Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

DClusterm: Model-based Detection of Disease Clusters

Virgilio Gómez-Rubio
Universidad de Castilla-La Mancha

Paula Moraga
Queensland University of Technology

John Molitor
Oregon State University

Barry Rowlingson
Lancaster University

Abstract

The detection of regions with unusually high risk plays an important role in disease mapping and the analysis of public health data. In particular, the detection of groups of areas (i.e., clusters) where the risk is significantly high is often conducted by Public Health authorities.

Many methods have been proposed for the detection of these disease clusters, most of them based on moving windows, such as Kulldorff's Spatial Scan Statistics. Here we describe a model-based approach for the detection of disease clusters implemented in the **DClusterm** package. Our model-based approach is based on representing a large number of possible clusters by dummy variables and then fitting many generalized linear models to the data where these covariates are included one at a time. Cluster detection is done by performing a variable or model selection among all fitted models using different criteria.

Because of our model-based approach, cluster detection can be performed using different types of likelihoods and latent effects. We cover the detection of spatial and spatio-temporal clusters, as well as how to account for covariates, zero-inflated datasets and overdispersion in the data.

Keywords: disease cluster, spatial statistics, R.

1. Introduction

The analysis of epidemiological data at small area level often involves accounting for possible risk factors and other important covariates using different types of regression models. However, it is not uncommon that after a number of covariates have been accounted for, residuals

still show a spatial distribution that defines some groups of areas with unusual high epidemiological risk. Hence, in many occasions it is not clear whether all spatial risk factors have been included in our model.

Public health data are often aggregated over small administrative areas due to issues of confidentiality. However, individual data are often available as well, and Generalized Linear Models (GLM, McCullagh and Nelder 1989) is a common framework used in disease mapping for modeling aggregated and individual-level information. GLMs not only model Poisson or binomial responses, but they can also link the outcome to a linear predictor on the covariates (and, possibly, other effects). However, until recently, it was not clear how to use GLMs to detect clusters of disease, a group of contiguous areas with significant high risk.

In order to detect disease clusters, the most widely used method is probably the Spatial Scan Statistic (SSS) proposed by Kulldorff (1997). Given a possible cluster z, the SSS will compare the relative risk in the cluster θ_z to the relative risk outside the cluster $\theta_{\overline{z}}$ using the following test:

$$\begin{aligned} H_0 &: \quad \theta_z = \theta_{\overline{z}} \\ H_1 &: \quad \theta_z > \theta_{\overline{z}} \end{aligned} \tag{1}$$

This test is performed via the use of a likelihood ratio statistic, where many different possible clusters are tested in turn by changing the areas in z and the most likely cluster (i.e., the one with the highest value of the test statistic) is selected. Significance of this cluster is assessed with a Monte Carlo test that also provides the p-value.

Several R packages include an implementation of the SSS. **DCluster** (Gómez-Rubio, Ferrándiz-Ferragud, and López-Quílez 2005) implements the orginal version of the SSS for spatial binomial and Poisson data and can compute cluster significancy by means of bootstrap or a Gumbel distribution. Package **SpatialEpi** (Kim and Wakefield 2016) also implements the SSS for binomial or Poisson data and assesses significancy using a Monte Carlo test. **rsatscan** (Kleinman 2015) provides a set of functions to create the necessary data files and run the SaTScan$^{\text{TM}}$ software (for Windows) from R. Hence, spatio-temporal detection of disease clusters can also be performed. Package **scanstatistics** (Allévius 2018) implements a number of space-time scan statistics, that includes spatio-temporal versions of the SSS. Similarly as the original SSS, these implementations lack of a general way of adjusting for covariates and assessing the significancy of non-primary clusters.

Other packages which can be used for the detection of spatial clusters are described now. The **smerc** package (French 2015) implements different scan statistics, including some for the detection of non-circular clusters. Package **AMOEBA** (Valles 2014) includes a function to detect spatial clusters based on the Getis-Ord statistic. Bayesian Dirichlet processes for spatial clustering are implemented in the **PReMiuM** package (Liverani, Hastie, Azizi, Papathomas, and Richardson 2015). A version of the spatial scan statistic adapted to detect clusters in social networks is available in package **SNscan** (Wang, Hsu, and Phoa 2016). Package **SpatialEpiApp** (Moraga 2017) includes a Shiny application for the analysis of spatial and spatio-temporal disease mapping that includes cluster detection using the SSS. Methods for the detection of spatial clusters with case-control point data are avilable in package **smacpod** (French 2018). The **surveillance** package (Salmon, Schumacher, and Höhle 2016; Meyer, Held, and Höhle 2017) also implements several methods for the detection of spatial and spatio-temporal clustering. Finally, package **ClustGeo** (Chavent, Kuentz, Labenne, and Saracco

2017) implements hierarchical clustering with spatial constraints that can be used to partition regions into different groups of contiguous areas.

In this paper we describe a novel approach to disease cluster detection that provides a link between GLMs and SSS. We have implemented this approach via the free and open source **DClusterm** package for the R statistical software (R Core Team 2016). This approach involves fitting many different GLMs for which dummy variables that represent possible clusters are included one at a time. Cluster detection is based on selecting a number of dummy cluster variables using variable selection methods.

This paper is organized as follows. Section 2 will introduce the link between GLM and SSS. Next, in Section 3 we show how to include random effects in the detection of disease clusters to account for over-dispersion. The detection of disease clusters for zero-inflated data is discussed in Section 4. Section 5 describes how to extend these ideas to detect clusters in space and time. Finally, a discussion and some final remarks are provided in Section 6.

2. Generalized linear models for cluster detection

2.1. General description

An explicit link between GLM's and the SSS is provided by Jung (2009); Zhang and Lin (2009) who show that the test statistic for a given cluster is equivalent to fitting a GLM using a cluster variable as predictor. This cluster variable is a dummy variable which is 1 for the areas in the cluster and 0 for the areas outside the cluster. By including cluster covariates in the model we obtain an estimate of the increased risk (as measured by its associated coefficient) and its significance (by means of the associated p-value, for example).

We demonstrate our aproach by first considering a Poisson model with expected counts E_i and observed cases O_i modeled as:

$$\begin{aligned} O_i &\sim Po(E_i\theta_i) \\ \log(\mu_i) &= \log(E_i) + \alpha + \beta x_i \end{aligned} \tag{2}$$

Here μ_i is the mean of the Poisson distribution (which is equal to $E_i\theta_i$), $\log(E_i)$ denotes an offset to account for the population "exposure" (age, gender, etc.) and θ_i is the relative risk and it measures any deviation (increase or decrease) in the incidence of the disease from the expected number of cases. Finally, x_i represents a covariate of the outcome of interest.

After fitting this model, we obtain estimates $\hat{\alpha}$ and $\hat{\beta}$ which account for the effects of non-cluster covariates in the model. We include the cluster covariates as follows:

$$\log(\mu_i) = \log(E_i) + \hat{\alpha} + \hat{\beta}x_i + \gamma_j c_i^{(j)} \tag{3}$$

The overall intercept is now $\log(E_i) + \hat{\alpha} + \hat{\beta}x_i$ and $c_i^{(j)}$ denotes a dummy variable associated with cluster j, with j taking values from 1 to the number of clusters being tests, defined as

$$c_i^{(j)} = \begin{cases} 1 & \text{if area } i \text{ belongs to cluster } j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Here, γ_j is a measure of the risk within cluster j. We are only interested in clusters whose coefficient is significantly higher than 0 (i.e., increased risk). Hence those with a significant negative coefficient will be ignored. We use model selection techniques to select the most important cluster in the study region. As such, the log-likelihood can be used to compare the model with the cluster variables to the null model (i.e., the one with no cluster covariates at all).

It is possible to perform cluster detection without considering non-cluster based covariates in the model. Here, cluster detection accounting for the non-cluster based covariates will likely provide a different number of clusters than models fit without these variables. By comparing the clusters detected in both models (with and without non-cluster based covariates), we will be able to find which clusters are linked to underlying risk factors included in the model and which clusters remain unexplained by these other variables. In the examples that we include in this paper, we will consider both scenarios to better understand how cluster detection works.

Similar approaches to detection of disease clusters have been proposed by Bilancia and Demarinis (2014); Gómez-Rubio, Moraga, and Molitor (2017) who describe a similar approach to the detection of disease clusters using Bayesian hierarchical models. The Integrated Nested Laplace Approximation (INLA, Rue, Martino, and Chopin 2009) is used in both cases for model fitting as it provides computational benefits over other computationally expensive methods, such as Markov Chain Monte Carlo.

