Title
Spatial Clustering to Search for Hot Spots

Permalink
https://escholarship.org/uc/item/78f8k1dc

Author
He, Fei

Publication Date
2015-01-01

Peer reviewed|Thesis/dissertation
Spatial Clustering to Search for Hot Spots

A Dissertation submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Applied Statistics

by

Fei He

December 2015

Dissertation Committee:

Dr. Daniel Jeske, Chairperson
Dr. James Flegal
Dr. Rick Redak
The Dissertation of Fei He is approved:

________________________________________

________________________________________

________________________________________

Committee Chairperson

University of California, Riverside
Acknowledgments

I am most grateful to my advisor, Dr. Daniel Jeske, for his patient and dedicated guidance. I would also like to thank Dr. James Flegal, for his insightful and constructive comments and suggestions on the McMC work presented in this thesis, and Dr. Rick Redak and Dr. Elizabeth Grafton-Cardwell, for their entomological motivation, experimental data and many helpful suggestions.
To my parents.
ABSTRACT OF THE DISSERTATION

Spatial Clustering to Search for Hot Spots

by

Fei He

Doctor of Philosophy, Graduate Program in Applied Statistics
University of California, Riverside, December 2015
Dr. Daniel Jeske, Chairperson

The cluster analysis has been widely applied to many fields. In this dissertation, Hot spot detection, as an important application of the spatial clustering, is thoroughly introduced and the current methodologies used in hot spot detection are presented and compared. This dissertation also introduces a GLMM based scan method to identify hot spots on spatial lattice arrays. Three features introduced by the proposed methodology are: (1) A Generalized Linear Mixed Model (GLMM) that offers realism for correlated count data and captures overdispersion in the data; (2) A border comparison that is used to determine the significance of a candidate hot spot at each stage of sequential searches; (3) An iterative process that finds secondary hot spots by conditioning on previously found hot spots. A heuristic search algorithm (MBSM-H) is proposed that reduces the high computational demands associated with a global search algorithm (MBSM-G). Both algorithms are illustrated through simulated examples and an application to Integrated Pest Management where an orchard is assessed for potential hot spots of a pest. Comparisons between the GLMM based scan method and alternative methodologies are presented in the dissertation.
Contents

List of Figures x
List of Tables xi

I First Part

1 Introduction 2
  1.1 Integrated Pest Management, a motivating example . . . . . . . . . . . . 2
  1.2 Clustering Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
  1.3 Global Test for Spatial Clustering . . . . . . . . . . . . . . . . . . . . . 10
  1.4 Hot Spot Detection on a Spatial Lattice . . . . . . . . . . . . . . . . . . 13

2 Preliminaries on GLMM 17
  2.1 Generalized Linear Mixed Model (GLMM) Definition . . . . . . . . . . . . 17
  2.2 GLMMs Fitting Algorithms . . . . . . . . . . . . . . . . . . . . . . . . . 18
    2.2.1 Gauss-Hermite Quadrature . . . . . . . . . . . . . . . . . . . . . 18
    2.2.2 EM Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
    2.2.3 Metropolis-Hasting Algorithm . . . . . . . . . . . . . . . . . . . . . 21
    2.2.4 Monte Carlo EM Algorithm . . . . . . . . . . . . . . . . . . . . . 22
    2.2.5 Pseudo-Likelihood . . . . . . . . . . . . . . . . . . . . . . . . . . 23

3 Related Clustering Methods 26
  3.1 Co-clustering . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
    3.1.1 GLMM based Checkerboard Structure Co-clustering . . . . . . . . . 28
      3.1.1.1 Global Optimization Algorithm . . . . . . . . . . . . . 29
      3.1.1.2 Heuristic Optimization Algorithm . . . . . . . . . . . . 30
      3.1.1.3 An Illustration to the IPM Applications . . . . . . . . . . 31
  3.2 Bayesian Disease Mapping . . . . . . . . . . . . . . . . . . . . . . . . . . 33
    3.2.1 Bayesian approach to Relative Risk Estimation . . . . . . . . . . 34
    3.2.2 Disease Cluster Detection . . . . . . . . . . . . . . . . . . . . . . 37
      3.2.2.1 Cluster Detection using Posterior Measures . . . . . . . . 37
      3.2.2.2 Cluster Detection using Residuals . . . . . . . . . . . . 38
  3.3 Spatial Scan Statistics . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
    3.3.1 The Geographical Analysis Machine (GAM) . . . . . . . . . . . . 40
    3.3.2 Cluster Evaluation Permutation Procedure (CEPP) . . . . . . . . 41
    3.3.3 Kulldorff’s Scan Statistics (KSS) . . . . . . . . . . . . . . . . . 43
List of Figures

1.1 Cottony cushion scale counts on 100% sampling of the *Citrus grandis Osbeck* × *C. Paradisi* Macf. ‘Melogold’ grapefruit orchard in Exeter, CA 3
1.2 Cottony cushion scale counts on 10% sampling of the *Citrus grandis Osbeck* × *C. Paradisi* Macf. ‘Melogold’ grapefruit orchard in Exeter, CA 4
1.3 Cottony cushion scale counts on 4% sampling of the *Citrus grandis Osbeck* × *C. Paradisi* Macf. ‘Melogold’ grapefruit orchard in Exeter, CA 4
1.4 Hierarchical Clustering of the Car Models 7
1.5 K-means Clustering of the Car Models 8

3.1 Co-clustering of basketball player data 27
3.2 Co-clustering of *Citrus grandis Osbeck* × *C. Paradisi* Macf. ‘Melogold’ grapefruit orchard in Exeter, CA 31

4.1 Border neighboring a candidate hot spot 61
4.2 Simulated example 1 67
4.3 Simulated example 2 68
4.4 Simulated example 3 69
4.5 Simulated example 4 70
4.6 Simulated example 5 71
4.7 Simulated example 6 72

5.1 Example of Cold spots 96
List of Tables

4.1 Ratio of expected number of lattice points in the border and in the candidate hot spot ........................................... 62
4.2 Normalized VI comparison of alternative hot spot identification methods 74
4.3 Computation time comparison of alternative hot spot identification methods 76
Part I

First Part
Chapter 1

Introduction

1.1 Integrated Pest Management, a motivating example

Integrated pest management (IPM) integrates practices to economically control for pest problems and minimize human health and environmental risks at the same time. Pests start to infect random small areas at their initial settlement into the orchard, and the random spread of high pest density in localized areas is referred to by pest management specialists as hot spots. In order to avoid loss caused by further expansion of the hot spots into large regions with smoother density, IPM procedures are carried out before or at the outbreak of hot spots. Therefore, it is essential to have hot spots detected accurately and efficiently to take action of IPM procedures. In addition, the orchards we usually experience are of spatial lattice structure where each observation can be indexed by an $x$ and $y$ coordinates. The new method of hot spot detection introduced in this dissertation is motivated by an application to IPM, where we identify hot spots within an orchard for potential pest problems. Figure 1.1 shows an exhaustive sampling of a grapefruit orchard on a spatial lattice. The data is the total counts of cottony cushion scale insects collected from 8 branches per tree for 850 trees. Figure 1.2 shows a 10%
subsampling, and Figure 1.3 shows a 4% subsampling of the data. A shifted scheme that will be introduced in a later chapter was used to subsample the orchard. The color coded regions on Figures 1.1-1.3 will be discussed in a subsequent chapter of this dissertation.

Figure 1.1: Cottony cushion scale counts on 100% sampling of the *Citrus grandis* Osbeck × *C. Paradisi* Macf. ‘Melogold’ grapefruit orchard in Exeter, CA

Figure 1.2: Cottony cushion scale counts on 10% sampling of the *Citrus grandis* Osbeck × *C. Paradisi* Macf. ‘Melogold’ grapefruit orchard in Exeter, CA
1.2 Clustering Analysis

One of the popular approaches for hot spot detection is through cluster analysis. Cluster analysis, also known as clustering, is an approach to group a set of objects based on some sense of similarity, and the groups that objects got assigned to are called clusters. The main purpose of clustering is to produce a rather simple group structure for a complex data set. It is a common technique for statistical data analysis that has been widely applied to many fields such as data mining, machine learning, and bioinformatics.

A number of different algorithms are developed to cluster observed data into meaningful structures. For instance, hierarchical clustering is a typical algorithm that finds clusters based on distance connectivity, and the hierarchical clustering results in that the distances among objects within a same cluster are smaller compared to objects from outside of the cluster. Hierarchical Clustering techniques proceed by either a series of successive mergers or a series of successive divisions, i.e. agglomerative or divisive.
Agglomerative hierarchical clustering starts with each individual object, and then the most similar objects are first grouped, and these initial groups are merged according to their similarities. Eventually, as the similarity decreases, all subgroups are combined into a single cluster. Therefore, every object ends up within one cluster by using this algorithm. In agglomerative hierarchical clustering, once a cluster is formed, it will no longer be split, and it can only be combined with other clusters. Divisive hierarchical clustering work in an opposite direction, which starts with every object included in one cluster, and then this initial single group is divided into two subgroups such that the objects in one subgroup are far from the objects in the other. These subgroups are then further divided into dissimilar subgroups, and the process continues until there are as many subgroups as objects. The results of both agglomerative and divisive hierarchical clusterings can be displayed in the form of a two-dimensional diagram known as dendrogram, which illustrates the mergers or divisions made at successive levels [27].

In hierarchical clustering algorithms, the merging of clusters is based on mainly three linkage criteria: single linkage (minimum distance or nearest neighbor), complete linkage (maximum distance or farthest neighbor), and average linkage (average distance). The following information should be known when conducting the hierarchical clustering:

- A measure of similarity or dissimilarity among objects, for example, to use the Euclidean distance

- A criteria used to make decision on which clusters are merged at successive steps, for example, to use the single linkage linkage

Figure 1.4 shows and example of clustering by hierarchical algorithm, and the data used in this example is from [20]. The data collected from the 1974 Motor Trend US magazine shows fuel consumption and 10 aspects such as miles per gallon, number cylinders, weight,
etc. of 32 1973 – 1974 automobiles models, and the clustering is to group the models based on the similarities considering all the aspects. As seen in the following figure 1.4, the dendrogram groups 32 models of cars into 5 groups based on all 10 aspects using the hierarchical clustering.

![Cluster Dendogram of MTCARS Data](image)

Figure 1.4: Hierarchical Clustering of the Car Models

Non-hierarchical clustering algorithms such as K-means is to group \( n \) observations into \( k \) clusters based on the centroid of the data so that each observation belongs to a cluster with the nearest centroid. It differs from hierarchical clustering in several ways:

- The number of clusters have to be determined in advance.
The algorithm repeatedly reassigns observations to clusters, so the same observation can move from cluster to cluster during the analysis. In contrast, in agglomerative hierarchical clustering, the members of each cluster is fixed once they have been assigned to an existing cluster.

The algorithm starts out with an initial set of means and classifies observations based on their distances to the centroids. Secondly, cluster means are computed again including the newly assigned members to the cluster. Thirdly, reclassify all observations based on the new set of means. Above steps repeat until cluster means do not change significantly between successive two steps. Finally, once again the cluster means are calculated, and observations are assigned to the permanent set of clusters. Figure 1.5 shows an example of clustering by K-means algorithm, and the data used in this example is again the Motor Trend data from [20].

![K-means Clustering of the Car Models](image)

**Figure 1.5: K-means Clustering of the Car Models**

Here all observations are first represented on a two-dimensional plot using first and second principal components, and in this example, we can see that these two
components explain 83.58\% of the point variability from all 10 aspects of the car motors. Around each cluster an ellipse is drawn, and in figure 1.5, 5 clusters are determined to be used before the clustering.

More clustering algorithms involve complex statistical models such as distribution-based clustering. Given that observations coming from the same cluster can usually be identified as they belong to a particular statistical distribution, distribution-based clustering produces complex models for clusters that can capture correlation and dependence between attributes. However, a complex model usually explains the data better but tend to overfit the data. One popular method is to use the expectation-maximization algorithm for Gaussian mixture models. Given proposed clusters, the EM algorithm computes and maximizes the overall probability or likelihood of the data, and then the one with highest likelihood value is confirmed as a cluster. Unlike the classic implementation of hierarchical or K-means method, the EM algorithm can be applied to both continuous and categorical variables. Despite the values added by the sophistication of the distribution-based clustering, these algorithms usually require rather strong assumptions on the distribution of the data, and there might not always be a defined model.

Cluster analysis is very useful in many fields of studies, and these studies are reviewed in [6]. In medicine field, clustering the symptoms of diseases improves the diagnosis and cure. In plant and animal ecology studies, cluster analysis is used to describe and compare organisms in heterogeneous environments, and it is also used in plant systematics to generate clusters of organisms that share a number of attributes. In bioinformatics applications, sequence analysis is used to cluster homologous sequences into gene families. In crime analysis, cluster analysis is used to identify areas having greater incidences of particular types of crime. In text mining, cluster analysis is used
to group similar items from different linguistic sources according to their meaning. In general, whenever we need to classify an extensive amount of information into a more manageable and meaningful structure, cluster analysis is a great utility to consider.

1.3 Global Test for Spatial Clustering

In addition to the clustering applications previously mentioned, the clustering applied to spatial data not only groups the observations into clusters, but also considers the spatial autocorrelation among the data, so the spatial clustering usually requires clusters consist of spatially consecutive rows and columns. The spatial clustering techniques in earlier development are for overall clustering, which only test for the existence of clusters in the data but not to detect the location of the clusters. Here, we introduce some of the test statistics for overall clustering. Proposed by [44], Moran’s I statistic measures the spatial autocorrelation. For geographical data, the similarity of values $X_i$ and $X_j$ between two areal units $i$ and $j$ are weighted by the proximity $w_{ij}$. The Moran’s I statistic is defined as follows

$$I = \frac{N}{\sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij} (X_i - \bar{X})(X_j - \bar{X})}{\sum_i (X_i - \bar{X})^2}$$

where $N$ is the number of areal units, $X_i$ and $X_j$ are the values at unit $i$ and $j$, $\bar{X}$ is the overall mean, and $w_{ij}$ is an element of the weight matrix, and each element is consists a binary weight such that

$$w_{ij} = \begin{cases} 
1 & \text{if unit } i \text{ and } j \text{ are connected} \\
0 & \text{if unit } i \text{ and } j \text{ are not connected} \end{cases}$$
The value of $I$ has range from $-1$ to $1$, and $-1$ indicates a perfect dispersion in the data while $1$ indicates perfect correlation, and $0$ means a random pattern in the spatial data. Under the null hypothesis of no spatial autocorrelation in the data, the Moran’s I statistic is asymptotically normally distributed with expected value of $I$

$$E(I) = \frac{-1}{N-1}$$

and variance of

$$Var(I) = \frac{NS_4 - S_3 S_5}{(N-1)(N-2)(N-3)(\sum_i \sum_j w_{ij})^2} - E^2(I)$$

where

$$S_1 = \frac{1}{2} \sum_u \sum_j (w_{ij} + w_{ji})^2$$

$$S_2 = \sum_i \left( \sum_j w_{ij} + \sum_j w_{ji} \right)^2$$

$$S_3 = \frac{N^{-1} \sum_i (x_i - \bar{x})^4}{(N^{-1} \sum_i (x_i - \bar{x})^2)^2}$$

$$S_4 = \left( N^2 - 3N + 3 \right) S_1 - NS_2 + 3(\sum_i \sum_j w_{ij})^2$$

$$S_5 = \left( N^2 - N \right) S_1 - 2NS_2 + 6(\sum_i \sum_j w_{ij})^2.$$  

Thus, if the two-sided p-value is less than 0.025, then it can be concluded that the spatial autocorrelation is significant at 5% level.
Similarly, Geary’s $c$ statistic [22] measures spatial similarity, and it is formulated as follows:

$$
c = \frac{N - 1}{2 \sum_i(X_i - X)^2} \frac{\sum_i \sum_j w_{ij}(X_i - X_j)^2}{\sum_i \sum_j w_{ij}}.
$$

The value of $c$ ranges from 0 to 2 with 0 indicating perfect positive spatial correlation and 2 indicating perfect negative spatial correlation. $c$ is also asymptotically normally distributed. Global measures are a single value that apply to the entire study region, and it assumes that the same pattern occurs over the entire geographic area. Global test for clustering is usually used as a first step to determine if there is evidence of spatial association among the data, and further procedure is needed if the interest is to find the location of the clusters. Under null hypothesis, the data is totally random, i.e. the population is equally likely to be a "case".

