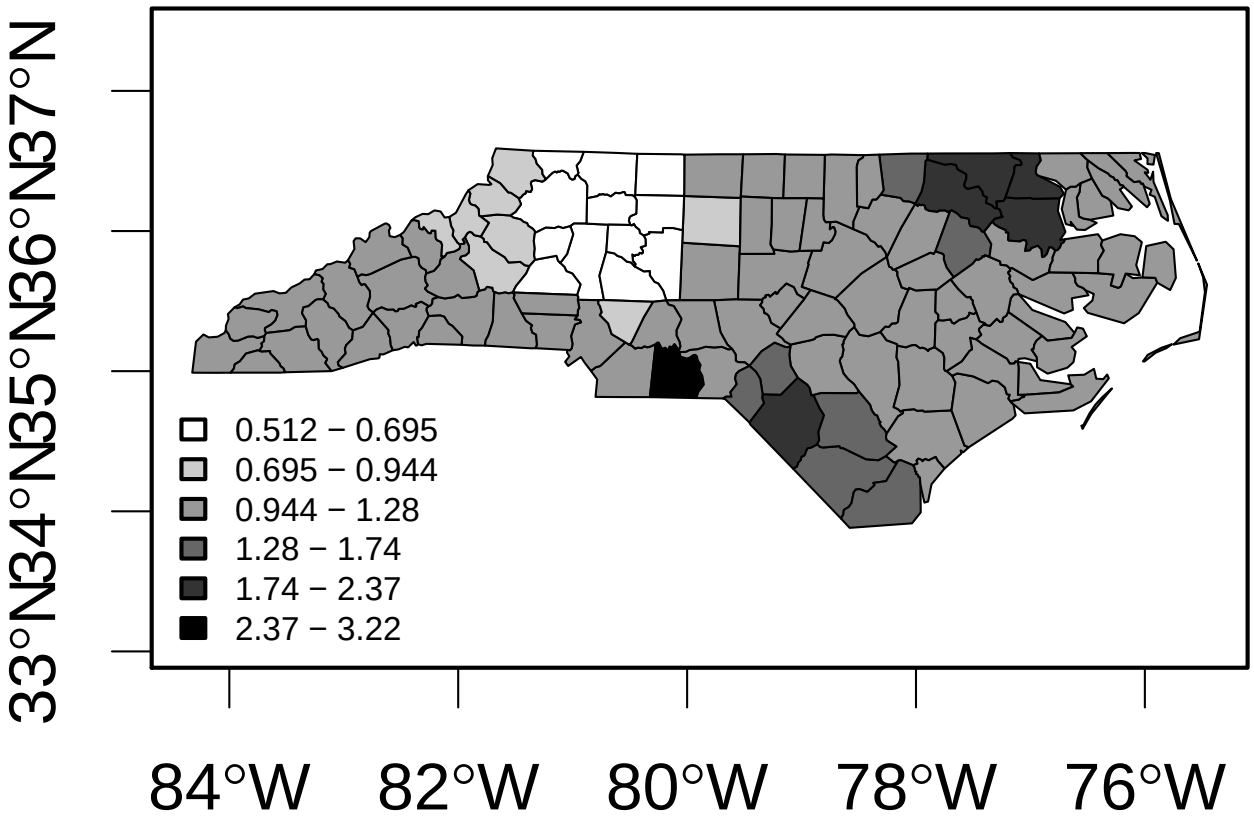


Figure 10: Posterior relative risk estimates