2.2. Leukemia in upstate New York

We first consider an example where we model counts of leukemia cases in upstate New York. These data are provided in the `NY8` dataset, available in package `DClusterm`. It provides cases of leukemia in different census tracts in upstate New York. This data set has been analyzed by several authors (Waller, Turnbull, Clark, and Nasca 1992; Waller and Gotway 2004). The location of leukemia is thought to be linked to the use of Trichloroethene (TCE) by several companies in the area. Figure 1 shows the Standardized Mortality Ratios of the census tracts and the locations of the industries using TCE.

In order to measure exposure, the inverse of the distance to the nearest TCE site has been used (variable `PEXPOSURE` in the dataset). In addition, two other socioeconomic covariates have been used: the percentage of people aged 65 or more (variable `PCTAGE65P`) and the percentage of people who own their home (variable `PCTOWNHOME`).

Hence, our first action is to load some required packages, such as **xts** (Ryan and Ulrich 2014), and the dataset itself.

```
R> library("DClusterm")
R> library("xts")
R> data("NY8")
```

A number of cases could not be linked to their actual location and they were distributed uniformly over the study area, making the counts real numbers instead of integers (see, Waller and Gotway 2004, for details). We have rounded these values as we intend to use a Poisson likelihood for the analysis. Furthermore, expected counts are computed using the overall incidence ratio (i.e., total number of cases divided by the total population). Age-sex standardization is not possible in this case as this information is not available in our dataset.

```
R> NY8$Observed <- round(NY8$Cases)
R> NY8$Expected  <- NY8$POP8 * sum(NY8$Observed) / sum(NY8$POP8)
R> NY8$SMR <- NY8$Observed / NY8$Expected
```

The centres of the areas will be stored in columns named `x` and `y`. This will be used later when ordering the areas by increasing distance to the putative cluster centre. If the location of the main populated cities are available these could be used but here we will use function `coordinates()` instead:

```
R> NY8$x <- coordinates(NY8)[, 1]
R> NY8$y <- coordinates(NY8)[, 2]
```

As **DClusterm** is designed to detect clusters in space and time, it will always expect data to be from one of the classes in packages **sp**, for purely spatial data, or **spacetime** (Pebesma 2012), for spatio-temporal data.

2.3. Cluster detection

Cluster detection with no covariates

First of all, a model with no covariates will be fitted and used as a baseline, so that other models can be compared to this one (for example, using the AIC or the log–likelihood) to assess whether they provide a better fit.

```
R> ny.m0 <- glm(Observed ~ offset(log(Expected)) + 1, family = "poisson",
+    data = NY8)
```

Function `DetectClustersModel()` will take this baseline model (using argument `model0`), create the cluster dummy variables and test them in turn. Then, those clusters with a highest significance will be reported.

Argument `thegrid` will take a 2-column `data.frame` (with names `x` and y) with the centres of possible clusters. If the grid of cluster centres is not defined, then a rectangular grid is used with a distance between adjacent points defined by argument `step`. Dummy cluster variables are created around these points by adding areas to the cluster until a certain percentage of the population (defined by argument `fractpop`), with the column to be used to compute the population at risk in the cluster defined by parameter `ClusterSizeContribution`, or until a certain distance about the centre (defined by argument `radius`) has been reached. When testing for significant cluster variables, argument `alpha` defines the significance level.

`DetectClustersModel()` can detect spatial and spatio-temporal clusters, which is why its first argument is a space-time object. The type of clusters that are investigated is defined by argument `typeCluster`. In the example we have used `typeCluster = "S"`.

The output of `DetectClustersModel()` returns all clusters found at significance level given by the parameter `alpha`. In the detection process, multiple tests are performed simultaneously which can increase the likelihood of incorrectly declaring a group of areas as a cluster. This situation is known as multiple testing problem and to avoid it several statistical methods have been developed. Here, we deal with this problem using the Bonferroni correction which uses

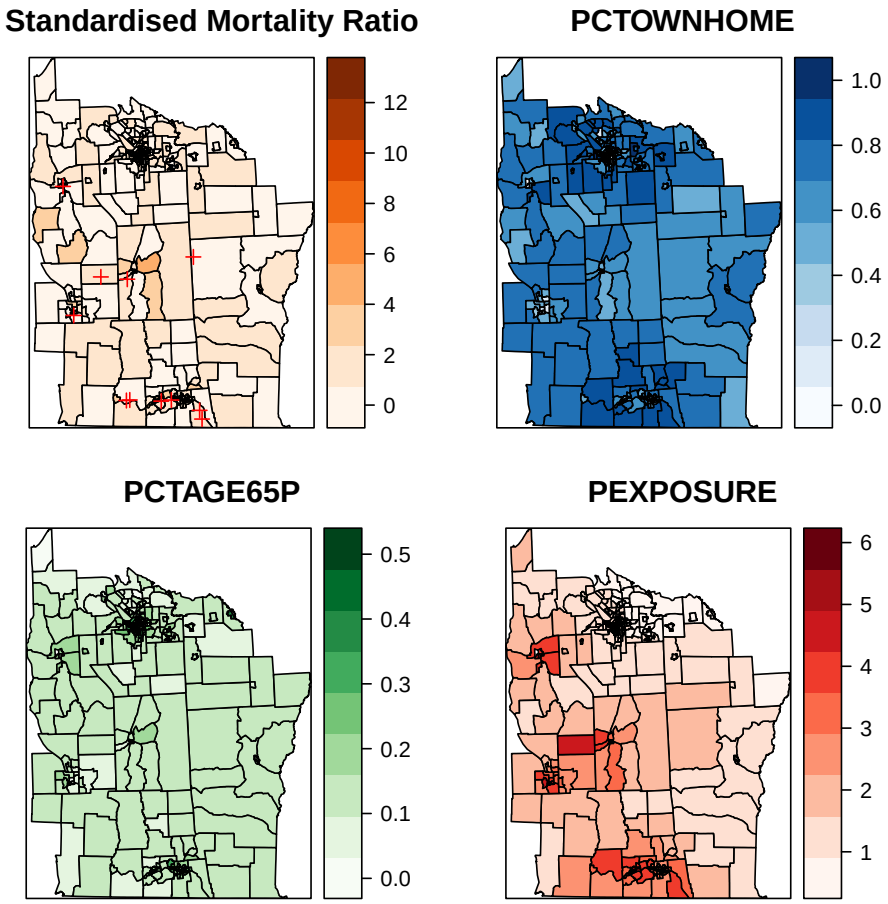

Figure 1: Standardised Mortality Ratios and covariates of the incidence of Leukemia in upstate New York dataset. `PEXPOSURE` is the inverse of the distance to the nearest TCE site, `PCTAGE65P` is the percentage people aged 65 or more, and `PCTOWNHOME` is the percentage of people who own their home. The red crosses in the top-left map mark the locations of the TCE sites.

a stricter significance level to declare a cluster (Dunn 1958). Specifically, a cluster is declared if its p-value is smaller than alpha divided by the number of clusters tested.

The output of `DetectClustersModel()` has a column called `alpha_bonferroni` that denotes the level of significance adjusted for multiple testing using the Bonferroni correction. Thus, users can avoid the multiple testing problem by filtering the output and keeping only the clusters with p-value less than `alpha_bonferroni`.