Another global test of clustering is the U-statistic test [61], which tests a total randomness under null hypothesis against a pattern of clustering under the alternative. Under the null hypothesis, it assumes that each of the study population is equally likely to be a disease case, i.e. within each of the predefined cells, the number of cases is an independent Poisson random variable with mean proportional to the population in that cell. This test statistic is constructed using the mean distance between all pairs of disease cases, and under the null hypothesis the test statistic has an asymptotically normal distribution. However, since the test statistic considers the position of the cases only though the distance between pairs of the cases, but not considering the population density from which the cases are discovered. For example, the test statistic tend to be negative when the clusters exist within urban areas where the population is more dense and the cases are close to each other. In contrast, the test statistic tend to be positive if the clusters exist within rural areas where the distance between a pair of cases is longer.
1.4 Hot Spot Detection on a Spatial Lattice

Hot spot detection is an application of spatial clustering, and in the IPM application requires a clustering tool which works for the data on a spatial lattice. Various hot spot detection methods can be applicable on a spatial lattice array data. A hot spot on a spatial lattice is a high density localized area, and a localized area with low density can be called a cold spot. The new method introduced in this dissertation can be used to detect both hot and cold spots on a spatial lattice, but for simplicity, the term "hot spot" is used for both hot and cold spots unless specified otherwise. From literature studies, two spatial cluster detection methodologies can be applied to identify hot spots for count data on a spatial lattice. Those are the spatial scan statistic [29] and the Spatial Analysis by Distance IndicEs (SADIE) red-blue plots [52]. An additional related work is the GLMM based checkerboard structure co-clustering methodology [63]. In the latter article, rows and columns are iteratively divided into groups to reach an optimal configuration that creates a set of co-clusters. Additional information on what levels of counts warrant treatment is then used to classify the co-clusters as 'treat' or 'do not treat.' Since our purpose in this new method is to identify hot spots in the sense that they are locally different from the neighboring areas, the checkerboard co-clustering methodology is not directly comparable but will be introduced in the subsequent chapter.

A significant amount of work has been done on mapping techniques to detect spatial patterns of epidemiological incidents. Some of the popular hot spot mapping techniques are Geographical Analysis Machine (GAM) [49], Cluster Evaluation Permutation Procedure(CEPP) [59], and Scan Statistic [29], [31]. GAM searches for high density areas of incidents by proposing fixed size circles, it was used in epidemiology disease mapping applications and relates closely to the scan statistic by Kulldorff [29], [31]. These
methods will also be introduced in subsequent chapter, and scan statistic is compared to our proposed new method. There is also a large literature on Bayesian disease mapping, which can be applied for hot spot detection. However, these mapping techniques mainly focus on smoothing risk estimates over a set of contiguous areas [3], [35], so they are not comparable to our proposed method in this dissertation but will be introduced in the subsequent chapter.

The model based scan method proposed in this dissertation combines scan statistics and GLMMs to search for hot spots on a spatial lattice array. A border test is used to confirm the presence of a hot spot. The presence of secondary hot spots is further identified through sequential searches with previously identified hot spots removed. The utility of the model based scan method is available for a variety of applications of hot spot detection on spatial lattice array settings, including but not limited to IPM and crime hot spot detections. Disease mapping applications typically do not involve lattice arrays. However, as is often done with crime rates, disease rates could conceivably be arrayed on a lattice through the use of an overlaid lattice structure.

The model based scan method introduced in this dissertation is demonstrated using both exhaustive and subsampled lattice arrays on a spatial lattice. The new methodology proposed in this dissertation has the following main contributions:

1. Three features are introduced by the model based scan method to improve the effectiveness of scan statistics: (1) A Generalized Linear Mixed Model (GLMM) that offers realism for correlated count data and captures overdispersion in the data; (2) A border comparison that is used to determine the significance of a candidate hot spot at each stage of sequential searches; (3) An iterative process that finds secondary hot spots by conditioning on previously found hot spots.
2. A global search algorithm (MBSM-G) is used to implement the new methodology, and a distributed computing algorithm is discussed to make it feasible. A heuristic search algorithm (MBSM-H) is proposed, which mitigates much of the extensive computing associated with the MBSM-G.

3. A comparison of alternative hot spot identification methods: Kulldorff’s scan statistics (KSS), SADIE red-blue plot (SRB), and the model based scan method (both MBSM-G and MBSM-H) using an information theory metric.

4. Illustrations of the proposed methods using real cottony cushion scale insects data.

This dissertation is organized as following. In Chapters 2, some preliminaries on Generalized Linear Mixed model is introduced. In Chapter 3, related clustering and hot spot detection methodologies such as GLMM based checkerboard structure co-clustering, SAIDE red-blue plot method, Bayesian Disease mapping and Kulldorff’s scan statistic are reviewed, and disadvantages of each are discussed. In Chapters 4, the model based scan method and the MBSM-G and MBSM-H algorithms used to implement it are introduced. The performance of alternative hot spot identification methods is analyzed through simulated examples as well as the real data set of the cottony cushion scale insects. An R function to implement MSBM-G and MBSM-H is available in the Appendix. We close the dissertation by giving recommendations on using the model based scan method and discussing future work.
Chapter 2

Preliminaries on GLMM

2.1 Generalized Linear Mixed Model (GLMM) Definition

GLMMs are extensions to the Generalized Linear Model in which the linear predictor contains random effects in addition to the usual fixed effects such as

\[ Y_{ij} \mid u \sim f_{Y_{ij} \mid u}(y_{ij} \mid u), i = 1, 2, \ldots, m; j = 1, 2, \ldots, n \]

\[ g(\mu_i) = x_i'\beta + z_i'u \]

\[ u = (u_1, u_2, \ldots, u_m)' \sim f_u(u \mid D) \]  

(2.1)

where \( Y_{ij} \) is the \( j \)th observation from the \( i \)th subject, \( m \) is the total number of subjects, \( n \) is the number of observations on each subject (supposed it is the same for all the subjects), \( \mu_i \) is the conditional (on \( u_i \)) expected value associated with the \( i \)th subject, and \( g() \) is a known function, called link function. \( x_i' \) is the \( i \)th row of the model matrix for the fixed effects, \( \beta \) is the fixed effects parameter vector, \( z_i' \) is the \( i \)th row of the model matrix for the random effects, \( u \) is the random effects vector, and \( D \) represents the parameters \( u \) depends on.
2.2 GLMMs Fitting Algorithms

2.2.1 Gauss-Hermite Quadrature

Start by considering a GLMM with a single, normally distributed random effect, i.e., $u \sim (0, \sigma_u^2 \mathbf{I})$, the likelihood for this model is just the product of one-dimensional integrals:

$$L = \int \cdots \int f_{Y_{ij}|U_i}(y_{ij} \mid u_i)f_{U_i}(u_i)du_i = \prod_i \prod_j \int f_{Y_{ij}|U_i}(y_{ij} \mid u_i)\frac{e^{-u_i^2/(2\sigma_u^2)}}{\sqrt{2\pi\sigma_u^2}}du_i.$$

Let $u = \sqrt{2}\sigma_u v$ and let $h(u) = f_{Y_{ij}|U_i}(y_{ij} \mid u_i)$, the integrals can be written as:

$$\int_{-\infty}^{\infty} h(u)\frac{e^{-u^2/(2\sigma_u^2)}}{\sqrt{2\pi\sigma_u^2}}du = \int_{-\infty}^{\infty} h(\sqrt{2}\sigma_u v)\frac{e^{-v^2}}{\sqrt{\pi}}dv \equiv \int_{-\infty}^{\infty} h^*(v)e^{-v^2}dv$$

where $h^*(g) = h(\sqrt{2}\sigma_u v)/\sqrt{\pi}$. It was showed in [41] that the Gauss-Hermite quadrature approximates the integrals as a weighted sum:

$$\int_{-\infty}^{\infty} h^*(v)e^{-v^2}dv = \sum_{k=1}^{D} h^*(x_k)w_k$$

where $w_k$’s are the weights and $x_k$’s are the evaluation points, which can be calculated as the roots of $n$ degree Hermite polynomial $H_n(x)$ and associated with weights $w_k$ given by

$$w_k = \frac{2^{n-1}n!\sqrt{\pi}}{n^2[H_{n-1}(x_k)]^2}.$$
Therefore, the log-likelihood becomes:

\[
l = \sum_i \log \int_{-\infty}^{\infty} \prod_j f_{Y_{ij} | U_i}(y_{ij} | u_i) f_{U_i}(u_i) e^{-\frac{u_i^2}{2\sigma_u^2}} \frac{1}{\sqrt{2\pi\sigma_u^2}} du_i = \sum_i \log \left[ \sum_{k=1}^{D} \left( h^*(x_k) \frac{w_k}{\sqrt{\pi}} \right) \right].
\]

It is shown that quadrature with \( D = 30 \) is usually enough for a good degree of approximation [41].

Now consider a more complicated GLMM with spatial correlated random effects, i.e., \( u \sim (0, \sigma_u^2 D) \) where \( D \) takes on spatial covariance structure, for example, an exponential spatial covariance structure: \( D = \sigma^2 \exp(-\frac{d_{lr}}{\theta}) \), where \( d_{lr} \) is the Euclidean distance between the \( l^{th} \) and \( r^{th} \) vector of the coordinates of two locations. Thus, the likelihood becomes:

\[
L = \int \cdots \int \prod_i \prod_j f_{Y_{ij} | U_i}(y_{ij} | u_i) f_{U_i}(u_i) (2\pi)^{-\frac{D}{2}} |D|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2} u'D^{-1}u \right\} du \]

Since \( u_i \)'s are no longer independently and identically distributed, we cannot write above into product of one-dimensional integrals, and hence it is generally infeasible to use Gauss-Hermite Quadrature approximation.

### 2.2.2 EM Algorithm

In order to set up the EM algorithm, consider the random effects \( u \), to be the missing data, so the complete data, \( u = (y', u') \), and the complete-data log-likelihood is given by

\[
\log L_w = \log f_{Y|U}(y | u, \beta) + \log f_U(u | D).
\]
We can see that the estimation of $\beta$ only use $f_{Y\mid U}$ treating $u$ as known, and the estimation of $D$ only involves the distribution of $D$. The EM algorithm then takes the following steps:

1. Choose starting values $\beta^{(0)}$ and $D^{(0)}$, and set $m = 0$.

2. Calculate a new iteration of $\beta^{(0)}$ and $D^{(0)}$ which maximizes the following expected values respectively:

   (a) $\beta^{(m+1)}$ to maximize $E_{U\mid y, \beta^{(m)}}[\log f_{Y\mid U}(y, \beta)]$

   (b) $D^{(m+1)}$ to maximize $E_{U\mid y, D^{(m)}}[\log f_{U}(u \mid D)]$

   (c) Set $m = m + 1$

3. If convergence is achieved, declare the current values to be the MLEs; otherwise, return to step 2.

In general, neither of the expectations in 2(a) or 2(b) can be computed in a closed form for the spatial GLMMs. This is because the conditional distribution of $u \mid y$ involves $f_Y$ which requires high-dimensional integration. However, it is possible to produce random draws from the conditional distribution of $u \mid y$ using a Metropolis algorithm [41], which does not require specification of $f_Y$.

### 2.2.3 Metropolis-Hasting Algorithm

A candidate distribution is a distribution from which potential new samples are drawn and the acceptance function that gives the acceptance probability for the new samples is calculated [41]. If choose the candidate distribution $h_U(u)$ to be $f_U(u \mid D^{(m)})$ (where $D^{(m)}$ is the current estimate of $D$), then the acceptance function takes a particularly clean form. Let $u$ denote the current accepted draw from the conditional distribution
u | y, then draw a new value from the candidate distribution u' ∼ MNV \( \left( 0, D^{(m)} \right) \), and let \( u'_k \) denote the \( k \)th component of u', for each k, calculate the acceptance probability \( A_k (u, u') \) and update each \( k \)th component to \( u'_k \) as the new value with probability \( A_k (u, u') \); if none of the component got updated, retain u. By choosing \( h_U = f_U \), the acceptance ratio can be simplified as follows:

\[
A_k (u, u') = \min \{ 1, \frac{f_{y|U} \left( u' | y, \beta^{(m)}, D^{(m)} \right) f_U \left( u' \right)}{f_{y|U} \left( u | y, \beta^{(m)}, D^{(m)} \right) f_U \left( u \right)} \}
\]

This calculation involves only \( y | u \), which is just the specification of the generalized linear model portion of the model.

A single-component Metropolis-Hastings algorithm which specifically solves the fitting of the spatial non-Gaussian data was introduced by [62]. Let u be the current draw from the distribution u | y. For a new draw from the distribution, draw a new value for the \( k \)th component of u' based on the conditional distribution of \( u_k | u_j, j \neq k \), which is \( N \left( -\sum_{j \neq k} Q_{kj} u_j Q_{kj}^{-1}, Q_{kk}^{-1} \right) \) when \( u' \sim MNV \left( 0, D^{(m)} \right) \), where \( Q_{kj} \) is the \((k, j)\) element of the inverse of \( D^{(m)} \). Then accept \( u'_k \) as the new value with probability

\[
A_k (u, u') = \min \{ 1, \frac{\prod_j f_{y_{ij}|u_i} \left( u'_k | y_{ij}, \beta^{(m)}, D^{(m)} \right)}{\prod_j f_{y_{ij}|u_i} \left( u_k | y_{ij}, \beta^{(m)}, D^{(m)} \right)} \}.
\]

Otherwise, retain u.
2.2.4 Monte Carlo EM Algorithm

Incorporating the Metropolis step into the EM algorithm gives a Monte Carlo EM (MCEM) algorithm as follows:

1. Choosing stating values $\beta^{(0)}$ and $D^{(0)}$. Set $m = 0$.

2. Generate $M$ values, $u^{(1)}, u^{(1)}, \ldots, u^{(M)}$ from $f_{u|y}(u | y, \beta^{(m)}, D^{(m)})$ using the Metropolis algorithm described previously.

   (a) Choose $\beta^{(m+1)}$ to maximize a Monte Carlo estimate of $E_{U\mid y, \beta}^{(m)}[\log f_{Y\mid U}(y | u, \beta)]$; that is, maximize $(1/M) \sum_{k=1}^{M} \log f_{Y\mid U}(y | u^{(k)} , \beta)$.

   (b) Choose $D^{(m+1)}$ to maximize $(1/M) \sum_{k=1}^{M} \log f_{U}(u^{(k)} | D)$.

   (c) Set $m = m + 1$.

3. If convergence is achieved, declare the current value to be MLEs; otherwise, return to Step 2.

2.2.5 Pseudo-Likelihood

Another way to avoid high-dimensional integration is to use Pseudo-Likelihood (PL) approximation. PL is an approximation to the likelihood involving distributions with high-dimensional dependencies. It may either provide a computationally simpler problem for estimation, or may provide a way of obtaining explicit estimates of model parameters. PL is based on the following 4 approximations:

1. $y_{n \times 1} = \mu_{n \times 1} + e_{n \times 1}$, where $E[e | \mu]$ and $\text{var}(e | \mu) = \text{R}_{\mu}^{1/2} \text{R} \text{R}_{\mu}^{1/2}$ is a diagonal matrix containing evaluations at $\mu$ of the conditional variances of $y | \mu$, and $\text{R}$ is an unknown correlation matrix.
2. Let $\hat{\beta}$ and $\hat{u}$ be current estimates of $\beta$ and $u$, and define $\hat{\mu} = g^{-1} \left( X \hat{\beta} + Z \hat{u} \right)$ as 1st order Taylor series expansion to $e = y - \mu$ about $\hat{\beta}$ and $\hat{u}$,

\[
y_t - g^{-1} (x_t' \beta + z_t' u) = y_t - g^{-1} \left( x_t' \hat{\beta} + z_t' \hat{u} \right) - \sum_{i=1}^{p} \frac{\partial g^{-1} (x_t' \beta + z_t' u)}{\partial \beta_i} \bigg|_{(\hat{\beta}, \hat{u})} (\beta_i - \hat{\beta}_i) - \sum_{i=1}^{q} \frac{\partial g^{-1} (x_t' \beta + z_t' u)}{\partial u_i} \bigg|_{(\hat{\beta}, \hat{u})} (u_i - \hat{u}_i)
\]

where $x_t'$ and $z_t'$ are $t^{th}$ rows of $X$ and $Z$. Thus,

\[
e = y - g^{-1} \left( x_t' \hat{\beta} + z_t' \hat{u} \right) - \left( g^{-1} \right)' \bigg|_{(\hat{\beta}, \hat{u})} \left( x_t' \hat{\beta} + x_t' \beta + z_t' \hat{u} + z_t' u - x_t' \hat{\beta} + z_t' \hat{u} \right)
\]

\[\equiv \tilde{e}\]

where $(g^{-1})' \bigg|_{(\hat{\beta}, \hat{u})}$ is a diagonal matrix with elements consisting of first derivative of $(g^{-1})$ evaluated at the $x_t' \hat{\beta} + z_t' \hat{u}$.

3. Assume:

(a) $\tilde{e} | (\beta, u) \sim \text{MVN}$

(b) $E[\tilde{e} \mid (\beta, u)] = E[e \mid (\beta, u)] = 0$

(c) $\text{var} (\tilde{e} \mid (\beta, u)) = R_{\tilde{e}}^{1/2} R \tilde{e} R_{\tilde{e}}^{1/2}$

4. Note that $\left( g^{-1} \right)' \bigg|_{(\hat{\beta}, \hat{u})} = \text{diag} \left\{ \frac{1}{g(\mu_i)} \right\}$, since

\[g[g^{-1}(x)] = x \Rightarrow g'[g^{-1}(x)] \frac{dg^{-1}(x)}{dx} = 1\]
Using this result and above assumptions,

\[ y - g^{-1}(x'\beta + z'u) - (g^{-1})'(x'\beta - x'\beta + z'u - x'\beta + z'u) \mid (\beta, u) \]

\[ \sim \text{MVN} \left( 0, R^{1/2}_\mu RR^{1/2}_\mu \right) \]

Equivalently, letting \( g' (\hat{\mu}) = \text{diag} (g' (\hat{\mu})) \), we have

\[ y - g^{-1}(x'\beta + z'u) - (g^{-1})'(x'\beta - x'\beta + z'u - x'\beta + z'u) \mid (\beta, u) \]

\[ \sim \text{MVN} \left( 0, g'(\hat{\mu}) R^{1/2}_\mu RR^{1/2}_\mu g'(\hat{\mu}) \right) \]

Let \( V = g' (\hat{\mu}) + g'(\hat{\mu}) (y - \hat{\mu}) \), then we have

\[ V \mid (\beta, u) \sim \text{MVN} \left( X\beta + Zu, g'(\hat{\mu}) R^{1/2}_\mu RR^{1/2}_\mu g'(\hat{\mu}) \right) \]

Since \( u \sim N(0, D) \), it would follow that

\[ V \sim \text{MVN} \left( X\beta, g'(\hat{\mu}) R^{1/2}_\mu RR^{1/2}_\mu g'(\hat{\mu} + ZDZ') \right) \]

\[ \equiv \text{MVN} \left( X\beta, W^{-1/2}RW^{-1/2} + ZDZ' \right) \]

where \( W = R^{-1}_\mu |g'(\hat{\mu})|^{-2} \).

Algorithms which execute PL (e.g. SAS procedure GLIMMIX) get estimates through a 2-step process:

1. Use approximated linear model to get estimates \( \beta \) and \( (R, D) \) and prediction of random effects \( u \).

2. Use these results to obtain a new iterate of \( V \) and \( \hat{\mu} \).
Then again we can get a new approximated linear model, and repeat above 2 steps until the estimates of $\beta$ and $(R, D)$ converges.
Chapter 3

Related Clustering Methods

3.1 Co-clustering

Co-clustering, also called biclustering, or bivariate clustering, is a technique that simultaneously groups the rows and columns of a matrix into co-clusters. Under the usual setting of co-clustering problems, each of \( n \) observations which are to be clustered is represented by a \( p \)-dimensional vector. The entire data set may then be represented by a \( n \times p \) matrix. Different from one-dimensional clustering methods that seek to identify similar rows and columns independently, co-clustering takes the advantage of dependencies by simultaneously clustering rows and columns [7]. The following is an example to conduct bivariate clustering of 50 basketball players on their record statistics such as field goals attempted, turnovers, etc. It uses dendrogram algorithm on a \( 50 \times 20 \) matrix where each element represents each one of 20 statistics of each of the 50 players. Dendrogram algorithm constructs hierarchical agglomerative clusters for rows and columns independently and simultaneously. As can be seen from the co-clustering result on the top left, the co-clusters together shows a clear pattern of certain players with similar skills.
Some other co-clustering method takes the dependency of the rows and columns into consideration. The block diagonal or checkerboard structures occur when rows and columns are divided into partitions. In a block diagonal co-cluster structure, each row and column belongs to exactly one co-cluster, and rearranging the rows and columns of the data matrix reveals the co-clusters on the diagonal. In the checkerboard case, each row belongs to all column clusters, and each column belongs to all row clusters. The co-clustering algorithms are not generalized until when it was proposed in [12] in which the co-clustering algorithm is based on variance, and has been applied to gene-data of biology. Later, the co-clustering algorithm proposed in [13] has been applied to the organizing the files and words. Two of the co-clustering algorithms were introduced in [13]. One is based on bipartite spectral graph partitioning, and the other is based on information theorem. However, the formulation of the co-clustering problems can be very complex, and for some of the algorithms mentioned above, it requires either large computational effort or find a more heuristic way to simplify the implementation.
As shown in earlier basketball player example, the common way of co-clustering is to build a dendrogram in which rows and columns are independently and simultaneously using hierarchical agglomerative clustering, and then the result is presented on a 'heat map'. In the IPM application, the data collected from an orchard is usually presented in a form of spatial lattice array when referring to Figure 1.1. Unlike other co-clustering applications, this application of co-clustering requires that the co-clusters consist of spatially consecutive rows and columns. To determine the existence of clusters on a spatial lattice, a checkerboard structure of co-clusters is more compatible. Here a model based co-clustering method is introduced to search for the optimal co-clustering designs [63]. The model proposed to calculate the proximity among the data points is generalized linear mixed model. At each step to determine a successful co-cluster, a maximized likelihood value is calculated, so it is essential to have a good numerical solution to the GLMM model fitting.

3.1.1 GLMM based Checkerboard Structure Co-clustering

Consider an $r \times c$ spatial lattice in which each lattice point is a potential sampling site. By simultaneously dividing rows and columns into a number of contiguous and disjoint groups, we obtain a checkerboard structure within the lattice. For a co-cluster that has $n$ groups of rows and $m$ groups of columns, we use the term “nomenclature” to refer to the $n \times m$ checkerboard structure. We use the term “design” to refer to the specific rows and columns within a given nomenclature. Note that the number of rows and columns in each nomenclature shall be determined through the search of optimal co-clustering algorithms [63].
The GLMM proposed for co-clustering according to a checkerboard structure is

\[ Y_{j(i)} \mid s \stackrel{ind}{\sim} \text{Negative Binomial}(\theta_i, \kappa), i = 1, 2, \ldots, nm; j = 1, 2, \ldots, n_i \]

\[ \log(\theta_i) = \mu + s_i \]

\[ s = (s_1, s_2, \ldots, s_{nm})' \sim (0, D) \]

where \( Y_{j(i)} \) is the count number from the \( j \)th lattice point in the \( i \)th co-cluster, \( n_i \) is the number of lattice points in the \( i \)th co-cluster, \( \theta_i \) is the conditional (on \( s_i \)) mean of counts associated with the \( i \)th co-cluster, \( \kappa \) represents an overdispersion parameter, \( \mu \) is a fixed intercept effect, and \( s_i \) is a random effect associated with the \( i \)th co-cluster, and 

\[ D = \sigma^2 \exp(-\frac{d_{ii}'}{\theta}) \]

is the exponential spatial covariance structure between \( i \)th and \( i' \)th co-clusters.

### 3.1.1.1 Global Optimization Algorithm

An optimal design is defined to be the one with the maximum log-likelihood among all the possible designs. To avoid co-clusters that are too small, a minimum co-cluster size of \( r_0 \times c_0 \) (\( r_0 > 1 \) and \( c_0 > 1 \)) is specified. A global optimization algorithm (GOA) would identify all possible designs for every possible nomenclature, and select the global optimal design as the one that maximizes \( l(\mu, \sigma^2, \kappa) \).

### 3.1.1.2 Heuristic Optimization Algorithm

It was illustrated in [63] that for a spatial lattice of size 80 \( \times \) 80 and a minimum co-cluster size of 12 \( \times \) 12, the number of designs that need to be examined when searching for the global optimal design is 382,241,601. The enormous number of candidates usually makes it infeasible to exhaustively search for the optimal design.
To circumvent the computational complexity associated with global optimization, it was proposed in [63] the following heuristic optimization algorithm (HOA):

1. Starting with the original spatial lattice, fit the GLMM to each of the designs associated with the $1 \times 2$ and $2 \times 1$ nomenclatures. Identify the design with the maximum $l\left(\hat{\mu}, \hat{\sigma}^2, \hat{\kappa}\right)$ as the "Current Optimal Design" and denote its log-likelihood by $l^*\left(\hat{\mu}, \hat{\sigma}^2, \hat{\kappa}\right)$.

2. Starting with the "Current Optimal Design", fit the GLMM to each of the designs with the nomenclature that has either one more row group or one more column group than the "Current Optimal Design." Identify the design with the maximum $l\left(\hat{\mu}, \hat{\sigma}^2, \hat{\kappa}\right)$ as the "Potential Optimal Design" and denote its log-likelihood is by $l^0\left(\hat{\mu}, \hat{\sigma}^2, \hat{\kappa}\right)$.

3. If $l^0\left(\hat{\mu}, \hat{\sigma}^2, \hat{\kappa}\right) > l^*\left(\hat{\mu}, \hat{\sigma}^2, \hat{\kappa}\right)$, replace the "Current Optimal Design" with the "Potential Optimal Design" and repeat Step 2; otherwise, stop the procedure and report the "Current Optimal Design" as the heuristic optimal design.

By implementing HOA to the selected spatial lattice, we obtain an optimal design of co-clustering and divide the lattice into co-clusters. To find the “hot spots” and make decision on whether or not to treat, an inference-based treatment decisions procedures will be discussed in the next section.

### 3.1.1.3 An Illustration to the IPM Applications

Consider the example shown in Figure 1.1, count data is collected from a spatial lattice of size $25 \times 34$. Figure 3.2 shows the co-clustering results from using HOA with the Negative Binomial GLMM defined earlier.
To further help pest management specialist to determine whether or not to apply the treatment on a co-cluster, an inference-based treatment decision process is introduced. Tradition IPM practices were based on hypothesis test:

$$H_0 : \theta \leq \theta_c \text{ vs. } H_a : \theta > \theta_c$$

where $\theta_c$ is a critical economic threshold for which the cost of treatment is equal to the cost of no treatment. Rejecting null hypothesis would call for treatment. However, here we use the model to predict its conditional mean for each of those found co-clusters:

$$\theta_i = \exp(\mu + s_i), \ i = 1, 2, \cdots, nm$$

It was shown in [63], the Best Linear Predictor (BLP) for $\theta_i$ is

$$\tilde{\theta}_i = \text{BLP} (\theta_i) = \frac{\exp(\mu + \sigma^2/2) (\exp(\sigma^2) - 1) \sum_{j=1}^{n_i} y_{j(i)} + \exp(2\mu + 2\sigma^2) / \kappa + \exp(\mu + \sigma^2 / 2)}{\exp(\mu + 3\sigma^2/2) / \kappa + 1 + n_i \exp(\mu + \sigma^2 / 2) (\exp(\sigma^2) - 1)}$$
and the variance of $\log \tilde{\theta}_i - \log \theta_i$ is

$$\text{Var} \left( \log \tilde{\theta}_i - \log \theta_i \right) \approx \left\{ n_i \exp(2\mu + \sigma^2)(\exp(\sigma^2) - 1)^2[\exp(\sigma^2/2 - \mu) + 1/\kappa] + \\
\exp(2\sigma^2)(\exp(\sigma^2) - 1)[\exp(\mu + 2\sigma^2/2)/\kappa + 1]^2 \right\} \\
\div \left[ \exp(\mu + 3\sigma^2/2)/\kappa + 1 + n_i \exp(\mu + \sigma^2/2)(\exp(\sigma^2) - 1) \right]^2.$$ 

By plugging the MLEs $\left( \hat{\mu}, \hat{\sigma}^2, \hat{\kappa} \right)$ in above equations, we get the empirical Best Linear Predictor (eBLP), say $\hat{\theta}_i$, and the estimated variance of $\log \tilde{\theta}_i - \log \theta_i$, say $\hat{\text{Var}} \left( \log \tilde{\theta}_i - \log \theta_i \right)$. Define

$$U_i = \frac{\left( \log \hat{\theta}_i - \log \theta_i \right)}{\sqrt{\hat{\text{Var}} \left( \log \tilde{\theta}_i - \log \theta_i \right)}}$$

and let $U_{i,\alpha}$ be the $100(1 - \alpha)^{\text{th}}$ conditional percentile of $U_i$ given $s$. Then a $100(1 - \alpha)\%$ lower conditional prediction bound for $\log \theta_i$ is

$$L_{\alpha} (\log \theta_i) = \log \hat{\theta}_i - U_{i,\alpha} \sqrt{\hat{\text{Var}} \left( \log \tilde{\theta}_i - \log \theta_i \right)}$$

A $100(1 - \alpha)\%$ lower conditional prediction bound for $\theta_i$ is

$$L_{\alpha} (\theta_i) = \frac{\hat{\theta}_i - U_{i,\alpha}}{\exp U_{i,\alpha}} \sqrt{\hat{\text{Var}} \left( \log \tilde{\theta}_i - \log \theta_i \right)}$$

For a pre-specified threshold $\theta_c$, the decision of "Treat" is made if $L_{\alpha} (\theta_i) > \theta_c$; otherwise the decision of "Do Not Treat" is made.

### 3.2 Bayesian Disease Mapping

The area of disease mapping has had a long history, but has been growing fast in dealing with geo-referenced data computationally since the great development of the
geographical information system (GIS). In the area of risk estimation and modeling, [2] first proposed to fit the Bayesian models with random effects using McMC, and since then a large use of such methods has increased. The use of scan statistics proposed by [31] for disease clustering has also become popular, and the scan method will be introduced later in this chapter. In this section, the Bayesian model for cluster detection is introduced.

3.2.1 Bayesian approach to Relative Risk Estimation

For each of the $m$ regions on the map, $y_i$ is the count of disease in the $i^{th}$ region, $e_i$ is the expected count in the $i^{th}$ region, and $\theta_i$ is the relative risk in the $i^{th}$ region, so the mean of disease count within $i^{th}$ region is represented using $e_i\theta_i$. Modern approaches to relative risk estimation rely on smoothing methods, and these methods often involve additional assumptions or model components, so here the Bayesian modeling is used to estimate the relative risk for given study region. One way to produce smoother relative risk estimators is to assume that the risk has a distribution, which is called a prior distribution in Bayesian terms.