Other options include the number of replicates for Monte Carlo tests (argument `R`) if cluster assessment is done by simulation. By default, Monte Carlo tests are not used. **DClusterm** allows for parallel computing using several cores as implemented in package **parallel**. The number of cores to use is defined in option `mc.cores` (now 4 cores are used):

```
R> options(mc.cores = 4)
```

In the following example, to reduce the computational burden, we have only looked for clusters

around 5 areas (whose rows in NY8 are defined in variable idxcl). In a real application we advise the use of all locations (area centroids or actual locations of individual data).

```
R> idxcl <- c(120, 12, 89, 139, 146)
R> ny.cl0 <- DetectClustersModel(NY8,
+    thegrid = as.data.frame(NY8)[idxcl, c("x", "y")],
+    fractpop = 0.15, alpha = 0.05, radius = Inf, step = NULL,
+    typeCluster = "S", R = NULL, model0 = ny.m0,
+    ClusterSizeContribution = "POP8")
```

Below is a summary of the clusters detected. Dates can be ignored as this is a purely spatial cluster analysis. In the case of spatio-temporal clusters, dates shown define the temporal range of the cluster. Values x and y defined the cluster centre, size is the number of areas included in the cluster, statistic is the value of the test statistic and risk represents the point estimate of the associated cluster coefficient. Also, note that only clusters with a lower pvalue than argument alpha are returned. cluster indicates whether the cluster is a significant one. Finally, note how detected clusters are ordered by increasing value of pvalue, so that the most significant clusters are reported first.

```
R> ny.cl0
```

```
           x       y size statistic       pvalue      risk cluster
11  424728.9 4661404   39  8.044846 0.0000604120 0.3916904    TRUE
88  409430.4 4720092    9  6.967107 0.0001893208 0.6455613    TRUE
119 404710.7 4768346   24  3.254824 0.0107290781 0.4445236    TRUE
    alpha_bonferroni
11              0.01
88              0.01
119             0.01
```

Function get.knclusters() can be used to extract the indices of the areas in the different clusters detected. Similarly, a vector with the cluster to which area belongs can be obtained with function get.allknclusters(). The detected clusters plus a 'no cluster' category are the levels of this categorical variable, which can be added to a spatial object as follows:

```
R> NY8$CLUSTER0 <- get.allknclusters(NY8, ny.cl0)
```

The areas and centers of the clusters detected are shown in Figure 2. Because of the lack of adjustment for covariates these clusters show regions of high risk based on the raw data (observed and expected counts) alone.

Cluster detection after adjusting for covariates

Similarly, clusters can be detected after adjusting for significant risk factors. First of all, we will fit a Poisson regression with the 3 covariates mentioned earlier. As it can be seen, two of them are clearly significant and the third one has a p-value very close to 0.05:

```
R> ny.m1 <- glm(Observed ~ offset(log(Expected)) + PCTOWNHOME + PCTAGE65P +
+    PEXPOSURE, family = "poisson", data = NY8)
R> summary(ny.m1)

Call:
glm(formula = Observed ~ offset(log(Expected)) + PCTOWNHOME +
    PCTAGE65P + PEXPOSURE, family = "poisson", data = NY8)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9099  -1.1294  -0.1768   0.6385   3.2426

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.65507    0.18550  -3.531 0.000413 ***
PCTOWNHOME  -0.36472    0.19316  -1.888 0.058998 .
PCTAGE65P    4.05031    0.60559   6.688 2.26e-11 ***
PEXPOSURE    0.15141    0.03165   4.784 1.72e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 459.05  on 280  degrees of freedom
Residual deviance: 384.01  on 277  degrees of freedom
AIC: 958.97

Number of Fisher Scoring iterations: 5
```

As we have three covariates in the model, the offset included in the model will be different now, which may produce that the detected clusters may be different in this case. Cluster detection is performed as in the previous example, but now we use the model that adjusts for covariates instead:

```
R> ny.cl1 <- DetectClustersModel(NY8,
+    thegrid = as.data.frame(NY8)[idxcl, c("x", "y")], fractpop = 0.15,
+    alpha = 0.05, typeCluster = "S", R = NULL, model0 = ny.m1,
+    ClusterSizeContribution = "POP8")

R> ny.cl1

          x       y size statistic       pvalue      risk cluster
88  409430.4 4720092    9  5.861204 0.0006175202 0.5869176    TRUE
119 404710.7 4768346   20  3.160591 0.0119304026 0.4882633    TRUE
    alpha_bonferroni
88              0.01
119             0.01
```

Similarly as in the previous example, we can use function `get.allkncluster()` to get the indices of the areas in the clusters and add a new variable to the `SpatialPolygonsDataFrame`:

```
R> NY8$CLUSTER1 <- get.allknclusters(NY8, ny.cl1)
```

Figure 2 shows the clusters detected after adjusting for covariates. Compared to the example with no covariate adjustment, the most significant cluster has disappeared. Hence, that cluster has been explained by the effect of the covariates. Another cluster is a bit smaller in size, which means that covariates only explain a small part of it. In all remaining clusters, cluster significance has been reduced by the effect of the covariates.

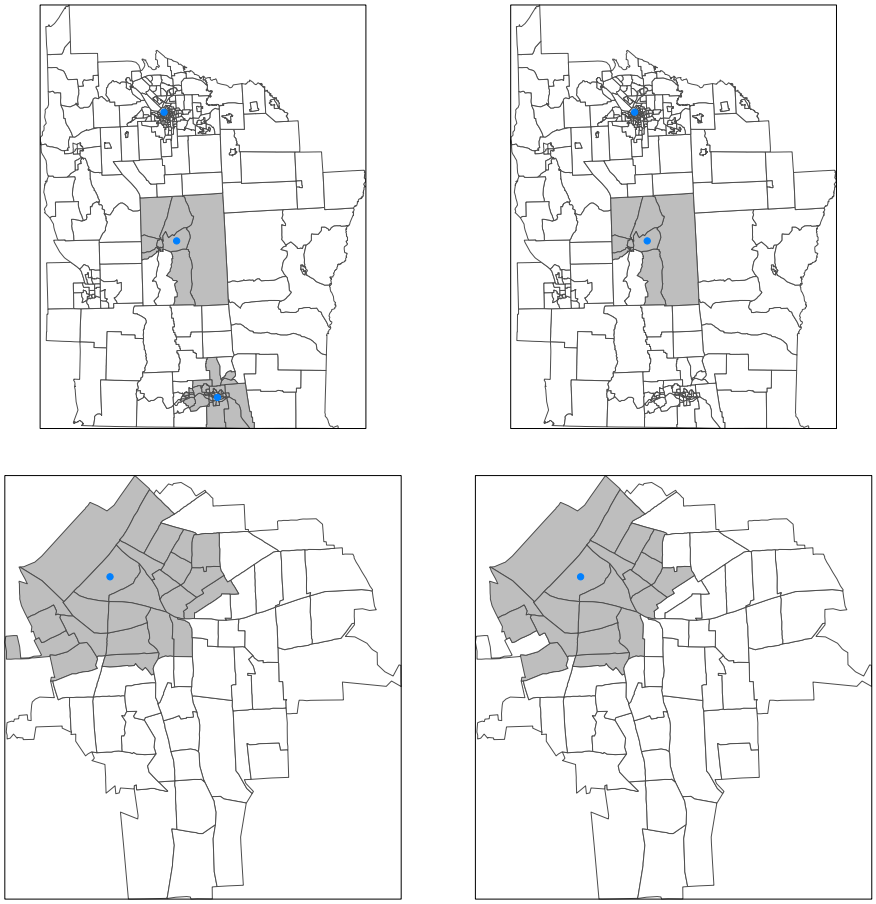

Figure 2: Clusters detected with no covariate adjustment (left) and after adjusting for covariates (right) in the whole study region (top row) and Syracuse city (bottom row). Areas in clusters are shaded in gray and cluster centres are represented by blue dots.

Finally, function `slimknclusters` could have been used to remove overlapping clusters with less significant coefficients. This procedure will sort all detected clusters in decreasing order according to the value of `statistic` and remove those clusters which overlap with another cluster with a larger value of `statistics`. This will be similar in spirit to the SSS as this will only consider a single cluster around each area and will help to control for the multiple testing problem inherent to the problem of the detection of disease clusters.

Function `slimknclusters` takes three arguments. The first one is the spatial or spatio-temporal object used in the call to `DetectClustersModel`, the second one is the table with the clusters detected and the third one is the minimum cluster size to consider. By default, this is 1, and it will set the minimum number of areas in a cluster to report it.

For example, the following code could be used to reduce the number of possible clusters to those non-overlapping cluster with a size of at least three in the cluster detection with covariates:

```
R> slimknclusters(NY8, ny.cl1, 3)
```

```
           x       y size statistic       pvalue      risk cluster
88  409430.4 4720092    9  5.861204 0.0006175202 0.5869176    TRUE
119 404710.7 4768346   20  3.160591 0.0119304026 0.4882633    TRUE
    alpha_bonferroni
88              0.01
119             0.01
```

In this case, as there are no overlapping clusters the results do not change.

3. Mixed-effects models for cluster detection

Random effects can be incorporated into our models to account for unmeasured risk factors. In this context, random effects will model area-specific intrinsic variation not explained by other covariates or cluster variables in the model. Cluster detection will be performed as usual, but we should keep in mind that by including random effects and dummy cluster variables there may be a clash between the two. By using dummy variables we are intentionally looking for unexplained spatial variation in the data. Hence, random effects should aim at modelling a different structure in the data. For example, if spatial random effects are included, these may tend to model the clusters and then an identifiability problem between them and the dummy cluster covariates may occur, which may lead to poor estimates of the random effects and/or the coefficients of the dummy cluster covariates.