The Poisson model is the most common model for small area count data, and this model can also deal with the case when there is a relatively low count of disease within a large amount of population. It is assumed that the count of disease in the $i^{th}$ region is $y_i$, and they are independently distributed as $y_i \sim Poi(\mu_i)$, then the likelihood is

$$L(y \mid \mu) = \prod_{i=1}^{m} \frac{\mu_i^{y_i} \exp(-\mu_i)}{y_i!}$$

To consider both the background population effect and relative risk component in the model, it is usually assumed that the data is independently distributed with
expectation $E(y_i) = \mu_i = e_i \theta_i$, where $e_i$ is the expected rate for the $i^{th}$ area and $\theta_i$ is the relative risk for the $i^{th}$ area. By developing a Bayesian hierarchical model, consider $\{y_i\}$ to be conditionally independent distributed given $\{\theta_i\}$. Usually the interest is to model the relative risk $\theta_i$ such that $log(\theta_i) = \alpha_0 + v_i + u_i$ adds spatially heterogeneity to the model. For example,

$$v \sim \text{MVN}(0, \sigma^2 I)$$

adds uncorrelated covariance component, and

$$u \sim \text{MVN} \left(0, \sigma^2 \exp(\sigma^2 \frac{d_{ij}}{\theta}) \right)$$

adds correlated spatial covariance structure to the model. This distribution is conditional autoregressive (CAR) and defines spatial correlation between two regions. Here we take the Improper CAR (ICAR) models for example.

First proposed by [2], the prior for correlated heterogeneity $u$ is defined as

$$p(u \mid \gamma) \propto \frac{1}{\gamma^{m/2}} \exp \left\{ -\frac{1}{2\gamma} \sum_i \sum_{j \in \delta_i} (u_i - u_j)^2 \right\}$$

where $\delta_i$ is a neighborhood of the $i^{th}$ region, and the neighborhood $\delta_i$ should be the adjacent neighbors only. However, more general weighting scheme could also be used such as the adjacent neighbors and nonadjacent defined by common boundary or by a distance cut-off are considered at the same time. The fixed effect $\alpha_0$ is assumed to have a uniformly distributed prior, i.e. $\alpha_0 \sim U(a, b)$. The uncorrelated heterogeneity $v$ is defined to have a conventional zero-mean Gaussian prior distribution:

$$p(v \mid \sigma) \propto \sigma^{-m/2} \exp \left\{ -\frac{1}{2\sigma} \sum_{i=1}^{m} v_i^2 \right\}$$
Both $\gamma$ and $\sigma$ have improper inverse exponential hyperpriors:

$$p(\gamma, \sigma) \propto e^{-\epsilon/2\gamma}e^{\epsilon/2\sigma}, \sigma, \gamma > 0$$

where $\epsilon$ is taken as 0.001, and alternative hyperpriors in gamma and inverse gamma family are also commonly used. Therefore, the full posterior distribution for the Poisson likelihood is given as follows:

$$
P(\alpha_0, u, v, \gamma, \sigma \ | \ y_i) = \prod_{i=1}^{m}\{\exp (-e_i\theta_i) (e_i\theta_i)^{y_i} / y_i!}\}
\times \frac{\alpha_0}{b - a}
\times \frac{1}{\pi^{m/2}} \exp\{-\frac{1}{2\gamma} \sum_{i} \sum_{j \in \delta_i} (u_i - u_j)^2\}
\times \sigma^{-m/2} \exp\{-\frac{1}{2\sigma} \sum_{i=1}^{m} v_i^2\}
\times \sigma^{-m/2} \exp\{-\frac{1}{2\sigma} \sum_{i=1}^{m} v_i^2\}
$$

This posterior distribution can be sampled using McMC algorithms such as the Gibbs or Metropolis-Hastings samplers.

### 3.2.2 Disease Cluster Detection

#### 3.2.2.1 Cluster Detection using Posterior Measures

One approach to define a cluster is the exceedance probability. Define exceedance probability as the probability that the relative risk $\theta$ exceeds some threshold $c$. From the posterior samples of $\theta_i$, we can compute the exceedance probability using

$$\hat{Pr} (\theta_i > c) = \sum_{g=1}^{G} I (\theta_i^g > c) / G$$
where $G$ is the number of samples, and

$$I(a) = \begin{cases} 
1 & \text{if } a \text{ true} \\
0 & \text{otherwise}
\end{cases}.$$

To display the cluster detected using the exceedance probability, a value $c$ has to be chosen as well as the threshold for the probability that is $b$ in $\hat{P}r(\theta_i > c) > b$, which is usually set to be at conventional level such as 0.95, 0.975, etc. There is a trade off between quantities $b$ and $c$, so one of them should be fixed before considering the other. One concern with using exceedance probability is that it only detects hot spot cluster which does not consider neighborhood information or possible additional clusters. In [25], it proposed a post hoc measure to include the neighborhood information when detecting the clusters. For neighborhood of the $i^{th}$ region $\delta_i$, the number of neighbors is $n_i$, and then

$$\bar{q}_i = \frac{\sum_{j=0}^{n_i} q_{ij}}{(n_i + 1)}$$

where $q_{ij} = Pr(\theta_j > c) \forall j \in \delta_i$ and $q_{i0} = Pr(\theta_i > c)$. Here $\bar{q}_i$ and $q_{i0}$ are used to detect other forms of clustering. Another concern with using exceedance probability is that the clustering measure depends on the output of a model, so if model is not fully capturing the cluster information, some clustering of interest is introduced in the residual noise. Lastly, the exceedance probability can be examined for other types of data and models, such as the intensity exceedance $Pr(\hat{X}_1(s_1) > 1)$ if it is for the case event data, and the exceedance if case probability $Pr(\hat{p}_i > 0.5)$ if it is for the binary or binomial data.
3.2.2.2 Cluster Detection using Residuals

When using residuals to define clusters, it is assumed that $y_i$ is the count of disease within the $i^{th}$ region and $E(y_i) = \mu_i = e_i\theta_i$. In the case of Poisson distributed data such that $y_i | \theta_i \sim Poi(e_i\theta_i)$, the relative risk $\theta_i$ can be modeled as

$$\log \theta_i = \alpha_0 + v_i + r_i$$

where $\alpha_0$ is the fixed effect, $v_i$ is the uncorrelated random effect, and $r_i$ is the residual which contains clustering information. To consider the residual purely associated with clustering, we can model the uncorrelated noise at observational level into $v_i$. If the clustering is more likely to be irregular such that the modeling effects are not confounding the clustering, then the residual is useful to detect clusters. However, we should always try to include the clustering components within the model if any prior information about the clusters is given. To fit the model, the prior distributions are assumed to be

$$\alpha_0 \sim U(a,b)$$

$$v_i \sim N(0, \tau_v)$$

with $\tau_v$ should be set large and $(a,b)$ should be in a wide range. Here we assume no correlated random effect and the clustering is to be found in the residuals. Bayesian residuals for this likelihood is standardized as

$$r_i = \frac{y_i - e_i\hat{\theta}_i}{\sqrt{e_i\hat{\theta}_i}}$$
where \( \hat{\theta}_i \) is the mean of the posterior distribution of the \( \theta_i \). To display the mapping of the cluster, for different cut-off \( c \), the average estimate of \( P (r_i > c) \) is colored in different levels.

Output of Bayesian disease mapping is a thematic map of exceedance probabilities for each of the assumed contiguous units which are usually represented by counties or states in traditional cases. For lattice applications such as count data from trees, we theoretically do not have contiguous units. Using BDM in these situations would result in a thematic map of the lattice points. Conceivably, a thematic map can be used to identify clusters. However, these clusters may not be "smooth" in the sense that there could be an isolated point or clusters of points that are barely separated and it might have been more sensible to join them into one cluster. There is no minimum size of clusters integrated into the solution this way. Moreover, the choice of exceedance probability threshold for identifying clusters is a required input that is an influential assumption compared to our proposed method. Finally, for subsampling on the spatial lattice, a tiling algorithm is necessary in order to define neighbors required by the conditional autoregressive model in BDM, and there are multiple ways to tile. To sum up, BDM is not an easy application to spatial lattice structured data.

### 3.3 Spatial Scan Statistics

#### 3.3.1 The Geographical Analysis Machine (GAM)

In [49], it examines the case counts within each of the proposed circles with variable sizes and determines potential clusterings based on the counts. The GAM procedure is described as follows:

1. Select a value for radius \( r \) of the searching circle.
2. Cover the study region using a fine spaced square lattice with grid points \( i = 1, 2, \cdots, K(r) \). The distance between two grid points is typically \( 1/5 \) or \( 1/10 \) of the radius \( r \).

3. For each of the \( K(r) \) grid point, compute the number of cases \( C_{ir} \) in a circle which is centered at \( i^{th} \) grid point with radius \( r \). If the observed \( C_{ir} \) exceeds the 99.8th percentile of the distribution of the number of cases found under null hypothesis, draw the corresponding circle on the map.

4. Return to step 1 and repeat the procedure for a higher value of \( r \).

Note that Monte Carlo simulation is used to obtain the 99.8th critical value for the test. This involves generating 499 replicated spatial point process under the null hypothesis, and for each replication, steps 1-4 is carried out. The result presents all significant circles on a map, and since the values of \( C_{ir} \) are correlated, there should be a lot of overlapping circles. Especially if the clusters are present in the data, the circles are bunched. Since the clusters are detected by overlapping circles using this method, the clusters identified are correlated, and the Bonferroni procedure could be too conservative to overcome the multiple comparison problem. However, later in [59], it addresses the multiple testing problem while identifying the clusters by constructing overlapping circles.

### 3.3.2 Cluster Evaluation Permutation Procedure (CEPP)

The cluster evaluation permutation procedure (CEPP) by [59] is not only used to identify the presence of clusters but also assess the significance of the detected clusters. The procedure is as follows:

1. First divide the study region into \( I \) cells, usually census tracts, and the distance between two cells are measured between two geographic centroids of the two.
Denote the number of population in \( i \)th cell \( n_i \), and the number of cases within that cell is \( C_i \). Note that \( \sum_i n_i = n \) and \( \sum_i C_i = C \) where \( n \) is the total number of population and \( C \) is the total number of cases in the study region.

2. For each cell, a two-dimensional window in which it contains a fixed number of population of \( R \) is constructed. More specifically, if the cell \( i \) has population \( n_i \) and \( n_i < R \), then we look at cell \( j \) whose centroid is closest to cell \( i \). If \( n_i + n_j = R \), then the window is formed with cell \( i \) and \( j \); if \( n_i + n_j > R \), then a fraction \( (R - n_i)/n_j \) is taken to be the window; if \( n_i + n_j < R \), then we continue to examine the neighboring cells of the \( i \)th cell until the window has one of previous two forms. The value \( R \) should be chosen as a sufficiently large number so that we rarely have the situation of \( n_i > R \).

3. Calculate the total number of cases \( C_{iR} \) within the constructed window. Since each window has same population, \( C_{iR} \) can also be viewed as the rate of the cases.

4. Under the null hypothesis, \( \{C_{iR} : i = 1, 2, \cdots, I\} \) are identically distributed random variables, and the circle with the maximum incident rate

\[
M_R = \max(C_{1R}, C_{2R}, \cdots, C_{IR})/R
\]

is picked and the significance of the circle is determined by test statistic \( M_R \). The null distribution of \( M_R \) is derived using a randomization test described by [18]. Randomly assign \( C \) cases among \( n \) population and calculate \( M_R \) for each of the Monte Carlo samples of the \( n!/ (C!(n-C)!) \) permutation. A cut-off value of \( K \) is determined from the collection of \( M_R \), and if the observed \( M_R \) is greater than \( K \), then the null hypothesis is rejected.
Compared to GAM, CEPP tests against a single composite alternative that there exist clusters among those circular zones having population $R$. However, the choice of $R$ is not arbitrary and it was suggested by [59] to experiment different choices of $R$. By experimenting different values of $R$, it brings back the multiple testing problem since the tests are highly correlated for different choices of $R$.

3.3.3 Kulldorff’s Scan Statistics (KSS)

Kulldorff’s Scan Statistic (KSS) [29], [31] is commonly used to detect clusters in a geographical space for disease mapping through use of scan zones. Other applications using scan statistics include detecting local anomalies in time space computer networks [48]. Scan zones take on a selected shape such as a circle or ellipse, and the size of the zones are allowed to vary as they scan the study region. Here, the discussion is based on the most commonly used circular scan zone.

Let $G$ denote the entire study region, and assume it can be partitioned into $K$ strata, and each stratum is represented by aggregated data. Centering at the centroid of a stratum, the scan zone moves over the study region and defines a collection of zones $Z \subset G$. At each location, the size of the scan zones expands by adding one more neighboring centroid of the $K$ strata up to where it contains half of the population in $G$ [31]. To illustrate the number of potential zones to be examined, suppose there are 50 geographical strata (e.g. states), for a population that is uniformly distributed, the total number of potential zones we need to examine is on the order of $(0.5 \times 50) \times 50$, or 1,250. Alternatively, if the roughly 3000 U.S. counties are defined as strata, the number of potential zones to examine is about 4.5 million.

Each data point from a stratum is considered a sum of independent binary responses over the entire population strata: "disease" or "no disease", and is thus a...
binomial random variable. Suppose the population in the entire study region is \( n_G \). For a particular zone \( Z \), the population is \( n_Z \). The observed number of "diseased" people in \( Z \) is \( Y_1 \), and the observed number of "diseased" people not in \( Z \) is \( Y_2 \). The probability of a randomly selected person with the "disease" in \( Z \) is \( p_1 \), and the probability of a randomly selected person with the "disease" not in \( Z \) is \( p_2 \). The binomial model stipulates that

\[
Y_1 \sim \text{Bin}(n_Z, p_1) \\
Y_2 \sim \text{Bin}(n_G - n_Z, p_2).
\]

The null hypothesis is \( H_0 : p_1 = p_2 \), and the alternative hypothesis is \( H_1 : p_1 > p_2 \). When the number of "diseased" people is far less than the number of population in the study region, the Poisson model is used to give a close approximation,

\[
Y_1 \sim \text{Poi}(p_1 n_Z) \\
Y_2 \sim \text{Poi}(p_2 (n_G - n_Z)).
\]

For each proposed zone \( Z \in \mathcal{Z} \), the likelihood ratio \( \lambda = \frac{\max_{H_0 \cup H_1} L(p_1, p_2)}{\max_{H_0} L(p_1, p_2)} \) based on the Poisson model is

\[
\lambda = \begin{cases} 
\left( \frac{y_1}{n_Z} \right)^{y_1} \left( \frac{y_2}{n_G - n_Z} \right)^{y_2} & \text{if } \frac{y_1}{n_Z} > \frac{y_2}{(n_G - n_Z)} \\
1 & \text{otherwise}
\end{cases}
\]

(See Appendix 5.1 for the derivation.) The scan zone with the largest likelihood ratio among all of zones is called the "most likely cluster". Denote the likelihood ratio associated with the "most likely cluster" by \( \lambda^* = \sup_{Z \in \mathcal{Z}} \lambda \).
Since the null distribution of \( \lambda^* \) has no simple analytical form, Monte Carlo simulation is used to sample from the exact distribution. The Monte Carlo simulation permutes the data points over all of the sampling locations to create 1000 replicates of the data set. For each replicate, a \( \lambda^* \) is calculated. The values are sorted from large to small. The 'most likely cluster' is significant if observed \( \lambda^* \) is among the 50 highest of those sorted values. For additional non-overlapping zones, we compare their likelihood ratio values to those same sorted values. The non-overlapping zones with likelihood ratio values placed among the top 50 sorted values are further identified as 'secondary clusters'.

The following drawbacks can be seen when using KSS:

1. It is a computational compromise to use the same reference distribution of the Monte Carlo replicates of \( \lambda^* \) created for making inference about 'most likely cluster' when trying to decide if 'secondary clusters' exist.