For the Poisson case, this will mean that the relative risk can be modelled as:

$$\begin{aligned} \log(\mu_i) &= \log(E_i) + \alpha + \beta x_i + \gamma_j c_i^{(j)} + u_i \\ u_i &\sim N(0, \sigma_u^2) \end{aligned} \tag{5}$$

where u_i represents a random effect normally distributed with zero mean and variance σ_u^2. Note that random effects can be defined to be spatially correlated, as suggested by Bilancia and Demarinis (2014). However, this can produce a clash between the dummy cluster variables and the random effects.

Random effects are particularly useful to model over-dispersion in count data, which occurs when the variance in the data is greater than the variance defined by the statistical model. For example, for a Poisson model the mean and variance should be equal. Hence, by including random effects in a Poisson model the variance of the data can be different to the mean.

3.1. Leukemia in upstate New York

We go back to the example on the leukemia incidence in upstate New York to show how models can include random effects and, at the same time, detect disease clusters. In this particular example, random effects will be important in order to reflect any over-dispersion present in the data. For this reason, our first step here is to test the data for over-dispersion using Dean's P_B and P'_B score tests (see, Dean 1992, for details). These two tests have been implemented in functions `DeanB()` and `DeanB2()` in the **DCluster** (Gómez-Rubio *et al.* 2005) package. They both take a `glm` object and perform the score tests:

```
R> DeanB(ny.m0)

        Dean's P_B test for overdispersion

data:  ny.m0
P_B = 5.5755, p-value = 1.234e-08
alternative hypothesis: greater

R> DeanB2(ny.m0)

        Dean's P'_B test for overdispersion

data:  ny.m0
P'_B = 5.6233, p-value = 9.368e-09
alternative hypothesis: greater
```

From the results, it is clear that when no covariates are included data are clearly over-dispersed. Hence, a Poisson distribution will not be appropriate to model the observed counts in each tract.

The same tests applied to the model with covariates produce a similar result:

```
R> DeanB(ny.m1)

        Dean's P_B test for overdispersion

data:  ny.m1
P_B = 2.0145, p-value = 0.02198
alternative hypothesis: greater

R> DeanB2(ny.m1)

        Dean's P'_B test for overdispersion

data:  ny.m1
P'_B = 2.2391, p-value = 0.01257
alternative hypothesis: greater
```

Although p-values have increased, they are both small and we may still consider that data are over-dispersed. Hence, we will aim at detecting clusters using a Poisson regression with independent random effects to account for census tract-level heterogeneity.

3.2. Cluster detection

Cluster detection with no covariates

The baseline model with no covariate will now be fitted with function `glmer()` from package **lme4** (Bates, Mächler, Bolker, and Walker 2015). This function is similar to `glm()` for GLMs but it will allow us to include random effects in the model as part of the `formula` argument.

```
R> library("lme4")
R> ny.mm0 <- glmer(Observed ~ offset(log(Expected)) + (1 | AREANAME),
+    data = as(NY8, "data.frame"), family = "poisson")

R> ny.clmm0 <- DetectClustersModel(NY8,
+    thegrid = as.data.frame(NY8)[idxcl, c("x", "y")], fractpop = 0.15,
+    alpha = 0.05, typeCluster = "S", R = NULL, model0 = ny.mm0,
+    ClusterSizeContribution = "POP8")

R> ny.clmm0

          x       y size statistic       pvalue      risk cluster
88  409430.4 4720092    9  7.577766 9.900776e-05 0.7880057    TRUE
119 404710.7 4768346   24  2.415959 2.793753e-02 0.7862091    TRUE
    alpha_bonferroni
88              0.01
119             0.01
```

After accounting for overdispersion, the number of clusters detected is 2. These are shown in Figure 3. The largest cluster detected before has now disappeared and only the two smallest clusters remain. This may be due to the fact that the first cluster has the smallest SMR and risk. Hence, allowing for extra-variation will make the discrepancy between observed and expected less extreme and this cluster will be the first one to be declared as non-significant. The two remaining clusters have the same size and a very similar risk than in the previous case.

Cluster detection with covariates

The mixed-effects model with covariates can be fitted as follows:

```
R> ny.mm1 <- glmer(Observed ~ offset(log(Expected)) + PCTOWNHOME +
+    PCTAGE65P + PEXPOSURE + (1 | AREANAME),
+    data = as(NY8, "data.frame"), family = "poisson")
R> #summary(ny.mm1)
```

The estimates of the coefficients are similar to that of the fixed-effects model, but the associated p-values are somewhat higher. This is probably to the inclusion of random effects in the model.

```
R> ny.clmm1 <- DetectClustersModel(NY8,
+    thegrid = as.data.frame(NY8)[idxcl, c("x", "y")], fractpop = 0.15,
+    alpha = 0.05, typeCluster = "S", R = NULL, model0 = ny.mm1,
+    ClusterSizeContribution = "POP8")

R> ny.clmm1

          x       y size statistic      pvalue       risk cluster
88 409430.4 4720092    1  4.051264 0.004420356 0.03289885    TRUE
   alpha_bonferroni
88             0.01
```

When overdispersion and covariates are included in the model, only one of the clusters remains with size 1 and a slightly reduced estimate of the cluster coefficient. This should not come as a surprise given that we have already seen how including covariates explains some of the extra-variation and that by including random effects the significance of the clusters is also reduced. A summary of the clusters detected can be found in Figure 3.

As in the example in Section 2.2, we have created two cluster variables to display the clusters obtained:

```
R> NY8$CLUSTERMM0 <- get.allknclusters(NY8, ny.clmm0)
R> NY8$CLUSTERMM1 <- get.allknclusters(NY8, ny.clmm1)
```

4. Zero-inflated models for cluster detection

The analysis of rare diseases often involves datasets where there are many areas with zero counts, leading to zero-inflated data. In this situation the Poisson or binomial likelihoods may not be suitable to fit a model, because there is an excess of areas with zeros with respect to the number predicted by the pure Poisson or binomial model, and other distributions for the data should be used. Gómez-Rubio and López-Quílez (2010) discuss this issue and they have extended model-based cluster detection methods to account for zero-inflation.

For count data, a zero-inflated Poisson model could be used. In this case, observed number of cases come from a mixture distribution of a Poisson and a distribution with all mass at zero. Probabilities are as follows:

$$Pr(O_i = n_i) = \begin{cases} \pi_i + (1 - \pi_i)Po(0|\theta_i E_i) & n_i = 0 \\ (1 - \pi_i)Po(n_i|\theta_i E_i) & n_i = 1, 2, \ldots \end{cases} \tag{6}$$

Here $Po(O_i|\theta_i E_i)$ represents the probability of observing O_i cases using a Poisson distribution with mean $\theta_i E_i$. π_i represents the proportion of zeroes observed that do not come from the Poisson distribution.

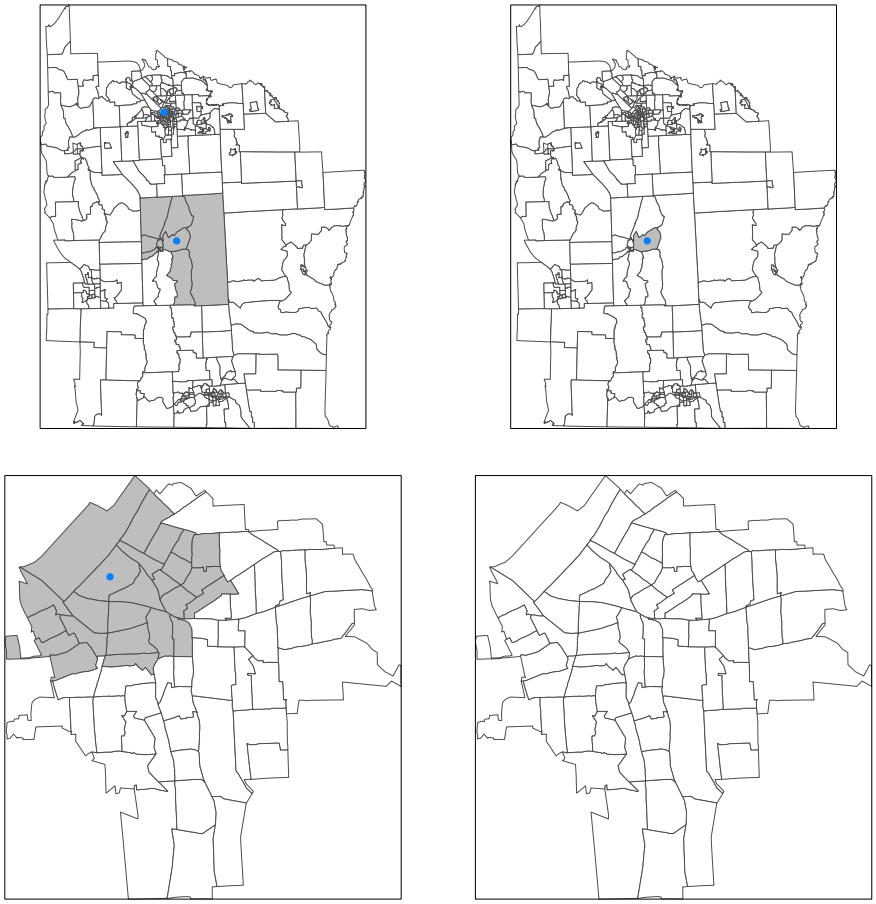

Figure 3: Clusters detected with no covariate adjustment (left) and after adjusting for covariates (right) in the whole study region (top row) and Syracuse city (bottom row) using a mixed-effects model to account for overdispersion of the data. Areas in clusters are shaded in gray and cluster centers are represented by blue dots.

Relative risks θ_i can be modeled using a log-linear model to depend on some relevant risk factors. Also, it is common that all π_i's are taken equal to a single value π.

4.1. Brain cancer in Navarre (Spain)

Ugarte, Ibáñez, and Militino (2006) analyze the incidence of brain cancer in Navarre (Spain) at the health district level. Figure 4 shows the Standardized Mortality Ratios. As it can be seen there are many areas where the SMR is zero because there are no cases in those areas. Ugarte, Ibáñez, and Militino (2004) have also reported a significant zero-inflation of these data compared to a Poisson distribution. For cluster detection, the method implemented in `DClusterm` is similar to the one used in Gómez-Rubio and López-Quílez (2010) for the detection of disease clusters of rare diseases.

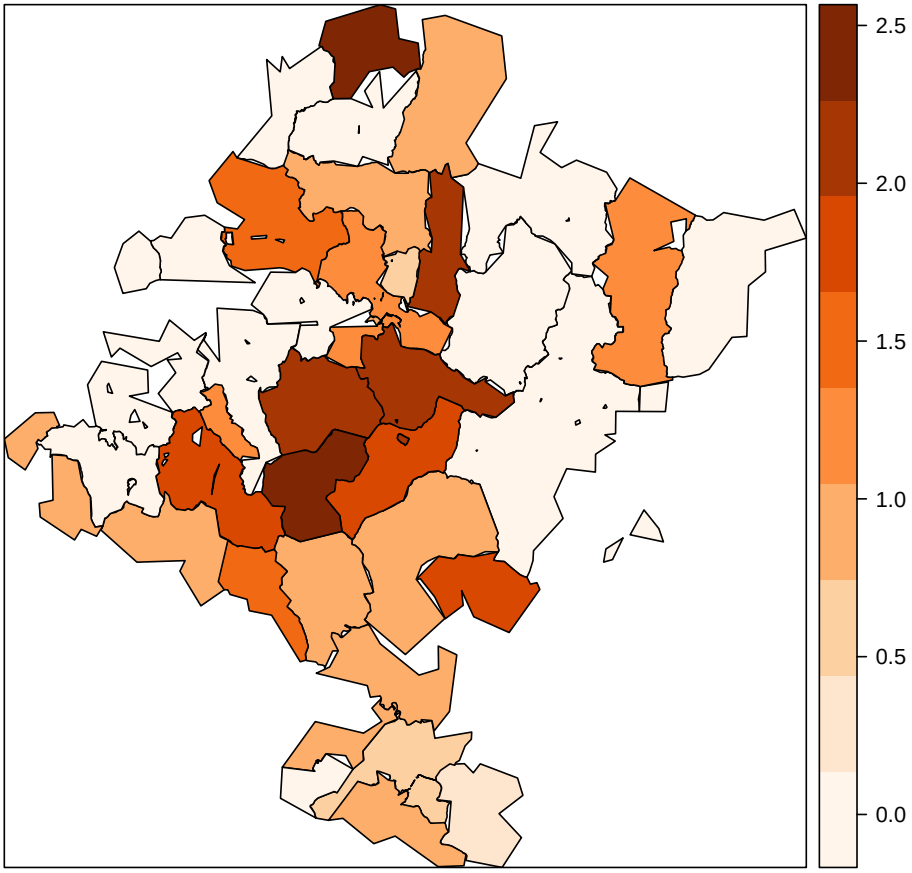

Figure 4: SMR of brain cancer in Navarre (Spain).

4.2. Cluster detection

Before starting our cluster detection methods, we will check the appropriateness of a Poisson distribution for this data. Fitting a log-linear model (with no covariates) gives the following model:

```
R> nav.m0 <- glm(OBSERVED ~ offset(log(EXPECTED)) + 1, family = "poisson",
+    data = brainnav)
```

Furthermore, a quasipoisson model has been fitted in order to assess any extra-variation in the data:

```
R> nav.m0q <- glm(OBSERVED ~ offset(log(EXPECTED)) + 1, data = brainnav,
+    family = "quasipoisson")
```

The dispersion parameter in the previous model is 1, which is higher than 1. This may mean that the Poisson distribution is not appropriate in this case due to the excess of zeros.

For this reason, and following Ugarte *et al.* (2004), a zero-inflated Poisson model has been fitted using function `zeroinfl()` from package **pscl** (Zeileis, Kleiber, and Jackman 2008). Here is the resulting model:

```
R> library("pscl")
R> nav.m0zip <- zeroinfl(OBSERVED ~ offset(log(EXPECTED)) + 1 | 1,
+    data = brainnav, dist = "poisson", x = TRUE)
R> summary(nav.m0zip)

Call:
zeroinfl(formula = OBSERVED ~ offset(log(EXPECTED)) + 1 | 1,
    data = brainnav, dist = "poisson", x = TRUE)

Pearson residuals:
    Min      1Q  Median      3Q     Max
-1.3585 -0.9137 -0.1378  0.7137  1.8091

Count model coefficients (poisson with log link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.09347    0.09459   0.988    0.323

Zero-inflation model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.6158     0.6435  -2.511    0.012 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in BFGS optimization: 9
Log-likelihood: -69.08 on 2 Df
```

Hence, the zero-inflated Poisson model will be used now to detect clusters of disease. As in the example on the New York leukemia dataset, a **spacetime** object will store all the information. The column for the expected counts must be named `Expected`, and this is our first step. Note also that, because only one time period is considered, data will have a single value and it is the 1st of January of 1990.

```
R> brainnav$Expected <- brainnav$EXPECTED
```

Function `DetectClustersModel()` will perform the cluster detection using a `zeroinfl` model. This provides a very flexible way of handling different types of models in R for cluster detection.

```
R> nav.cl0 <- DetectClustersModel(brainnav, coordinates(brainnav),
+    fractpop = 0.25, alpha = 0.05, typeCluster = "S", R = NULL,
+    model0 = nav.m0zip, ClusterSizeContribution = "EXPECTED")
```

The output will show the following clusters:

```
R> nav.cl0

          x       y size statistic     pvalue      risk cluster
31 596886.8 4710520    4  2.520092 0.02476587 0.5987255    TRUE
30 611795.5 4713762    3  2.016942 0.04459518 0.6139100    TRUE
   alpha_bonferroni
31          0.00125
30          0.00125
```

As it can be seen, two clusters (with a p-value lower than 0.05) are detected. However, they overlap and we will just consider the one with the lowest p-value, which is shown in Figure 5.