2. The SaTScan software [32] was developed to implement KSS with circular or elliptical scan zones. However, when the shape of scan zone changes to rectangles, as is more natural for use on lattice arrays, the number of potential zones to examine increases dramatically. Let the minimum size of a scan zone be \( m_0 \times n_0 \), then the maximum number of rectangular zones to be examined is

\[
N = \sum_{i=m_0}^{r} \sum_{j=n_0}^{c} (r - i + 1) (c - j + 1)
\]

(3.2)

To illustrate, for a lattice of size 100 \( \times \) 100 and a minimum scan zone size of 5 \( \times \) 5, the number of possible combinations we have to examine is over 20 millions. The
number of circular scan zone to be examined on a lattice is not more than 1.4 million.

3. The computational challenge does not only arise when scanning for the 'most likely cluster', but also when trying to identify 'secondary clusters.' In order to find secondary significant clusters, we need to identify the non-overlapping zones among those having the top 50 highest likelihood ratio values.

4. KSS was designed for use with independent data and its performance with correlated count data has not been studied.

3.3.4 Sequential Approach to Adjust for Multiple Clusters

From previously summarized limitations of KSS, we can see that the secondary clusters identifications seems to be a problem of interest. The likelihood ratio test statistic from KSS uses the cases within the searching zone against the remaining cases outside the zone, so when calculating the likelihood ratio test statistic for less likely clusters, the statistic will be affected by previously detected most likely clusters. The p-values for the less likely clusters are conservatively high given the presence of significantly detected most likely clusters, so the less likely clusters often end up being missed because of the non significant p-values. A sequential version of the spatial scan statistic which adjusts for the presence of other clusters than the most likely clusters is proposed by [64] to eliminate this shadow effect of the most likely clusters on other potential less likely clusters. At each of the sequential steps, the usual spatial scan statistics to find the mostly likely cluster is carried out on the reduced data from deleting previously found significant clusters, the procedure is iterated until no more significant cluster is found. In order to test the statistical significance of multiple clusters sequentially, test the second most
likely cluster only if the most likely cluster is significant, and the third most likely cluster only if the second most likely cluster is significant and so on. By doing so, it removes the effect due to the more likely clusters on less significant clusters. This sequential approach has ability to detect and test the significance of secondary clusters which are not overlapping with the primary cluster. By deleting the effect of primary cluster, the p-value associated with the secondary clusters are more accurately representing the true significance level of the secondary cluster. Since the p-values for testing for a secondary cluster are calculated conditionally on ignoring the primary cluster, the p-values are conservative than those by the standard method, and it could result in loss of power [30].

A benchmark data set described in detail in [33] is used to evaluate the adjusted spatial scan statistic. The data set simulates 600 disease cases and distributed among 1990 female population within 245 counties in the northeastern United States. Different sizes of true clusters, comprising 1, 4, or 16 counties are generated. The simulated disease counts have a multinomial distribution with probabilities proportional to \( nr \), where \( n \) is the number of population and \( r \) is the relative risk of the disease within each county. The hypothesis test for the clustering is \( H_0 : r = 1 \) vs. \( H_a : r > 1 \). The type I error here is defined as when there is only one true cluster simulated, the probability of detecting two or more clusters using this method. The power is defined as when there are two clusters being generated, the probability of detecting one or less. Each calculation of type I error is based on 10000 simulated data sets, and each power calculation is based on 1000 simulated data sets. Overall, it shows that the type I error rate of this method is very close to the nominal levels and the power of sequential approach is in general higher than traditional KSS especially for larger cluster sizes.
3.3.5 A multiple Cluster Detection Algorithm

The sequential version of the scan statistic shows better power for detecting the second weaker cluster, but did not improve the ability of detecting the most likely cluster. In [38], a new extension of the spatial scan statistic which could be used to detect multiple clusters through constructing two or more clusters in the alternative hypothesis is proposed. The performance of the proposed method is compared to the sequential method through an intensive simulation study, and it shows better power in terms of both rejecting the null hypothesis and accurately detecting the coexisting clusters.

The multiple cluster detection algorithm considers two-cluster at the same time when conducting the scan searches. Suppose $Z_i$ and $Z_j$ are two non-overlapping candidate single scan zones described in KSS. If the total population within these two zones does not exceed a predefined percentage of the total population in the study region, the combination of $(Z_i, Z_j)$ is taken as a candidate two-cluster scan zone to detect two clusters at the same time. Suppose the disease counts within $Z_i, Z_j$ and background are $c_1, c_2, c_3$, and $c_1 \sim \text{Poi}(m_1 p_1)$, $c_2 \sim \text{Poi}(m_2 p_2)$, and $c_3 \sim \text{Poi}(m_3 p_3)$, where $m_1, m_2, m_3$ are the number of population and $p_1, p_2, p_3$ are the relative risk within each of the scan zones and background. Therefore, the likelihood function is

$$L((Z_i, Z_j), p_1, p_2, p_3) = \frac{e^{-p_1 m_1} (p_1 m_1)^{c_1}}{c_1!} \frac{e^{-p_2 m_2} (p_2 m_2)^{c_2}}{c_2!} \frac{e^{-p_3 m_3} (p_3 m_3)^{c_3}}{c_3!}$$

and then the likelihood ratio test statistic is

$$\lambda = \max_{(Z_i, Z_j)} \frac{\max_{H_0: p_1 = p_3} L((Z_i, Z_j), p_1, p_2, p_3)}{\max_{H_0: p_2 = p_3} L((Z_i, Z_j), p_1, p_2, p_3)}$$
Under the null hypothesis, the MLEs of the parameters are

\[
\hat{p}_1 = \hat{p}_2 = \hat{p}_3 = \frac{c_1 + c_2 + c_3}{m_1 + m_2 + m_3} = \frac{N}{M}
\]

and under the alternative hypothesis, the MLEs of the parameters are

\[
\hat{p}_1 = \frac{c_1}{m_1}, \quad \hat{p}_2 = \frac{c_2}{m_2}, \quad \hat{p}_3 = \frac{c_3}{m_3}
\]

Thus, the likelihood ratio test statistic is

\[
\lambda = \max_{(Z_i, Z_j)} \left( \frac{c_1}{m_1} \right)^{c_1} \left( \frac{c_2}{m_2} \right)^{c_2} \left( \frac{c_3}{m_3} \right)^{c_3} \left( \frac{N}{M} \right)^N \times I \left( \frac{c_1}{m_1} > \frac{c_3}{m_3} \text{ and } \frac{c_1}{m_1} > \frac{c_3}{m_3} \right)
\]

For each proposed set of non-overlapping zones, this test statistic is evaluated, and the significance level is determined through Monte Carlo simulation as shown in KSS. Depending on how many single clusters we want to detect through a one-time spatial scan, we can change the alternative hypothesis and likelihood ratio test statistic formulations. Here the two-cluster version is discussed, and the three and more clusters versions can be constructed in a similar way.

Similar to previously mentioned power evaluation for the sequential approach, the same simulated data sets were used to study the power of the two-cluster method. With one true cluster simulated, both of the sequential approach and two-cluster method perform well in terms of the power of rejecting the null hypothesis, and the sequential method is superior. With two true clusters simulated, the two-cluster method performs
better than the sequential method (about 8% better) in terms of the power of rejecting
the null hypothesis. This means that about 8% of the most likely cluster are shadowed
by the second likely cluster. With three true clusters simulated, the two-cluster method
is better (17% better) than the sequential method since there are more true clusters
coexisting in the study area, it is more difficult to reject the null hypothesis when
testing for most likely cluster. The most likely cluster’s shadowing effect on the second
likely cluster can be eliminated by the sequential method, but the second likely cluster’s
shadowing effect on the most likely cluster still exists. If there exists a second cluster,
it shadows the effect of the most likely clusters so that the p-value of the most likely
cluster is higher than it should be. This multiple cluster detection approach reduces the
potential for a situation when inference about a candidate primary cluster is impaired
by an additional cluster in the background. However, this approach becomes infeasible
to use when considering three or more clusters simultaneously.

3.4 SADIE Red-Blue Plot

The analysis of spatial pattern has developed largely independently in plant
and animal ecology. Often times, the ecological data are spatially-referenced such that
the data points are related directly to their locations usually on a two-dimensional space.
Ecologists are interested in testing for randomness and quantifying spatial patterns.

For spatial count data, spatial pattern is the non-random distribution of the
counts over an area, and the pattern is usually found in the form of clusters, when large
patches of relatively large counts occur close together, and gaps when large areas have
relatively small counts, often zero. The SADIE (Spatial Analysis by Distance IndIcEs)
method studies clustering in the spatial count data. The index quantifies the 'effort’
to redistribute counts over all of the sampling units in the study region so that each sampling unit has equal counts [51], and the red-blue plots present the index of clustering with graphical displays.

3.4.1 Distance to Regularity

This 'effort' is measured in 'distance to regularity' $D$, which is the minimum distance required to move to complete regularity. The value $D$ is provided by the Transportation algorithm [28]. The larger value of $D$, the more spatially aggregated the counts are. Define a donor unit as a sampling unit having counts greater than the mean counts of all the sampling units, and a receiver unit as a sampling unit having counts less than the mean counts of all the sampling units. Suppose there are $m$ donor units and $n$ receiver units, then the transportation algorithm [19] identifies $m \times n$ routes and associated payloads from the donor to the receiver units, which minimizes the total cost of transportation.

Given $m$ sources and $n$ destinations, the supply at source $i$ is $a_i$ and the demand at destination $j$ is $b_j$. The cost of transporting one unit of goods from source $i$ to destination $j$ is $c_{ij}$, and the amount of goods transported from source $i$ to destination $j$ is $x_{ij}$. The goal here is to minimize the total transportation cost $\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$ while satisfying all the supply and demand constraints. The constraint for supply is that for each source node $i$, $\sum_{j=1}^{n} x_{ij} = a_i$, and the demand constraint is for each destination node $j$, $\sum_{j=1}^{n} x_{ij} = b_j$. The general transportation model is

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

such that
\[
\sum_{j=1}^{n} x_{ij} = a_i, \quad i = 1, \ldots, m
\]
\[
\sum_{i=1}^{m} x_{ij} = b_j, \quad j = 1, \ldots, n
\]
\[
x_{ij} \geq 0, \quad i = 1, \ldots, m; \quad j = 1, \ldots, n
\]

Here, we assume the transportation model is balanced, i.e. the total demand equals the total supply
\[
\sum_{j=1}^{n} x_{ij} = a_i = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} = \sum_{i=1}^{m} x_{ij} = b_j,
\]

The transportation algorithm consists the following steps:

1. select an initial basic feasible solution. Several possible algorithms are available, but here we introduce the Northwest-corner method. start with the first supplier and go along the supplier list, and for each supplier:

   (a) Let the supplier allocate all of its resources to the demand centers going along the demand list.

   (b) Allocation should satisfy the demand of each center until the supplier’s resources are exhausted.

2. Following the preceding solution, now with the initial basic feasible solution, the typical simplex method is used to

   • Compute the reduced costs of the nonbasic variables.

   • Look at the reduced cost values and check the optimality, i.e. the reduced cost is zero for the nonbasic variables.
• If not optimal, i.e. the reduced cost is positive for the nonbasic variables, change the basis and update the basic feasible solution.

3.4.2 SADIE clustering

For a donor unit \( i \) at position \((x_i, y_i)\), the outflow of the \( j^{th} \) of \( n_i \) receiver units, \( j = 1, \cdots, n_i \) at position \((x_j, y_j)\) is denoted \( v_{ij} \). The distance of this flow is 
\[
[(x_i - x_j)^2 + (y_i - y_j)^2]^{\frac{1}{2}}
\]
and is denoted as \( d_{ij} \). The average distance of outflow from unit \( i \), weighted by the magnitude of each individual flow, is
\[
Y_i = \frac{\sum_{j=1}^{n_i} d_{ij} v_{ij}}{\sum_{j=1}^{n_i} v_{ij}}
\]

Each receiver unit can also be attributed a value of \( Y \) but the distance from a receiver to each of its donor is taken as negative. Units within groups of relatively large counts close to one another (i.e. units within a patch) will have relatively large values, compared to a unit with a large isolated count. Units within groups of relatively small counts close to one another (i.e. units within gaps) will have relatively large negative values. Hence, clustering is indicated by clumps of unexpected large values of \( Y_i \) or of \( |Y_i| \).

Since \( Y_i \) depends on coordinate system units. A scaling step that aims to produce a dimensionless index of clustering is carried out.

1. Randomly shuffle the cells (corresponding to no spatial association), and follow the count of each cell in each randomization and compute
   \[
c_Y = \text{average weighted distance formula value, where } c \text{ is for count as in following the count to all units it goes to.}
   \]

2. Randomly shuffle the cells, and keep the location of a particular count fixed and for each randomization, compute
\( iY \) = average weighted distance formula value, where \( i \) is for unit as in fixing the \( i^{th} \) unit and looking over all counts that come to it.

3. Compute \( 0Y \) = average absolute value of \( iY \) over all the units (which is equal to average absolute value of \( cY \) over all the units).

4. Finally, compute \( vi = \frac{Y0Y}{iY} \), which is a standardized and dimensionless index of clustering.

These indices are further classified by a heuristic threshold \( \pm 1.5 \) which was suggested by [52] to identify hot spots. The result is then presented as a 'red-blue' graphical display where 'red' shows the hot spots comprising larger counts, i.e. the sampling units having large indices \( vi > 1.5 \), 'blue' shows cold spots with smaller counts i.e. the sampling units having small indices \( vi < -1.5 \), and areas with no specific color representing sampling units with indices \(-1.5 < vi < 1.5\).

The disadvantages using the SADIE method can be summarized as follows:

1. The irregular shape of the red hot spots may not be convenient to work with in lattice type applications.

2. SRB works best with exhaustive sampling on a spatial lattice. Subsampling introduces ambiguity through the need to define an appropriate level of aggregation.

3. This method provides a visualization of where the hot spots are, but does not report a magnitude of the counts within each potential hot spot. SADIE provides a permutation test of randomness for the global hot spot instead of reporting statistical inference associated with individual hot spots.
Chapter 4

Identification of Hot Spots for Count Data on Spatial Lattice Arrays

4.1 PROPOSED METHOD

4.1.1 Hot Spot Identification

4.1.1.1 Model Formulation

To search for the first spot on an $r \times c$ spatial lattice, we define scan windows by moving a $m \times n$ rectangle through all possible locations in the spatial lattice. Let $m_0 \times n_0$ be the smallest window size considered during the scanning process. Due to the limited resources, exhaustive sampling is not always feasible, so sometimes the spatial lattice is subsampled at a rate of $p$. Simple random sampling could lead to samples that are not representative of the entire spatial lattice. To ensure that every proposed window has at least one sampled lattice point inside, a shifted sampling strategy such
as what was suggested by Zhang et al [63] should be employed. Figures 1.2 and 1.3 were subsampled using the shifted scheme where instead of sampling lattice points every 3 rows and 3 columns, the lattice points are sampled one row shifted down along the column direction and one column shifted to the right along the row direction.

The choice of \((m_0, n_0)\) should be different for exhaustive sampling and subsampling situations. For exhaustive sampling situation, it would be reasonable to choose \((m_0, n_0)\) equal to the smallest useful window to practitioners, say \((r_0, c_0)\). However, in subsampling situations this choice might yield scan windows that do not contain any sampled points. To overcome this difficulty we set the minimum size of the scan window to be larger than \((r_0, c_0)\) in subsampling situations. If the lattice is subsampled with a fraction \(p\), then the total number of sampled points is \(r \times c \times p\), and there are approximately \(\sqrt{r c p}\) lattice points sampled in each row and column, thus lattice points are sampled approximately every \(\frac{r}{\sqrt{r c p}}\) row and \(\frac{c}{\sqrt{r c p}}\) column. Therefore, we should choose \((m_0, n_0)\) equal to \((\frac{r}{\sqrt{r c p}}, \frac{c}{\sqrt{r c p}})\).

For a fixed \((m, n)\) where \(m \geq m_0\) and \(n \geq n_0\), let \(W(m, n)\) be a set of all windows of dimension \(m \times n\), and let \(w(m, n) \in W(m, n)\) be a particular window, and \(w^c(m, n)\) denote its complement. The number of sampling units in \(w(m, n)\) will be denoted by \(|w(m, n)|\). Let \(\{Y_j\}_{j=1}^{w(m,n)}\) denote the counts on the sampling units in \(w(m, n)\). Counts in \(w(m, n)\) are assumed to follow a conditional Poisson distribution with mean \(S_1\) where \(S_1 \sim exp(\alpha_1)\). Counts in \(w^c(m, n)\) are assumed to follow a conditional Poisson distribution with mean \(S_2\) where \(S_2 \sim exp(\alpha_2)\). Here, \(\alpha_1\) and \(\alpha_2\) denote the rate parameters of the respective exponential distributions. The one-sided hypothesis test for hot spot detection is \(H_0 : \alpha_1 = \alpha_2 = \alpha_0\), versus \(H_a : \alpha_1 < \alpha_2\), and for cold spot detection is \(H_0 : \alpha_1 = \alpha_2 = \alpha_0\), versus \(H_a : \alpha_1 > \alpha_2\). The likelihood ratio is calculated for each proposed window, and the searching window with the largest likelihood ratio
is identified as a hot spot at each iteration of search. In this paper, we mainly focus on finding only one type of either hot or cold spot on the spatial lattice. However, a two-sided hypothesis test such as \( H_0 : \alpha_1 = \alpha_2 = \alpha_0 \), versus \( H_a : \alpha_1 \neq \alpha_2 \) can be used to simultaneously detect hot or cold spot on the same spatial lattice.

Note that one could choose a variety of alternatives to this particular GLMM without overly complicating to the proposed procedure. For example, some applications might warrant the inclusion of fixed effects and/or additional random effects that capture additional variability in the data.

4.1.1.2 Likelihood

Let \( y_j \) denote the observed count on a sampling unit, then the marginal likelihood of the data corresponding to a particular window under the full parameter space is

\[
L(\alpha_1, \alpha_2, m, n, w(m, n)) = \int_0^\infty \int_0^\infty \prod_{y_j \in w(m, n)} e^{-s_1 s_1 y_j} \frac{y_j!}{\alpha_1 e^{-\alpha_1 s_1}} \prod_{y_j \in w_c(m, n)} e^{-s_2 s_2 y_j} \frac{y_j!}{\alpha_2 e^{-\alpha_2 s_2}} ds_1 ds_2
\]

\[
= \frac{\alpha_1 \alpha_2}{\prod_{y_j \in w(m, n)} y_j^{1} \prod_{y_j \in w_c(m, n)} y_j^{1}} \frac{\Gamma \left( \sum_{y_j \in w(m, n)} y_j + 1 \right)}{\Gamma \left( \sum_{y_j \in w_c(m, n)} y_j + 1 \right)} \frac{\sum_{y_j \in w(m, n)} y_j + 1}{\sum_{y_j \in w_c(m, n)} y_j + 1} \tag{4.1}
\]

The likelihood under the null hypothesis is
\[
L(\alpha_0, m, n, w(m,n)) = \frac{\alpha_0^2}{\prod_{y_j \in w(m,n)} y_j! \prod_{y_j \in w^c(m,n)} y_j!} \frac{\Gamma \left( \sum_{y_j \in w(m,n)} y_j + 1 \right)}{\left( |w(m,n)| + \alpha_0 \right)^{\sum_{y_j \in w^c(m,n)} y_j + 1}} \frac{\Gamma \left( \sum_{y_j \in w^c(m,n)} y_j + 1 \right)}{\left( |w^c(m,n)| + \alpha_0 \right)^{\sum_{y_j \in w^c(m,n)} y_j + 1}}
\]

(4.2)

The primary hot spot, \( \hat{w}(\hat{m}, \hat{n}) \), has the maximum likelihood ratio overall possible searching windows

\[
\lambda(\hat{\alpha}_1, \hat{\alpha}_2, \hat{\alpha}_0, \hat{m}, \hat{n}, \hat{w}(\hat{m}, \hat{n})) = \max \frac{\arg \max_{m \geq m_0, n \geq n_0, \alpha_1 \leq \alpha_2, w(m,n)} L(\alpha_1, \alpha_2, m, n, w(m,n))}{\arg \max_{m \geq m_0, n \geq n_0, \alpha_0, w(m,n)} L(\alpha_0, m, n, w(m,n))}
\]

The primary cold spot, \( \hat{w}(\hat{m}, \hat{n}) \), has the maximum likelihood ratio

\[
\lambda(\hat{\alpha}_1, \hat{\alpha}_2, \hat{\alpha}_0, \hat{m}, \hat{n}, \hat{w}(\hat{m}, \hat{n})) = \max \frac{\arg \max_{m \geq m_0, n \geq n_0, \alpha_1 \geq \alpha_2, w(m,n)} L(\alpha_1, \alpha_2, m, n, w(m,n))}{\arg \max_{m \geq m_0, n \geq n_0, \alpha_0, w(m,n)} L(\alpha_0, m, n, w(m,n))}
\]

4.1.1.3 Confirmation of Primary Hot Spot

Once a candidate primary hot spot is found, we seek to determine if it is statistically different from the neighboring region around it. To that end, we define a border region that circumscribes the candidate hot spot. The border is characterized by a width of rows and columns. Figure 1.4 illustrates the concept of a border region. Depending on the position of a candidate hot spot within the lattice, the border may not be always at its maximum width.
Consider a $m \times n$ hot spot, and let $k$ denote the to-be-determined width of the border. The ratio of the number of lattice points in the border region to the number of lattice points in the candidate hot spot is $(m + 2k)(n + 2k)/mn - 1$. Table 4.1 illustrates the value of this ratio for a square candidate hot spot (i.e. $m = n$).

From Table 4.1, a border width of 1 gives an approximately equal number of lattice points in the border as in a $5 \times 5$ potential hot spot, a border width of 2 gives an approximately equal number of points in the border as in a $10 \times 10$ hot spot, and a border width of 3 gives an approximately equal number of points in the border as in a $15 \times 15$ hot spot. As the size of hot spots increases, the border width should increase accordingly in order for the border to include approximately same amount of lattice points as that is in the hot spot. However, we do not want to have a border which is too wide because a wide border may include data from other potential hot spots. Therefore, a border width of 3 is recommended if the candidate hot spot size is found to be greater than...
than $10 \times 10$. If the data is subsampled, a constant border width of 3 is recommend to ensure there will be enough data points in the border.

Let $\{Y_{1j}\}_{j=1}^{n_1}$ and $\{Y_{2j}\}_{j=1}^{n_2}$ denote the counts on the sampling units in the candidate hot spot and the border region, respectively. Conditional on $S_1 = s_1$ and $S_2 = s_2$, $\{Y_{1j}\}_{j=1}^{n_1} \sim \text{Poi}(s_1)$ and $\{Y_{2j}\}_{j=1}^{n_2} \sim \text{Poi}(s_2)$. To test if the candidate is a hot spot, we test $H_0 : s_1 \leq s_2$, versus $H_a : s_1 > s_2$. To test if the candidate is a cold spot, the null hypothesis is $H_0 : s_1 \geq s_2$, and the alternative hypothesis is $H_a : s_1 < s_2$.

Let $U_1 = \sum_{j=1}^{n_1} Y_{1j}$ and $U_2 = \sum_{j=1}^{n_2} Y_{2j}$, then under the null hypothesis, the conditional distribution of $U_1 = u_1 \mid u \sim \text{Bin} \left( u, \frac{n_1}{n_1+n_2} \right)$, where $u = u_1 + u_2$. (See Appendix B for derivation) The null hypothesis $H_0 : s_1 \leq s_2$ should be rejected if $U_1$ exceeds the upper $\alpha$ percentile of $\text{Bin} \left( u, \frac{n_1}{n_1+n_2} \right)$ distribution. In this case, the conditional mean of the candidate hot spot is declared greater than the border and the candidate is then confirmed to be a hot spot. The null hypothesis $H_0 : s_1 \geq s_2$ should be rejected if $U_1$ is less than the lower $\alpha$ percentile of $\text{Bin} \left( u, \frac{n_1}{n_1+n_2} \right)$ distribution. In this case, the conditional mean of the candidate cold spot is declared less than the border and the candidate is confirmed as a cold spot. If the primary hot (cold) spot is confirmed, the process moves on to identify secondary hot spots. Otherwise, the process stops.

### 4.1.1.4 Secondary Hot Spots

If a primary hot spot is confirmed based on the border test, an iterative process is used to further search for secondary hot spots. Suggested by [64], each confirmed hot spot (based on the border test) is removed from the spatial lattice, and the remaining spatial lattice is searched through in the same way as was done to identify the primary hot spot. However, windows $w(m,n)$ that overlap with deleted lattice points are discarded from the search. The rationale here is that if a secondary hot spot overlaps with the
deleted area, they will be found through multiple searches in the iterative process. The search process terminates the moment a candidate secondary hot spot is not determined significant from the border test.

### 4.1.2 Search Algorithms

#### 4.1.2.1 Global Search (MBSM-G)

1. Start with the minimum window size $m_0 \times n_0$ located at the lower left corner of the lattice. Sequentially expand this window one row or one column at a time until the size of the window reaches a specified proportion of the size of the lattice. Expanding the searching window to the full size of the lattice is not usually considered because by definition a hot spot is a concentrated localized area.

2. For each window $w(m, n)$ generated in step 1, the conditional likelihood ratio, denoted by $\lambda_{c}^{(MBSM-G)}(\alpha_1, \alpha_2)$ is (3) with $m, n$ and $w(m, n)$ fixed. Evaluate the conditional likelihood score for $w(m, n)$ as

$$S[w(m, n)] = \frac{\text{argmax}_{\alpha_1 \leq \alpha_2} L(\alpha_1, \alpha_2)}{\text{argmax}_{\alpha_0} L(\alpha_0)}$$

when searching for a hot spot, and

$$S[w(m, n)] = \frac{\text{argmax}_{\alpha_1 \geq \alpha_2} L(\alpha_1, \alpha_2)}{\text{argmax}_{\alpha_0} L(\alpha_0)}$$

when searching for a cold spot.

3. Find $\hat{w}(\hat{m}, \hat{n}) = \arg\max_{m \geq m_0, n \geq n_0, w(m, n)} S[w(m, n)]$.  

---

57
4. Perform the border test on window \( \hat{w}(\hat{m}, \hat{n}) \) to determine if it is a hot spot. If the border test is significant, proceed to search for secondary hot spots. Otherwise, terminate the search.

4.1.2.2 Heuristic Search (MBSM-H)

1. MBSM-H differs from MBSM-G in that the window size is not varied. While the window continues to move around the lattice, it is a fixed size \( m_0 \times n_0 \).

2. For each window \( w(m_0, n_0) \) generated in step 1, the conditional likelihood, denoted by \( L^{(\text{MBSM-H})}(\alpha_1, \alpha_2) \), is (3) with \( (m, n) \) replaced by \( (m_0, n_0) \) and \( w(m, n) \) replaced by \( w(m_0, n_0) \). Evaluate the conditional likelihood score for \( w(m_0, n_0) \) as

\[
S[w(m_0, n_0)] = \frac{\text{argmax}_{\alpha_1 \leq \alpha_2} L(\alpha_1, \alpha_2)}{\text{argmax}_{\alpha_0} L(\alpha_0)}
\]

when searching for a hot spot, and

\[
S[w(m_0, n_0)] = \frac{\text{argmax}_{\alpha_1 \geq \alpha_2} L(\alpha_1, \alpha_2)}{\text{argmax}_{\alpha_0} L(\alpha_0)}
\]

when searching for a cold spot.

3. Find \( \hat{w}(m_0, n_0) = \text{argmax}_{w(m_0, n_0)} S[w(m_0, n_0)] \).

4. Perform the border test on window \( \hat{w}(m_0, n_0) \) to determine if it is a hot spot. If the border test is significant, proceed to search for the secondary hot spots. Otherwise, terminate the search.
4.2 PERFORMANCE ANALYSIS

4.2.1 Simulated Data Sets

In this section, six examples are discussed, and in each example, a $100 \times 100$ spatial lattice is used. With exhaustive sampling, each lattice point on the spatial lattice has data simulated from a conditional Poisson distribution $Y_j \mid S^{\text{ind}} \sim \text{Poi}(S)$ where the mean follows $S \sim \exp(\alpha)$. For the first five examples, the value of $\alpha$ within a hot spot is chosen to be smaller than it is outside of the hot spots, and for the last example, the value of $\alpha$ within a cold spot is chosen to be greater than it is outside of the cold spots. To search for hot spots, a minimum window size of $m_0 \times n_0 = 5 \times 5$ is used for KSS and MBSM-G. For MBSM-H, the fixed window size is also chosen to be $5 \times 5$. For SRB, a $5 \times 5$ minicell is used to reduce the lattice to size $20 \times 20$ with each lattice point now being a sum of counts within the $5 \times 5$ minicell. The results for each hot spot identification method, as well as the true configurations of hot spots, are shown in Figures 4.2 - 4.7. For KSS, MBSM-G, and MBSM-H, a legend is included in the figure to show the estimated magnitude of the identified hot spots.
Figure 4.2: Simulated example 1
Figure 4.3: Simulated example 2
Figure 4.4: Simulated example 3
Figure 4.5: Simulated example 4
Figure 4.6: Simulated example 5
As shown in Figures 4.2 - 4.7, KSS cannot consistently and accurately detect the secondary hot spots. The MBSM-G seems to circumscribe irregularly fragmented hot spots that are closer to each other as a broader region. Both KSS and MBSM-G have difficulties separating hot spots in the mountain gradient example shown in Figure 4.4. In each example, the fine granularity of MBSM-H captures the true hot spots and gives very good estimates of their magnitudes. However, MBSM-H can represent a hot spot in a somewhat fragmented way, so practitioners might need to interpolate the result.
for smoother conclusions. In addition, it is recommended to incorporate prior knowledge about the hot spot to pick the size of searching window when using MBSM-H since if the structural hot spots are large in size, a too small of a searching window may not be able to find them. In addition to the simulated hot spots using higher Poisson mean, MBSM-G and MBSM-H also capture random high density areas as hot spots. Because of the binary (i.e. red/blue) and irregular nature of the SRB, it shows irregular shaped hot spots without giving information on the estimated magnitude of each hot spot.

4.2.2 The Comparison Metric

After visualizing the hot spot configurations in the previous examples, the question remains is to how they might be compared quantitatively to each other and to the truth. The criterion we use to address this question is the variation of information (VI). The VI metric measures the amount of information lost and gained in changing from clustering $\mathcal{C}$ to clustering $\mathcal{C}'$ [43].

Suppose we have $K$ clusters $\{C_k\}_{k=1}^{K}$ associated with clustering $\mathcal{C}$ and $K'$ clusters $\{C'_k\}_{k'=1}^{K'}$ associated with clustering $\mathcal{C}'$, then the VI is calculated as

$$VI (\mathcal{C}, \mathcal{C}') = [H (\mathcal{C}) - I (\mathcal{C}, \mathcal{C}')] + [H (\mathcal{C}') - I (\mathcal{C}, \mathcal{C}')]$$  (4.3)

where $H (\mathcal{C}) = -\sum_{k=1}^{K} P (k) \log P (k)$, $H (\mathcal{C}') = -\sum_{k'=1}^{K'} P (k') \log P (k')$, and $I (\mathcal{C}, \mathcal{C}') = \sum_{k=1}^{K} \sum_{k'=1}^{K'} P (k, k') \log \frac{P (k, k')} {P (k) P (k')}$. Here define $P (k) = \frac{n_k} {n}$ where $n_k$ is the number of points in $C_k$, $P (k, k') = \frac{|C_k \cap C'_{k'}|} {n}$ where $n$ is the total number of data points in the study region, and $P (k') = \frac{n_{k'}} {n}$ where $n_{k'}$ is the number of points in $C'_{k'}$. 
Table 4.2: Normalized VI comparison of alternative hot spot identification methods

The first term of the VI measures the amount of information about \( C \) that we lose when moving from \( C \) to \( C' \), while the second term measures the amount of information that we gain by that move. The VI is bounded between 0 and \( \log n \), and the closer it is to 0, the better match between two clustering results. Table 4.2 shows the normalized VI (i.e. \( \frac{1}{\log n} \) VI) values comparing the results from all four methods to the truth. The VI values for all four methods are toward the left end on a \((0, 1)\) scale. Using this metric, SRB is out performed significantly by the other 3 methods, and MBSM-G does the best followed by KSS. For applications, examples 1, 2, and 5 might be considered the most realistic situations, and for those it can be seen that MBSM-H is competitive with both MBSM-G and KSS while enjoying appreciable computational advantages.