An index for the areas in each of the detected clusters can be obtained with function `knbinary()`. This function will return a `data.frame` with all the dummy cluster variables, i.e., the returned `data.frame` will have as many columns as clusters and a number of rows equal to the number of areas. Entries will be 1 if the corresponding areas are in a cluster and 0 otherwise. These indices can be used for a number of analyses, such as checking whether two clusters overlap or computing the number of times an area is included in a cluster. In the following example we obtain the representation of all the clusters detected and the first one, the most significant, is added as a new column to the original `SpatialPolygonsDataFrame` to be displayed in Figure 5.

```
R> nav.clusters <- get.knclusters(brainnav, nav.cl0)
R> brainnav$CLUSTER <- ""
R> brainnav$CLUSTER [ nav.clusters[[1]] ] <- "CLUSTER"
R> brainnav$CLUSTER <- as.factor(brainnav$CLUSTER)
```

5. Spatio-temporal clusters

Jung (2009) discusses how to extend model-based approaches for the detection of spatial disease clusters to space and time. Gómez-Rubio *et al.* (2017) propose the following model:

$$\log(\mu_{i,t}) = \log(E_{i,t}) + \gamma_j c_{i,t}^{(j)} \tag{7}$$

where $\mu_{i,t}$ is the mean of area i at time t and $c_{i,t}^{(j)}$ a cluster dummy variable for spatio-temporal cluster j. Random effects can also be included in Equation 7 as described in Section 3 and zero-inflated distributions can also be considered as in Section 4.

Note how now data are indexed according to space and time. Dummy cluster variables are defined as in the spatial case, by considering areas in the cluster according to their distance to the cluster center, for data within a particular time period. When defining a temporal cluster, areas are aggregated using all possible temporal windows up to a predefined temporal range.

5.1. Brain cancer in New Mexico

The `brainNM` dataset (included in **DClusterm**) contains yearly cases of brain cancer in New Mexico from 1973 to 1991 (inclusive) in a **spacetime** object. The data set has been taken from

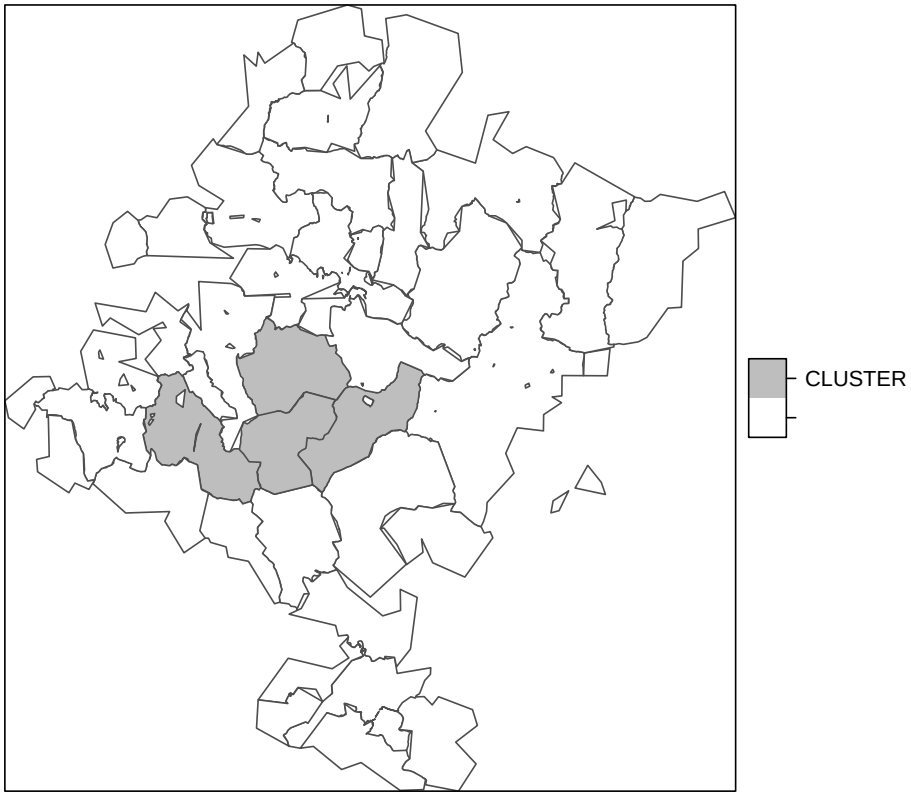

Figure 5: Cluster of brain cancer detected in Navarre (Spain).

the SaTScan website and the area boundaries from the U.S. Census Bureau. In addition, the location of Los Alamos National Laboratory (LANL) has been included (from Wikipedia). Inverse distance to this site can be used to test for increased risk in the areas around the Laboratory as no other covariates are available.

```
R> data("brainNM")
```

Expected counts have been obtained using age and sex standardization over the whole period of time. Hence, yearly differences are likely to be seen when plotting the data. Standardized Mortality Ratios have been plotted in Figure 6.

5.2. Cluster detection

Cluster detection with no covariates

Similarly as in the purely spatial case, a Poisson model with no covariates will be fitted first:

```
R> nm.m0 <- glm(Observed ~ offset(log(Expected)) + 1, family = "poisson",
```

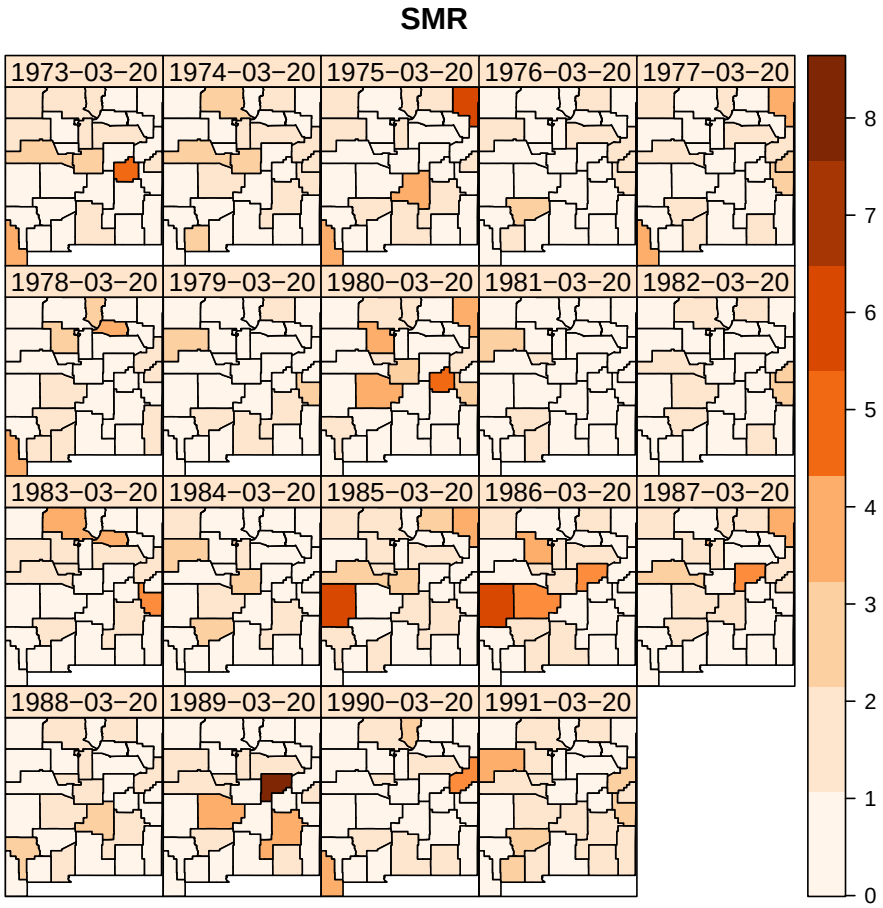

Figure 6: Standardized Mortality Ratios of brain cancer in New Mexico.

```
+    data = brainst)
R> summary(nm.m0)

Call:
glm(formula = Observed ~ offset(log(Expected)) + 1, family = "poisson",
    data = brainst)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4874  -0.9998  -0.4339   0.3773   3.1321

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 2.834e-16  2.917e-02       0        1

(Dispersion parameter for poisson family taken to be 1)
```

```
    Null deviance: 631.64  on 607  degrees of freedom
Residual deviance: 631.64  on 607  degrees of freedom
AIC: 1585.6

Number of Fisher Scoring iterations: 5
```

Before proceeding to disease cluster detection, we have extracted the centroids of the counties in New Mexico by using function `coordinates()` from the `sp` slot in the `STIDF` object that stores the data.

```
R> NM.coords <- coordinates(brainst@sp)
```

Cluster detection with function `DetectClustersModel()` takes arguments `minDateUser` and `maxDateUser` to define the start and end date of the period which is considered when looking for clusters. In this example the time period has been constrained to 1985–1989. Furthermore, `typeCluster = "ST"` is used to look for spatio-temporal clusters.

```
R> nm.cl0 <- DetectClustersModel(brainst, NM.coords,
+    minDateUser = "1985-01-01", maxDateUser = "1989-01-01",
+    fractpop = 0.15, alpha = 0.05, typeCluster = "ST", R = NULL,
+    model0 = nm.m0, ClusterSizeContribution = "Expected")
```