4.2.3 Real Data Analysis

The cottony cushion scale and its natural control agents were studied extensively by Quezada and DeBach [53] in southern California. Since natural enemies were so successful in controlling this pest throughout California, no studies on the economic threshold or sampling methods for detecting damaging levels of cottony cushion scale were conducted [8]. In recent years, various insecticides have disrupted cottony cushion scale by eliminating natural enemies, causing periodic local outbreaks [23], [24]. Our
data was collected to help develop a method that would reveal orchard hot spots that could be used to guide the release of natural enemies or the use of localized insecticide treatments.

The orchard used in this real data analysis was an 8 acre *Citrus grandis Osbeck × C. Paradisi* Macf. ‘Melogold’ grapefruit orchard (25 rows by 34 trees) located in Exeter, CA. The cottony cushion scales were sampled by examining 8 branches (4 branches at 2 m and 4 branches at 0.5 m from all four quadrants of the tree) in the interior of the tree and counting all adult female scales.

By using the MBSM-G, one hot spot is found and indicated by the red region in Figure 1.1. The pseudo subsampling in Figure 1.2 and 1.3 provides an opportunity to measure the effect of sub-sampling on the proposed hot spot identification procedures. Four hot spots are found on the 10% shifted subsampling data shown in Figure 1.2. Five hot spots are found on the 4% shifted subsampling data shown in Figure 1.3. The normalized VI for the hot spot configuration in Figure 1.2 relative to Figure 1.1 is 0.2238, and the normalized VI for the hot spot configuration in Figure 1.3 relative to Figure 1.1 is 0.2175. We can conclude that the MBSM-G finds hot spots and preserves similar amount of information with 4% and 10%.

### 4.2.4 Computational Time

To benchmark the computational time of the four compared methods, the run times for the six simulated examples and the real data on cottony cushion scale insects (Figure 1.1 - 1.3) were recorded. Table 4.3 presents the computation time of the analyses to find all significant hot spots, and the number in the parenthesis indicates the number of 64 GB 2.4 GHz Intel Xeon CPU processors that were in use to produce the results.
Table 4.3: Computation time comparison of alternative hot spot identification methods

While all of the computations are within a useable timeframe, MBSM-H shows a very advantageous strength when computing resources are limited. The MBSM-G can be used provided number of windows it requires to assess is not too large. The number of windows that MBSM-G assesses is \( N \) from equation 3.2. Note this formula holds for both exhaustive sampling and subsampling. Our experience is that the MBSM-G is feasible on a single computer if \( N \) is on the order of 1 million. For \( N \) on the order of 10 million, a moderate cluster of computers is likely to be necessary. Using the MBSM-G when \( N \) is much bigger that 10 million would require a very large cluster of computers.

### 4.3 CONCLUSION AND FUTURE WORK

As seen in the performance evaluation in Section 4.2, through Figures 4.2 - 4.7 and Table 4.2, all the methods have some potential for identifying hot spots. However, there is evidence in these evaluations that the GLMM based scan method offers superior resolution. MBSM-H seems better if capturing the varying magnitude of hot spots is of interest, whereas MBSM-G seems better if the goal is to identify practical regions of the lattice for subsequent treatment.
Our proposed MBSM-G/H procedures are defined under a Poisson GLMM model that sequentially searches for one cluster at a time. With regard to the multiple comparison concern associated with the sequential searches, it is shown in [64] that the type I error rate is very close to the nominal level when using a sequential scan procedure on data sets with only one true hot spot simulated. However, absent from [64] is a discussion of the type I error rate when two or more true hot spots are simulated. Another method was proposed in [38] to find multiple clusters where simultaneous scanning for two or more clusters is employed. This approach reduces the potential for a situation when inference about a candidate primary cluster is impaired by an additional cluster in the background. However, this approach becomes infeasible to use when considering three or more clusters simultaneously. Our proposed border test is an alternative approach that is significantly more scalable for applications that have multiple clusters. It is very encouraging that for all of our simulated examples our identified hot spots are always in the immediate vicinity of the true hot spots.

Our proposed MBSM-G/H procedure does not specify how the actual hot spots arise in the data set. In our simulated examples we considered a variety of alternatives. In all of these cases the proposed procedure succeeded in finding a good representation of the hot spots. Furthermore, the useful application to the real data set of cottony cushion scale insects provide additional evidence of robustness to the actual manner in which hot spots are generated. A more comprehensive robustness study would be useful for the future work.
Bibliography


Chapter 5

Appendix

5.1 Derivation of Equation 3.1

For a specific searching zone, the Poisson model proposed is $Y_1 \sim Poi(p_1 n_Z)$ and $Y_2 \sim Poi(p_2 (n_G - n_Z))$, with parameter space $\Theta = \{(p_1, p_2) : p_1 \geq p_2\}$. The likelihood function is

$$L(p_1, p_2) = \frac{e^{-p_1 n_Z} (p_1 n_Z)^{y_1}}{y_1!} \frac{e^{-p_2 (n_G - n_Z)} (p_2 (n_G - n_Z))^{y_2}}{y_2!}$$

Under the full parameter space $\Theta$, the likelihood function is maximized when $p_1 = \frac{y_1}{n_Z}$ and $p_2 = \frac{y_2}{(n_G - n_Z)}$, provided $\frac{y_1}{n_Z} > \frac{y_2}{(n_G - n_Z)}$, and otherwise is maximized when $p_1 = p_2 = \frac{(y_1 + y_2)}{n_G}$. Therefore, the likelihood ratio is

$$\lambda = \frac{\max_{H_0: H_1} L(p_1, p_2)}{\max_{H_0} L(p_1, p_2)} = 1 + \left(\frac{y_1}{n_Z}\right)^{y_1} \left(\frac{y_2}{n_G - n_Z}\right)^{y_2} I\left(\frac{y_1}{n_Z} > \frac{y_2}{(n_G - n_Z)}\right)$$
5.2 Derivation of Conditional test for Poisson mean

This is the derivation for the conditional test of the border test for candidate hot spot \( H_0 : s_1 \leq s_2 \) vs. \( H_a : s_1 > s_2 \).

Let \( U_1 = \sum_{j=1}^{n_1} y_{1j} \) and \( U_2 = \sum_{j=1}^{n_2} y_{2j} \), then conditionally we have \( U_1 \sim \text{Poi}(n_1 s_1) \) and \( U_2 \sim \text{Poi}(n_2 s_2) \). On the boundary of \( H_0 \), denoted by \( B_0 \), let \( s \) denote the common value of \( s_1 \) and \( s_2 \). Conditioning on \( U = U_1 + U_2 \),

\[
P_{B_0}(U_1 = u_1 \mid u_1 + u_2 = u) = \frac{P_{B_0}(U_1 = u_1, U_2 = u - u_1)}{P_{B_0}(u_1 + u_2 = u)}
= \frac{e^{-n_1 s_1} (n_1 s_1)^{u_1} e^{-n_2 s_2} (n_2 s_2)^{u-u_1}}{u_1! (u-u_1)!}
= \frac{u!}{u_1!(u-u_1)!} \left( \frac{(n_1 s_1)^{u_1} (n_2 s_2)^{u-u_1}}{(n_1 + n_2)^u} \right)
= \frac{u!}{u_1!(u-u_1)!} \left( \frac{(n_1)^{u_1} (n_2)^{u-u_1}}{(n_1 + n_2)^u} \right)
\]

Therefore, the size \( \alpha \) test is to reject \( H_0 \) if \( U_1 \) exceeds the upper \( \alpha \) percentile of a binomial \( \left( u, \frac{n_1}{n_1+n_2} \right) \) distribution. Similarly, the size \( \alpha \) test to test the candidate cold spot \( H_0 : s_1 \geq s_2 \) vs. \( H_a : s_1 < s_2 \) is to reject \( H_0 \) if \( U_1 \) is less than the lower \( \alpha \) percentile of a binomial \( \left( u, \frac{n_1}{n_1+n_2} \right) \) distribution.

5.3 Supplementary Material

This is the guide to functions \texttt{LatticeHotSpot()}, \texttt{LatticeColdSpot()}, and \texttt{HotSpotPlot()} written in R to implement the Generalized Linear Mixed Model based scan method using both global and heuristic search algorithms (MBSM-G and MBSM-H).
for hot/cold spots detection described in the paper as well as visualizing the output of hot/cold spots on a lattice array.

---

**LatticeHotSpot**  
*GLMM based scan method to search for hot spot*

---

**Description**

This function is used to implement the GLMM based scan method using both global and heuristic search algorithms (MBSM-G and MBSM-H) for hot spots detection described in the paper respectively.

**Usage**

`LatticeHotSpot(data, percpop, min_R, min_C, border_width, sigalpha, method=" ")`

**Arguments**

- **data**  
a matrix where each element input is the observed count at corresponding row and column location

- **percpop**  
the maximum percentage of lattice points considered

- **min_R**  
minimum row size of the searching windows for the MBSM-G or the fixed row size for the MBSM-H

- **min_C**  
minimum column size of the searching windows for the MBSM-G or the fixed column size for the MBSM-H
**border_width** number of rows and columns in the border

**sigalpha** type I error rate selected for the conditional border test

**method** “global” or “heuristic” to indicate the usage of MBSM-G or MBSM-H respectively

**Details**

Our experience is that the MBSM-G is feasible on a single computer if the number of search window \( N \) (defined in the paper) is on the order of 1 million. For \( N \) on the order of 10 million, a moderate cluster of computers is likely to be necessary. Using the MBSM-G when \( N \) is much bigger that 10 million would require a very large cluster of computers. MBSM-H shows a very advantageous strength when computing resources are limited.

If searching on the subsampling dataset, the missing values should be set as “NA”.

**Value**

**HotSpotConfig** a vector containing information of the size, location, and magnitude \((\text{Rsize}, \text{Csize}, \text{Rloc}, \text{Cloc}, \text{Shat})\) of the discovered hot spot

**Nhot** total number of hot spots found

**Rsize** number of rows of each discovered rectangular hot spot

**Csize** number of columns of each discovered rectangular hot spot

**Rloc** row index (y coordinate) of the lower left vertex of each discovered rectangular hot spot
**Cloc** column index (x coordinate) of the lower left vertex of each discovered rectangular hot spot.

**Shat** conditional mean of the count within each discovered rectangular hot spot.

**Examples**

test1_heu<-LatticeHotSpot(init, 1, 5, 5, 3, 0.05, "heuristic")

test1_heu

```r
## $HotSpotConfig
## [1,]   5   5  16  21  5.68  1.045831e-02
## [2,]   5   5  11  26  5.64  1.322903e-07
## [3,]   5   5   1  9  5.36  6.084189e-08
## [4,]   5   5  16  16  5.04  4.245628e-02

## $Nhot
## [1] 4

## $Rsize
## [1] 5 5 5 5

## $Csize
## [1] 5 5 5 5

## $Rloc
```

80
## [1] 16 11 1 16

## $Cloc

## [1] 21 26 9 16

## $Shat

## [1] 5.68 5.64 5.36 5.04
**LatticeColdSpot**  
*GLMM based scan method to search for cold spot*

**Description**

These functions are used to implement the GLMM based scan method using both global and heuristic search algorithms (MBSM-G and MBSM-H) for hot spots detection described in the paper respectively.

**Usage**

LatticeColdSpot(data, percpop, min_R, min_C, border_width, sigalpha, method=" ")

**Arguments**

- **data** a matrix where each element input is the observed count at corresponding row and column location
- **percpop** the maximum percentage of lattice point considered
- **min_R** minimum row size of the searching windows for the MBSM-G or the fixed row size for the MBSM-H
- **min_C** minimum column size of the searching windows for the MBSM-G or the fixed column size for the MBSM-H
- **border_width** number of rows and columns in the border
- **sigalpha** type I error rate selected for the conditional border test
- **method** “global” or “heuristic” to indicate the usage of MBSM-G or MBSM-H respectively
Details

Our experience is that the MBSM-G is feasible on a single computer if the number of search window N (defined in the paper) is on the order of 1 million. For N on the order of 10 million, a moderate cluster of computers is likely to be necessary. Using the MBSM-G when N is much bigger than 10 million would require a very large cluster of computers. MBSM-H shows a very advantageous strength when computing resources are limited.

If searching on the subsampling dataset, the missing values should be set as “NA”.

Value

**HotSpotConfig** a vector containing information of the size, location, and magnitude (Rsize, Csize, Rloc, Cloc, Shat) of the discovered hot spot

**Nhot** total number of hot spots found

**Rsize** number of rows of each discovered rectangular hot spot

**Csize** number of columns of each discovered rectangular hot spot

**Rloc** row index (y coordinate) of the lower left vertex of each discovered rectangular hot spot

**Cloc** column index (x coordinate) of the lower left vertex of each discovered rectangular hot spot

**Shat** conditional mean of the count within each discovered rectangular hot spot
Examples

test3_heu<-LatticeColdSpot(init, 1, 5, 5, 3, 0.05, "heuristic")

test3_heu

## $ColdSpotConfig
## [1,] 5 5 1 24 0.3333333 0.040672582
## [2,] 5 5 6 10 1.0000000 0.001587482
## [3,] 5 5 6 23 1.3333333 0.002931752
## [4,] 5 5 16 4 1.5000000 0.001839295

## $Ncold
## [1] 4

## $Rsize
## [1] 5 5 5 5

## $Csize
## [1] 5 5 5 5

## $Rloc
## [1] 1 6 6 16

## $Cloc
## [1] 24 10 23 4
## $Shat

## [1] 0.3333333 1.0000000 1.3333333 1.5000000
HotSpotPlot Visualizing results from GLMM based scan method for hot spot detection

Description

This function is used to plot the hot/cold spots found using functions Lattice-HotSpot() and LatticeColdSpot() to better visualize the results on a lattice array.

Usage

HotSpotPlot(data, hotspotconfig, spottype)

Arguments

data a matrix where each element input is the observed count at corresponding row and column location

hotspotconfig a matrix with each row contains information of the size, location, and magnitude (Rsize, Csize, Rloc, Cloc, Shat) of each discovered hot or cold spot

spottype “hot” or “cold”

Details

This function requires packages RColorBrewer and fields.

The maximum level of the color for hot/cold spots is 5 based on equal distance between the largest and smallest magnitude of the hot/cold spots.

Examples

HotSpotPlot(init, test3_heu$ColdSpotConfig, "cold")
## Loading required package: spam

## Loading required package: grid

## Spam version 0.40-0 (2013-09-11) is loaded.

## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.

## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

## Attaching package: 'spam'

## The following objects are masked from 'package:base':

##    backsolve, forwardsolve

## Loading required package: maps
Figure 5.1: Example of Cold spots
5.3.1 Function LatticeHotSpot()

```r
LatticeHotSpot <- function(data, percpop, min_R, min_C, border_width, sigalpha, method){
  nRow <- nrow(data)
  nCol <- ncol(data)
  data <- na.omit(data.frame(cbind(rep(seq(1:nRow),nCol),rep(seq(1:nCol),each=nRow)),
                               c(data[c(nRow:1),])))
  colnames(data) <- c("Row","Column","Total")
  nRow <- max(data$Row)
  nCol <- max(data$Column)
  sample <- (data$Column-1)*nRow+(data$Row)
  count <- tapply(data$Total,as.factor(sample),sum)
  newdata <- unique(data.frame(data$Row,data$Column,sample))
  newcount <- count[match(newdata$sample,as.numeric(names(count)))]
  newdata <- data.frame(newdata,newcount)
  sample <- newdata$sample
  size <- length(sample)
  sampleInd <- sample[order(sample)]
  y <- newdata$newcount[order(sample)]
  block <- matrix(1:(nRow*nCol),nrow=nRow,ncol=nCol)
  f_null <- function(a) {2/a - (sum(y_cur_zone) + 1)/(n_cur_zone + a) - (sum(y_out) + 1)/(n_out + a)}
  loglike <- function(a1,a2){
    log(a1) = sum(lgamma(y_cur_zone+1)) + lgamma(sum(y_cur_zone)+1)
    -(sum(y_cur_zone)+1)*log(n_cur_zone+a1)
    +log(a2) - sum(lgamma(y_out+1)) + lgamma(sum(y_out)+1)
    -(sum(y_out)+1)*log(n_out+a2)
  }
  # initial values for the searching
  Rsize = c()
  Csize = c()
  wr = c()
  wc = c()
}
```
b1hat = c()
b2hat = c()
Sihat = c()
pre_zone = c()
delete_block = c()
pvals = c()
Shat = c()
k = 1

## choose a searching algorithm: global or heuristic
if(method == "global"){
  R_expand <- seq(min_R,nRow,by=1)
  C_expand <- seq(min_C,nCol,by=1)
}
if(method == "heuristic"){
  R_expand <- min_R
  C_expand <- min_C
}

repeat {
  save = NULL
  index = NULL
  LR = NULL

  # start proposing searching zones
  for (cur_R in R Expand){
    for (cur_C in C扩and){
      cur_window_r <- 1:(nRow+1-cur_R)
      cur_window_c <- 1:(nCol+1-cur_C)
      allcomb = expand.grid(cur_window_r, cur_window_c)
      combnum = nrow(allcomb)
      for (i in 1:combnum){
        cur_wr <- allcomb[i,1]
        cur_wc <- allcomb[i,2]
        # constructing searching zones based on the location
        cur_zone <- which(is.element(sampleInd,
                                c(block[cur_wr:(cur_wr+cur_R-1)],
                                cur_wc:(cur_wc+cur_C-1)))==TRUE,arr.ind=TRUE)
      }
    }
  }
}
n_cur_zone <- length(cur_zone)
cur_block <- c(block[cur_wr:(cur_wr+cur_R-1),cur_wc:(cur_wc+cur_C-1)])
if(n_cur_zone<=percpop*nRow*nCol){
  # test if the current searching zone overlaps with the previous found clusters
  if (any(is.element(cur_block,delete_block))==FALSE) {
    y_cur_zone <- y[cur_zone]
    out <- union(pre_zone,cur_zone)
    y_out <- y[-out]
    n_out <- length(y_out)

    # maximize the likelihood for each proposed zone
    if(all(is.element(y_cur_zone,0)==TRUE)==FALSE &
    all(is.element(y_out,0)==TRUE)==FALSE){
      if (n_cur_zone>=1 & n_out>=1){
        a0 <- uniroot(f_null,c(0,100))$root
        if(n_cur_zone/sum(y_cur_zone)<n_out/sum(y_out)){
          ahat1 <- n_cur_zone/sum(y_cur_zone)
          ahat2 <- n_out/sum(y_out)
          # define likelihood function of specified model
          fit <- loglike(ahat1,ahat2)-loglike(a0,a0)
        } else{
          ahat1 <- ahat2 <- a0
          fit <- 0
        }
        LR <- rbind(LR,fit)
        index <- rbind(index,c(cur_R,cur_C,cur_wr,cur_wc,ahat1,ahat2))
        save <- cbind(index,LR)
      }
    }
  }
}
}
## identify the zone with maximum likelihood value

```r
save <- save[which(LR == max(LR), arr.ind = TRUE)[1,1],]
if(is.null(save)) == FALSE){
  Rsize <- c(Rsize, save[1])
  Csize <- c(Csize, save[2])
  wr <- c(wr, save[3])
  wc <- c(wc, save[4])
  b1hat <- c(b1hat, save[5])
  b2hat <- c(b2hat, save[6])
  zone <- which(is.element(sampleInd, c(block[wr[k]: (wr[k] + Rsize[k]-1),
                           wc[k]: (wc[k] + Csize[k]-1)])) == TRUE, arr.ind = TRUE)
  pre_zone <- c(pre_zone, zone)
  delete_block <- c(delete_block, c(block[wr[k]: (wr[k] + Rsize[k]-1),
                                wc[k]: (wc[k] + Csize[k]-1)]))

  ## define a border and conduct the border test
  if(wr[k] >= border_width + 1) {
    rowlow <- wr[k] - border_width
  } else { rowlow = 1 }
  if(wc[k] >= border_width + 1) {
    colow <- wc[k] - border_width
  } else { colow = 1 }
  if(wr[k] + Rsize[k] - 1 <= nRow - border_width) {
    rowup <- wr[k] + Rsize[k] - 1 + border_width
  } else { rowup = nRow }
  if(wc[k] + Csize[k] - 1 <= nCol - border_width) {
    colup <- wc[k] + Csize[k] - 1 + border_width
  } else { colup = nCol }
  background <- which(is.element(sampleInd, c(block[rowlow:rowup, colow:colup])) == TRUE, arr.ind = TRUE)
  border <- background[!background %in% pre_zone]
  x_zone <- y[zone]
  x_border <- y[border]
  m_zone <- length(zone)
  m_border <- length(border)
  x1s <- sum(x_zone)
```
x2s <- sum(x_border)
Shat <- c(Shat, mean(x_zone))
## conditional test of the means
pvals <- c(pvals, (1 - pbinom(x1s, (x1s + x2s), (m_zone / (m_zone + m_border))))
if (pvals[k] < sigalpha) { k = k + 1 }
else { break }
}
}
allConfig <- cbind(Rsize, Csize, wr, wc, Shat, pvals)
SigConfig <- matrix(allConfig[which(allConfig[,6] < sigalpha),], ncol=6)
return(list(HotSpotConfig = SigConfig, Nhot = nrow(SigConfig),
            Rsize = SigConfig[,1], Csize = SigConfig[,2],
            Rloc = SigConfig[,3], Cloc = SigConfig[,4], Shat = SigConfig[,5]))
5.3.2 Function LatticeColdSpot()

```r
LatticeColdSpot <- function(data, percpop, min_R, min_C, border_width, sigalpha, method) {
    nRow <- nrow(data)
    nCol <- ncol(data)
    data <- na.omit(data.frame(cbind(rep(seq(1:nRow),nCol),rep(seq(1:nCol),each=nRow),
    c(data[c(nRow:1),])))
    colnames(data) <- c("Row","Column","Total")
    nRow <- max(data$Row)
    nCol <- max(data$Column)
    sample <- (data$Column-1)*nRow+(data$Row)
    count <- tapply(data$Total,as.factor(sample),sum)
    newdata <- unique(data.frame(data$Row, data$Column, sample))
    newcount <- count[match(newdata$sample,as.numeric(names(count)))]
    newdata <- data.frame(newdata, newcount)
    sample <- newdata$sample
    size <- length(sample)
    sampleInd <- sample[order(sample)]
    y <- newdata$newcount[order(sample)]
    block <- matrix(1:(nRow*nCol),nrow=nRow,ncol=nCol)
    f_null <- function(a) {2/a -(sum(y_cur_zone)+1)/(n_cur_zone+a) - (sum(y_out)+1)/(n_out+a)}
    loglike <- function(a1,a2){
        log(a1)=sum(lgamma(y_cur_zone+1)+lgamma(sum(y_cur_zone)+1))
        -(sum(y_cur_zone)+1)*log(n_cur_zone+a1)
        +(log(a2)-sum(lgamma(y_out+1)+lgamma(sum(y_out)+1))
        -(sum(y_out)+1)*log(n_out+a2)
    }
    # initial values for the searching
    Rsize = c()
    Csize = c()
    wr = c()
    wc = c()
}```
b1hat = c()
b2hat = c()
Sihat = c()
pre_zone = c()
delete_block = c()
pvals = c()
Shat = c()
k = 1

## choose a searching algorithm: global or heuristic
if(method == "global"){
  R_expand <- seq(min_R,nRow,by=1)
  C_expand <- seq(min_C,nCol,by=1)
}
if(method == "heuristic"){
  R_expand <- min_R
  C_expand <- min_C
}
repeat{
  save = NULL
  index=NULL
  LR=NULL

  # start proposing searching zones
  for (cur_R in R_expand){
    for (cur_C in C_expand){
      cur_window_r <- 1:(nRow+1-cur_R)
      cur_window_c <- 1:(nCol+1-cur_C)
      allcomb = expand.grid(cur_window_r,cur_window_c)
      combnum = nrow(allcomb)
      for (i in 1:combnum){
        cur_wr <- allcomb[i,1]
        cur_wc <- allcomb[i,2]
        cur_zone <- which(is.element(sampleInd,
                                c(block[cur_wr:(cur_wr+cur_R-1),
                                cur_wc:(cur_wc+cur_C-1)])==TRUE,arr.ind=TRUE))
      }
    }
  }
}
n_cur_zone <- length(cur_zone)
cur_block <- c(block[cur_wr:(cur_wr+cur_R-1),cur_wc:(cur_wc+cur_C-1)])
if(n_cur_zone<=percpop*nRow*nCol){
    # test if the current searching zone overlaps with the previous found clusters
    if (any(is.element(cur_block,delete_block))==FALSE) {
        y_cur_zone <- y[cur_zone]
        out <- union(pre_zone,cur_zone)
        y_out <- y[-out]
        n_out <- length(y_out)
        # maximize the likelihood for each proposed zone
        if(all(is.element(y_cur_zone,0)==TRUE)==FALSE & all(is.element(y_out,0)==TRUE)==FALSE){
            if (n_cur_zone>=1 & n_out>=1){
                a0 <- uniroot(f_null,c(0,100))$root
                if(n_cur_zone/sum(y_cur_zone)>n_out/sum(y_out)){
                    ahat1 <- n_cur_zone/sum(y_cur_zone)
                    ahat2 <- n_out/sum(y_out)
                    # define likelihood function of specified model
                    fit <- loglike(ahat1,ahat2)-loglike(a0,a0)
                } else{
                    ahat1 <- ahat2 <- a0
                    fit <- 0
                }
                LR <- rbind(LR,fit)
                index <- rbind(index,c(cur_R,cur_C,cur_wr,cur_wc,ahat1,ahat2))
                save <- cbind(index,LR)
            }
        }
    }
}
## identify the zone with maximum likelihood value
save <- save[which(LR==max(LR),arr.ind=TRUE)[1,1],]

if(is.null(save)==FALSE){
  Rsize <- c(Rsize,save[1])
  Csize <- c(Csize,save[2])
  wr <- c(wr,save[3])
  wc <- c(wc,save[4])
  b1hat <- c(b1hat,save[5])
  b2hat <- c(b2hat,save[6])
  zone <- which(is.element(sampleInd,c(block[wr[k]:(wr[k]+Rsize[k]-1),
                                 wc[k]:(wc[k]+Csize[k]-1)]))==TRUE,arr.ind=TRUE)
  pre_zone <- c(pre_zone,zone)
  delete_block <- c(delete_block,c(block[wr[k]:(wr[k]+Rsize[k]-1),
                                    wc[k]:(wc[k]+Csize[k]-1)]))

  ## define a border and conduct the border test
  if(wr[k] >= border_width+1)
    rowlow <- wr[k]- border_width
  else { rowlow = 1}
  if(wc[k] >= border_width+1)
    colow <- wc[k]- border_width
  else { colow = 1}
  if(wr[k]+Rsize[k]-1 <= nRow - border_width)
    rowup <- wr[k]+ Rsize[k]-1+ border_width
  else { rowup <- nRow}
  if(wc[k]+Csize[k]-1 <= nCol - border_width)
    colup <- wc[k]+ Csize[k]-1+ border_width
  else { colup <- nCol}
  background <- which(is.element(sampleInd,
                               c(block[rowlow:rowup,colow:colup]))==TRUE,arr.ind=TRUE)
  border <- background[!background%in%pre_zone]
  x_zone <- y[zone]
  x_border <- y[border]
  m_zone <- length(zone)
  m_border <- length(border)
  x1s <- sum(x_zone)
  x2s <- sum(x_border)
Shat <- c(Shat, mean(x_zone))
## conditional test of the means
pvals <- c(pvals, pbinom((x1s), (x1s+x2s), (m_zone/(m_zone+m_border))))
if (pvals[k] < sigalpha) k = k+1
   else{break}
}
else{break}
}
al1Config <- cbind(Rsize, Csize, wr, wc, Shat, pvals)
SigConfig <- matrix(allConfig[which(allConfig[,6] < sigalpha),], ncol=6)
return(list(ColdSpotConfig=SigConfig,
  Ncold=nrow(SigConfig),
  Rsize=SigConfig[,1],
  Csize=SigConfig[,2],
  Rloc=SigConfig[,3],
  Cloc=SigConfig[,4],
  Shat=SigConfig[,5]))
}
5.3.3 Function HotSpotPlot()

```r
HotSpotPlot <- function(data, hotspotconfig, spottype) {
    library(RColorBrewer)
    library(fields)
    nRow <- nrow(data)
    nCol <- ncol(data)
    data = data.frame(cbind(rep(seq(1:nRow),nCol), rep(seq(1:nCol),each=nRow), c(data[c(nRow:1),])))
    colnames(data) <- c("Row","Column","Total")
    # assign sample index to each sampling site
    sample <- (data$Column-1)*nRow+(data$Row)
    count <- tapply(data$Total, as.factor(sample), sum)
    newdata <- unique(data.frame(data$Row, data$Column, sample))
    newcount <- count[match(newdata$sample, as.numeric(names(count)))]
    newdata <- data.frame(newdata, newcount)
    sample <- newdata$sample
    size <- length(sample)
    sampleInd <- sample[order(sample)]
    y <- newdata$newcount[order(sample)]
    block <- matrix(1:(nRow*nCol), nrow=nRow, ncol=nCol)
    datamap <- rep(0, nRow*nCol)
    zone <- list()
    Shat <- c()
    min_S <- min(hotspotconfig[,5])
    max_S <- max(hotspotconfig[,5])
    hotspots = NULL
    for (i in 1:nrow(hotspotconfig)) {
        iR = hotspotconfig[i,1]
        iC = hotspotconfig[i,2]
        wir = hotspotconfig[i,3]
        wic = hotspotconfig[i,4]
        Si= hotspotconfig[i,5]
        zone[[i]] = which(is.element(sampleInd, c(block[wir:(wir+iR-1),
            wic:(wic+iC-1)]))==TRUE, arr.ind=TRUE)
    }
}
```
```r
# Zone

datamap[zone[[i]]]=ceiling((Si-min_S)/((max_S-min_S)/(5-1)))+1

Row=rep(1:nRow,nCol)
Column=rep(1:nCol,each=nRow)
hotspotindex = c(datamap)
clusterpoints=data.frame(cbind(Row,Column,hotspotindex))
plot(seq(1:max(nRow,nCol)),seq(1:max(nRow,nCol)),
    type="n",xlab="Column",ylab="Row",xlim=c(0,nCol),ylim=c(0,nRow))
if (spottype=="cold") {spotcolor="Blues"}
else {spotcolor="Reds"}
for (i in 1:nrow(clusterpoints)){
    points(clusterpoints[i,2],clusterpoints[i,1],pch=19,col=brewer.pal(5,
        spotcolor)
        [clusterpoints[i,3]])
}
image.plot(legend.only=TRUE,nlevel=5, zlim=c(min_S,max_S),
    col=brewer.pal(5,spotcolor),
    smallplot=c(0.9,0.92,0.3,0.7))
    text(107,50,expression(hat(S)))
```