180 possible clusters have been found this time. However, note that most of them overlap and may be different configurations of the same cluster. However, as mentioned earlier, secondary overlapping clusters will be removed in the analysis:

```
R> nm.cl0.s <- slimknclusters(brainst, nm.cl0)
R> nm.cl0.s
```

```
             x        y size      minDateCluster      maxDateCluster
0286 -106.3073 35.86930    3 1986-03-20 01:00:00 1988-03-20 01:00:00
0291 -107.7498 32.18214   12 1985-03-20 01:00:00 1986-03-20 01:00:00
0617 -107.7734 34.87512    1 1987-03-20 01:00:00 1987-03-20 01:00:00
0279 -105.4592 33.74524    3 1988-03-20 01:00:00 1988-03-20 01:00:00
     statistic       pvalue      risk cluster alpha_bonferroni
0286  7.493492 0.0001082553 0.6814588    TRUE       0.00015625
0291  4.019769 0.0045767186 0.2975976    TRUE       0.00015625
0617  2.122591 0.0393618579 0.8943567    TRUE       0.00015625
0279  1.985876 0.0462696218 0.7467474    TRUE       0.00015625
```

Hence, only 4 remain after removing overlapping clusters and these will be the only considered in the analysis.

Cluster detection after adjusting for covariates

In this case, we will use the inverse of the distance to LANL as a covariate as no other information about the areas is available. Distances have been computed using function `spDistsN1()` from package **sp** (Pebesma and Bivand 2005). Given that coordinates are expressed in longitude and latitude great circle distances are used.

```
R> dst <- spDistsN1(pts = NM.coords, pt = losalamos, longlat = TRUE)
```

Distances need to be put together in a way that values are available for all time periods. In this case, given that distances do not change over time, a vector is created by repeating the vector of distances as many times as time slots (years) we have in the dataset.

```
R> nyears <- length(unique(brainst$Year))
R> brainst$IDLANL <- rep(1 / dst, nyears)
```

With all these data we are now able to fit a baseline model.

```
R> nm.m1 <- glm(Observed ~ offset(log(Expected)) + IDLANL,
+    family = "poisson", data = brainst)
R> summary(nm.m1)

Call:
glm(formula = Observed ~ offset(log(Expected)) + IDLANL, family = "poisson",
    data = brainst)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4832  -0.9982  -0.4280   0.3775   3.1424

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.005721   0.029897  -0.191    0.848
IDLANL       0.338467   0.365200   0.927    0.354

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 631.64  on 607  degrees of freedom
Residual deviance: 630.84  on 606  degrees of freedom
AIC: 1586.8

Number of Fisher Scoring iterations: 5
```

Note how now the included covariate is not significant. For illustrative purposes, we will still keep the covariate in our model for the cluster detection. However, non-significant covariates will have a tiny impact on the clusters detected as they will not produce a change in the expected number of cases.

```
R> nm.cl1 <- DetectClustersModel(brainst, NM.coords, fractpop = 0.15,
+    alpha = 0.05, minDateUser = "1985-01-01", maxDateUser = "1989-01-01",
+    typeCluster = "ST", R = NULL, model0 = nm.m1,
+    ClusterSizeContribution = "Expected")
```

The number of clusters detected in this case is 179, very similar to the 180 found when no covariates where included in the model. This was expected as the included covariate was not significant. Similarly as in the previous example, non-overlapping clusters will be removed:

```
R> nm.cl1.s <- slimknclusters(brainst, nm.cl1)
R> nm.cl1.s
```

```
             x        y size      minDateCluster      maxDateCluster
0286 -106.3073 35.86930    3 1986-03-20 01:00:00 1988-03-20 01:00:00
0291 -107.7498 32.18214   12 1985-03-20 01:00:00 1986-03-20 01:00:00
0617 -107.7734 34.87512    1 1987-03-20 01:00:00 1987-03-20 01:00:00
0279 -105.4592 33.74524    3 1988-03-20 01:00:00 1988-03-20 01:00:00
     statistic       pvalue      risk cluster alpha_bonferroni
0286  6.857035 0.0002128539 0.6487021    TRUE       0.00015625
0291  4.103669 0.0041721334 0.3008419    TRUE       0.00015625
0617  2.138255 0.0386426187 0.8981373    TRUE       0.00015625
0279  2.007058 0.0451208529 0.7512122    TRUE       0.00015625
```

Now, the same 4 clusters are detected. Note how these clusters only cover a few areas, but over several years.

In order to exploit the output from `DetectClustersModel()`, function `get.stclusters()` will take the data and this output to return a list with the indices of the areas in the clusters. The next example shows how to add a new variable to `brainst` with the space-time regions in the most significant cluster, which is displayed in Figure 7.

```
R> stcl <- get.stclusters(brainst, nm.cl0.s)
R> brainst$CLUSTER <- ""
R> brainst$CLUSTER[ stcl[[1]] ] <- "CLUSTER"
```

6. Discussion

In this paper we have introduced **DClusterm**, a new package for the R statistical computing software for the detection of disease clusters using a model-based approach, following recent developments by several authors. Clusters are represented by dummy variables that are introduced into a generalized linear model and different likelihoods can be used to account for different types of data. Because of this model-based approach, fixed effects (to consider relevant risk factors) and random effects (to account for other non-spatial unmeasured risk factors) can be put in the linear predictor as well.

In our examples we have considered well-known datasets to show how the functions in the **DClusterm** package tackle the problem of cluster detection. The results are similar to those found in relevant papers where the same datasets have been analyzed using a similar methodology. In particular, we have considered the case of the detection of clusters in space and space-time, zero-inflated data and over-dispersed data.

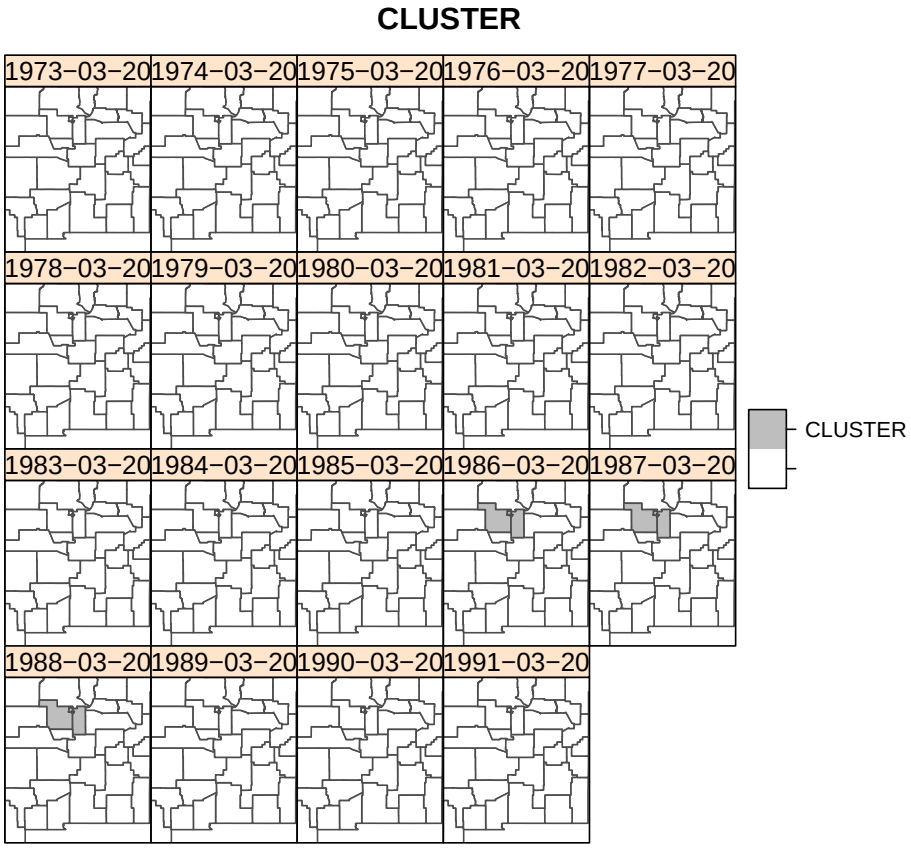

Figure 7: Most significant spatio-temporal cluster of brain cancer detected in New Mexico.

7. Acknowledgments

This package was initially developed by Paula Moraga as a Google Summer of Code project. Virgilio Gómez-Rubio has been supported by grant PPIC-2014-001, funded by Consejería de Educación, Cultura y Deportes (JCCM) and FEDER, and grant MTM2016-77501-P, funded by Ministerio de Economía y Competitividad.

References

Allévius B (2018). *scanstatistics: Space-Time Anomaly Detection using Scan Statistics*. R package version 1.0.1, URL https://CRAN.R-project.org/package=scanstatistics.

Auguie B (2016). ***gridExtra**: Miscellaneous Functions for "Grid" Graphics*. R Package Version 2.2.1, URL https://CRAN.R-project.org/package=gridExtra.

Bates D, Mächler M, Bolker B, Walker S (2015). "Fitting Linear Mixed-Effects Models Using **lme4**." *Journal of Statistical Software*, **67**(1), 1–48. doi:10.18637/jss.v067.i01.

Bilancia M, Demarinis G (2014). "Bayesian Scanning of Spatial Disease Rates with Integrated Nested Laplace Approximation (INLA)." *Statistical Methods & Applications*, **23**(1), 71–94. ISSN 1618-2510. URL http://dx.doi.org/10.1007/s10260-013-0241-8.

Chavent M, Kuentz V, Labenne A, Saracco J (2017). *ClustGeo: Hierarchical Clustering with Spatial Constraints*. R package version 2.0, URL https://CRAN.R-project.org/package=ClustGeo.

Dean CB (1992). "Testing for Overdispersion in Poisson and Binomial Regression Models." *Journal of the American Statistical Association*, **87**(418), 451–457.

Dunn OJ (1958). "Estimation of the Means for Dependent Variables." *Annals of Mathematical Statistics*, **4**(29), 1095–1111. doi:doi:10.1214/aoms/1177706374.

French J (2015). *smerc: Statistical Methods for Regional Counts*. R package version 0.2.2, URL https://CRAN.R-project.org/package=smerc.

French J (2018). *smacpod: Statistical Methods for the Analysis of Case-Control Point Data*. R package version 2.0, URL https://CRAN.R-project.org/package=smacpod.

Gómez-Rubio V, López-Quílez A (2010). "Statistical Methods for the Geographical Analysis of Rare Diseases." *Advances in Experimental Medicine and Biology*, **686**, 151–171.

Gómez-Rubio V, Moraga P, Molitor J (2017). "Fast Bayesian Classification for Disease Mapping and the Detection of Disease Clusters." *Under Review*.

Gómez-Rubio V, Ferrándiz-Ferragud J, López-Quílez A (2005). "Detecting Clusters of Disease with R." *Journal of Geographical Systems*, **7**(2), 189–206.

Jung I (2009). "A Generalized Linear Models Approach to Spatial Scan Statistics for Covariate Adjustment." *Statistics in Medicine*, **28**(7), 1131–1143.

Kim AY, Wakefield J (2016). ***SpatialEpi**: Methods and Data for Spatial Epidemiology*. R Package Version 1.2.2, URL https://CRAN.R-project.org/package=SpatialEpi.

Kleinman K (2015). ***rsatscan**: Tools, Classes, and Methods for Interfacing with SaTScan Stand-Alone Software*. R Package Version 0.3.9200, URL https://CRAN.R-project.org/package=rsatscan.

Kulldorff M (1997). "A Spatial Scan Statistic." *Communications in Statistics — Theory and Methods*, **26**(6), 1481–1496.

Liverani S, Hastie DI, Azizi L, Papathomas M, Richardson S (2015). "PReMiuM: An R Package for Profile Regression Mixture Models Using Dirichlet Processes." *Journal of Statistical Software*, **64**(7), 1–30. URL http://www.jstatsoft.org/v64/i07/.

McCullagh P, Nelder J (1989). *Generalized Linear Models*. 2nd edition. Chapman and Hall, London.

Meyer S, Held L, Höhle M (2017). "Spatio-Temporal Analysis of Epidemic Phenomena Using the R Package surveillance." *Journal of Statistical Software*, **77**(11), 1–55. doi:10.18637/jss.v077.i11.

Moraga P (2017). "SpatialEpiApp: A Shiny web application for the analysis of spatial and spatio-temporal disease data." *Spatial and Spatio-temporal Epidemiology*, (23), 47–57.

Neuwirth E (2014). ***RColorBrewer**: ColorBrewer Palettes*. R Package Version 1.1-2, URL https://CRAN.R-project.org/package=RColorBrewer.

Pebesma E (2012). "**spacetime**: Spatio-Temporal Data in R." *Journal of Statistical Software*, **51**(1), 1–30. ISSN 1548-7660. doi:10.18637/jss.v051.i07. URL https://www.jstatsoft.org/index.php/jss/article/view/v051i07.

Pebesma E, Bivand R (2005). "Classes and Methods for Spatial Data in R." *R News*, **5**(2). URL https://cran.r-project.org/doc/Rnews/.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rue H, Martino S, Chopin N (2009). "Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations (with discussion)." *Journal of the Royal Statistical Society, Series B*, **7**(2), 319–392.

Ryan JA, Ulrich JM (2014). ***xts**: eXtensible Time Series*. R Package Version 0.9-7, URL https://CRAN.R-project.org/package=xts.

Salmon M, Schumacher D, Höhle M (2016). "Monitoring Count Time Series in R: Aberration Detection in Public Health Surveillance." *Journal of Statistical Software*, **70**(10), 1–35. doi:10.18637/jss.v070.i10.

Sarkar D, Andrews F (2016). ***latticeExtra**: Extra Graphical Utilities Based on Lattice*. R Package Version 0.6-28, URL https://CRAN.R-project.org/package=latticeExtra.

Ugarte MD, Ibáñez B, Militino AF (2004). "Testing for Poisson Zero Inflation in Disease Mapping." *Biometrical Journal*, **46**(5), 526–539.

Ugarte MD, Ibáñez B, Militino AF (2006). "Modelling Risks in Disease Mapping." *Statistical Methods in Medical Research*, **15**, 21–35.

Valles G (2014). *AMOEBA: A Multidirectional Optimum Ecotope-Based Algorithm*. R package version 1.1, URL https://CRAN.R-project.org/package=AMOEBA.

Waller L, Turnbull B, Clark L, Nasca P (1992). "Chronic Disease Surveillance and Testing of Clustering of Disease and Exposure: Application to Leukemia Incidence in TCE-contamined Dumpsites in Upstate New York." *Environmetrics*, **3**, 281–300.

Waller LA, Gotway CA (2004). *Applied Spatial Statistics for Public Health Data*. John Wiley & Sons, Hoboken, New Jersey.

Wang TC, Hsu TC, Phoa FKH (2016). *SNscan: Scan Statistics in Social Networks*. R package version 1.0, URL https://CRAN.R-project.org/package=SNscan.

Zeileis A, Kleiber C, Jackman S (2008). "Regression Models for Count Data in R." *Journal of Statistical Software*, **27**(1), 1–25. ISSN 1548-7660. doi:10.18637/jss.v027.i08. URL https://www.jstatsoft.org/index.php/jss/article/view/v027i08.

Zhang T, Lin G (2009). "Spatial Scan Statistics in Loglinear Models." *Computational Statistics and Data Analysis*, **53**(8), 2851–2858.

Affiliation:

Virgilio Gómez-Rubio
Department of Mathematics
School of Industrial Engineering
University of Castilla-La Mancha
02071 Albacete, Spain
E-mail: virgilio.gomez@uclm.es
URL: http://www.uclm.es/profesorado/vgomez

Paula Moraga
Centre for Health Informatics, Computing and Statistics (CHICAS)
Lancaster Medical School
Lancaster University
Lancaster, LA1 4YW
United Kingdom
E-mail: p.e.moraga-serrano@lancaster.ac.uk URL: https://paula-moraga.github.io/

John Molitor
College of Public Health and Human Sciences
Oregon State University
Corvallis, Oregon 97331, United States
E-mail: John.Molitor@oregonstate.edu
URL: http://health.oregonstate.edu/people/molitor-john

Barry Rowlingson
Lancaster Medical School
Furness Building
Lancaster University
Bailrigg, Lancaster LA1 4YG, United Kingdom
E-mail: b.rowlingson@lancaster.ac.uk
URL: http://www.lancaster.ac.uk/fhm/about-us/people/barry-rowlingson

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd