Chapter 4

Bayesian Computation for Point Patterns

4.1. Simulating point patterns

We begin this chapter by discussing methods for simulating point patterns under various model specifications. Simulating realizations of point patterns underlies our general Bayesian inference approach (Section 3.1) and is a powerful tool for model validation and prediction, as outlined in Section 3.2. Algorithms along with R software available for simulating point patterns will be given for some models.

4.1.1. Homogeneous Poisson Process (HPP)

To start, recall the HPP, the simplest stationary process with complete spatial randomness discussed in Section 2.1.1. The HPP assumes a constant intensity function, $\lambda(\mathbf{s}) = \lambda$ for all $\mathbf{s} \in D$. In the simulations to follow, we will assume $D \in \mathbb{R}^2$, but the approaches are easily generalizable to domains with higher dimension. Additionally, for convenience, we assume D to be the unit square, such that $D = [0,1] \times [0,1]$. At the end of this section, we offer an extension to these simulation procedures using rejection sampling that readily accommodates irregular domains.

Let $|D|$ be the area of the spatial domain. A simple two stage process to simulate from an HPP with $\lambda(\mathbf{s}) = \lambda$ for all $\mathbf{s} \in D$ begins by first drawing $N \sim Po(\lambda|D|)$ to obtain the random number of points in the region. Then, each point, $\mathbf{s}_i$ for $i = 1, 2, \ldots, N$, is randomly *located* independently and uniformly over D. That is, letting $\mathbf{s}_i = (s_{i1}, s_{i2})$, we draw $s_{i1} \sim Unif(0,1)$ and $s_{i1} \sim Unif(0,1)$. Collectively, $\mathcal{S} = \{\mathbf{s}_i; i = 1, \ldots, N\}$ specifies the realized point pattern of the HPP.

If D is not a product set, we embed it into a rectangle. Then we draw points uniformly over the rectangle, retaining those that fall in D. See Section 4.1.7 below for further discussion.

4.1.2. Nonhomogeneous Poisson Process (NHPP)

We immediately extend the simulation to the NHPP where $\lambda(\mathbf{s})$ is location-specific and varies with $\mathbf{s}$. This method dates to [122]. Letting $\lambda_{max} = \underset{\mathbf{s} \in D}{max}\ \lambda(\mathbf{s})$, we use the HPP simulation procedure by first drawing $N_{max} \sim Po(\lambda_{max}|D|)$ and then randomly locating the points in D. Let $\mathcal{S}_{max}$ denote the observed spatial point pattern from the HPP. The realization of the NHPP is obtained using a thinning procedure where each point, $\mathbf{s}_i \in \mathcal{S}_{max}$, is retained with probability $\frac{\lambda(\mathbf{s}_i)}{\lambda_{max}}$. That is, for each $\mathbf{s}_i$, $i = 1, 2, \ldots, N_{max}$, we sample an independent Bernoulli random variable with $p(\mathbf{s}_i) = \frac{\lambda(\mathbf{s}_i)}{\lambda_{max}}$ where a 1 means $\mathbf{s}_i$ is retained, 0 otherwise. The number, N, and collection of points retained, $\mathcal{S}$, yields the realization from the NHPP with intensity $\lambda(\mathbf{s})$. The proof is straightforward; we only need to show that, for any set $A \in D$, $N(A) \sim Po(\lambda(A))$.

Two issues to consider when simulating from an NHPP are the following. First, it is common for NHPPs to be specified using spatial covariates. For example, a spatial point process of the locations of tree species might be driven by climate variables such as temperature and precipitation, or elevation. Here, the intensity surface might be specified in log-linear form where $\log\lambda(\mathbf{s}) = \mathbf{x}^T(\mathbf{s})\boldsymbol{\gamma}$ where $\mathbf{x}(\mathbf{s})$ is a vector of covariates for location $\mathbf{s}$, which is, in theory, observable for all $\mathbf{s} \in D$ (recall Section 2.3). In practice, spatial covariates are often only available across a collection of grid cells that form a disjoint partition of the spatial domain. Therefore, the best we can do is approximate the continuous intensity surface, $\lambda(\mathbf{s})$, with a piecewise constant intensity. For locations simulated randomly within the domain using the NHPP simulation procedure, two locations within the same grid cell will have the same probability of being retained but independent Bernoulli random variables are sampled for each.

Second, the procedure outlined above assumes λ_{max} can be computed directly. That is, $\lambda(\mathbf{s})$, and, therefore, $\mathbf{x}(\mathbf{s})$ need to be available everywhere in order to obtain the maximum over the intensity surface. In practice, again, we will have to rely on the disjoint partition of the spatial domain in order to approximate the maximum intensity to be used in the simulation procedure.

4.1.3. Log Gaussian Cox Process (LGCP)

Recall that the LGCP, introduced in Section 2.3, is an extension of the NHPP where the intensity function, $\lambda(\mathbf{s})$, is a stochastic process. Simulating a realization of a LGCP requires first simulating a realization of the stochastic process, namely, a realization of the Gaussian process. Then, conditioning on this realization, we have an NHPP with intensity $\lambda(\mathbf{s})$ and can follow the procedure above using thinning to obtain our realization from the LGCP.[1] Due to the multiple stages of random sampling, i.e., the Gaussian process realization, the number of points, and the location of points, we provide additional details for this approach.

To begin, we must first obtain a realization of the Gaussian process, referred to as $z(\mathbf{s})$, that is defined over $\mathbf{s} \in D$. We will assume $z(\mathbf{s})$ is a mean zero GP with valid spatial covariance function $C(\mathbf{s}, \mathbf{s}')$ specified according to Section 1.4. To obtain a realization of the GP, we first need to define a set of representative points, $\mathcal{U}$, at which we will *observe* $z(\mathbf{s})$. In theory, this set of representative points can be chosen in any way; for example, randomly sampling a set of locations in D. Since, however, we will be using this realization of the GP to aid in simulating from the point process, it is often more efficient to discretize the domain into a collection of grid cells that form a disjoint partition of D and use, for example, the centroid of each grid as a representative point. The realization of the LCGP is sensitive to this discretization and a finer resolution of the surface will result in a better approximation of the true Gaussian process[2].

Let $\mathcal{U}$ denote the set of representative points such that $\mathbf{u}_j \in \mathcal{U}$ and $|\mathbf{u}_j|$ is the area of the grid cell j with centroid $\mathbf{u}_j$. Then, we obtain a finite realization of the GP at the set of representative points by drawing $\mathbf{z} \sim MVN(\mathbf{0}, \mathbf{C})$ where $\mathbf{C}$ is the covariance matrix between the set of representative points. The realization of $\mathbf{z}$ at the grid cell centroids results in a tiled surface over D, and, in turn, a tiled

[1] To be clear, the resulting point pattern realization is *given* the GP realization. We are not attempting to create point patterns *marginalized* over the GP.

[2] The downside with increasing the grid resolution is that as the dimension of $\mathcal{U}$ increases, we have to sample from an increasingly higher dimensional multivariate normal distribution which becomes increasingly computationally expensive.

surface for the intensity. That is, $\lambda_0(\mathbf{u}_j) = e^{z(\mathbf{u}_j)}$ denotes the *baseline* intensity for the entire grid cell containing $\mathbf{u}_j$ for each $\mathbf{u}_j \in \mathcal{U}$.

Then, as in Section 4.1.2, we add the regressors, $\mathbf{x}^T(\mathbf{s})\boldsymbol{\gamma}$, resulting in $\lambda(\mathbf{u}_j) = e^{\mathbf{x}^T(\mathbf{u}_j)\boldsymbol{\gamma}}\lambda_0(\mathbf{u}_j) = e^{\mathbf{x}^T(\mathbf{u}_j)\boldsymbol{\gamma}+z(\mathbf{u}_j)}$.

Now, conditional on the realization of the GP, $\mathbf{z}$, $\lambda(\mathbf{u}_j)$ fully specifies the intensity of an NHPP. Therefore, we can obtain a realization of the LGCP using the NHPP simulation procedure where $\lambda_{max} = \underset{\mathcal{U}}{max}\ \lambda(\mathbf{u})$. That is, we draw $N_{max} \sim Po(\lambda_{max}|D|)$ and randomly locate the N_{max} points in D, where $\mathcal{S}_{max}$ denotes the collection of points. The thinning procedure introduced above is again employed to obtain the simulated spatial point pattern under the specified LGCP. Letting $\mathbf{u}_i$ denote the centroid of the grid cell containing simulated point $\mathbf{s}_i \in \mathcal{S}_{max}$, $\mathbf{s}_i$ is retained with probability $p(\mathbf{u}_i) = \frac{\lambda(\mathbf{u}_i)}{\lambda_{max}}$. It is important to note that each sample point $\mathbf{s}_i \in \mathcal{S}_{max}$ is retained independently of all other points, even if there are multiple $\mathbf{s}_i$'s located in the same grid cell. Again, the number and locations of the points retained makes up the realization of the point pattern $\mathcal{S}$ from the LGCP.

4.1.4. Cluster Processes

We now turn from Poisson processes and the conditional independence assumption to the simulation of point patterns that exhibit dependence in the form of spatial clustering. In general, clustering in spatial point patterns can be thought of as groups, or *clusters*, of points that have interpoint distances smaller than the average distance between points across the domain. Two common families of cluster processes include the Neyman Scott process and the shot noise processes (Section 2.4). The simulation procedures for each of these cluster processes are directly generative and outlined in detail below.

The Neyman Scott process can be described as a *parent-offspring* process. The two-stage procedure for simulating this family of cluster processes begins by randomly simulating a set of *parent* nodes/locations within the spatial domain. Then, for each parent, *offspring* are simulated randomly and independently around the parent node, which requires first sampling randomly the number of offspring for each parent and then randomly locating them within D. A simulated realization of the Neyman Scott process is the result of the collection of offspring from this two-stage process (refer to Figure 2.4). That is, the parents are removed.

More formally, the Neyman Scott process requires a specification of the parent intensity and an offspring intensity, which could be specified conditionally given the parent. A common choice for the parent intensity might be an NHPP, where $\lambda(\mathbf{s})$ is defined over D and parent nodes can be simulated according to the procedure outlined above. Note that a simpler HPP or a more general LGCP would also be suitable models for the parent process.

Next, for each parent node, we need to generate the number and locations of the offspring. Assuming K parent nodes were generated with locations $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$, one approach is to simulate N_k, the number of offspring for parent k where $k = 1, 2, \ldots, K$, according to, e.g., $N_k \overset{iid}{\sim} Po(\delta)$, and then locate them by simulating i.i.d. samples from the bivariate density $f(\mathbf{s}; \boldsymbol{\mu}_k)$. Customarily, the offspring of parent $\boldsymbol{\mu}_k$ are located using the bivariate Gaussian density $N(\boldsymbol{\mu}_k, \tau^2\mathbf{I})$ making this analogous to a (modified) Thomas process [101]. Alternatively, a Matérn process can be used in which the offspring are simulated uniformly within a circle of radius R centered at $\boldsymbol{\mu}_k$. The resulting collection of offspring from all K parent nodes yields the simulated point pattern from Neyman Scott process.

One might recognize the similarity between the collection of K bivariate densities, $f(\mathbf{s}; \boldsymbol{\mu}_k)$, used to produce the offspring in the Neyman Scott and mixture models, defined by a convex combination of a collection of densities. Recalling Section 2.4, we can reformulate the simulation of offspring under the Neyman Scott process given the collection of parent nodes, $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$, in the context of mixture models. First, under the Poisson specification of N_k above, we simulate the total number of offspring according to, e.g., $N \sim Po(K\delta)$. Other distributions are possible for simulating N_k, however, distributions in the exponential family make derivation of the distribution of $N = \sum_{k=1}^{K} N_k$ easy to obtain. Given N, we then locate the offspring by simulating i.i.d. samples from the mixture distribution $\mathbf{s}_i \sim \sum_{k=1}^{K} \frac{1}{K} f(\mathbf{s}; \boldsymbol{\mu}_k, \omega^2$ for $i = 1, 2, \ldots, N$.

Shot noise processes are another class of models for cluster processes where, like the LGCP, the intensity is stochastic. Here, a common form for a shot noise process is $\lambda(\mathbf{s}) = e^{\mathbf{x}^T(\mathbf{s})\boldsymbol{\gamma}} \lambda_0(\mathbf{s})$ where $e^{\mathbf{x}^T(\mathbf{s})\boldsymbol{\gamma}}$ is borrowed from the NHPP specified with covariates $\mathbf{x}(\mathbf{s})$ and $\lambda_0(\mathbf{s})$ is a mean 1 shot process. As discussed in Section 2.4, a simple yet flexible form for the shot process is $\lambda_0(\mathbf{s}) = \sum_{\mathbf{s}_i \in \mathcal{S}_{shot}} f(\mathbf{s} - \mathbf{s}_i) m(\mathbf{s}_i)$. Here, f is a unimodal density over D centered at 0, $m(\mathbf{s}_i) \geq 0$, and the summation is over the collection of points in the shot process. $m(\mathbf{s}_i)$ is referred to as the "shot" and the density f spreads the influence of the shot at $\mathbf{s}_i$ on $\lambda(\mathbf{s})$ according to the distance between $\mathbf{s}$ and $\mathbf{s}_i$.

To simulate from the shot noise process, first simulate $\mathcal{S}_{shot} \sim HPP(\lambda)$ to obtain a realization of points for the shot process. If we assume a fixed constant for the shot such that $m(\mathbf{s}_i) = m$, then, by letting $m = 1/\lambda$, $E(\lambda_0(\mathbf{s})) = 1$ and thus, $E(\lambda_0(D)) = |D|$. Now, using the simulated points of the shot process, compute $\lambda_0(\mathbf{s}) = \sum_{\mathbf{s}_i \in \mathcal{S}_{shot}} f(\mathbf{s} - \mathbf{s}_i) m(\mathbf{s}_i)$ and finally, $\lambda(\mathbf{s}) = e^{\mathbf{x}^T(\mathbf{s})\boldsymbol{\gamma}} \lambda_0(\mathbf{s})$ is the resulting shot noise intensity for $\mathbf{s} \in D$.

Conditional on $\lambda(\mathbf{s})$, we find ourselves with an NHPP intensity from which we need to simulate a point pattern. Recall that in simulating from an NHPP, we first obtained $\lambda_{max} = \underset{\mathbf{s} \in D}{max}\ \lambda(\mathbf{s})$ and then simulated from an HPP and used rejection sampling to obtain the realization from the NHPP. Here, we again need to obtain λ_{max}, which can be done by taking the maximum $\lambda(\mathbf{s})$ over a set of representative points. For a large collection of representative points we expect to obtain a good approximation to the true maximum value of $\lambda(\mathbf{s})$ over D. Then, using λ_{max}, we can obtain a realization from the shot noise process using the same approach as for the NHPP. First, simulate $N_{max} \sim Po(\lambda_{max}|D|)$ and locate them randomly in D with $\mathcal{S}_{max}$ denoting their locations. Then, each simulated point $\mathbf{s}_j \in \mathcal{S}_{max}$, is retained with probability $\frac{\lambda(\mathbf{s}_j)}{\lambda_{max}}$ where $\lambda(\mathbf{s}_j) = e^{\mathbf{x}^T(\mathbf{s}_j)\boldsymbol{\gamma}} \lambda_0(\mathbf{s}_j)$ and $\lambda_0(\mathbf{s}_j) = \sum_{\mathbf{s}_i \in \mathcal{S}_{shot}} f(\mathbf{s}_j - \mathbf{s}_i) m(\mathbf{s}_i)$. The resulting collection of points retained yields a simulated point pattern from the shot noise process.

4.1.5. Gibbs Process

Gibbs processes offer a flexible class of models for point patterns when there is dependence between points. Most commonly, Gibbs processes are used to capture negative dependence, or *inhibition*, but can be used to model clustering of points as well. Various versions of inhibition processes are described in detail in Section 2.5. Depending on the process specifications, such as strength of inhibition or attraction of the point pattern or whether the number of points in the process is fixed or random, different simulation algorithms have been proposed in the literature.

For example, consider the Strauss process from Section 2.5, which exhibits inhibition through a penalty based on distance between pairwise points. Here, it is possible for two points to lie closer than distance d_0 apart like in a hard-core process but can be unlikely. The Strauss process is intuitive from a simulation perspective. First, generate a realization, $\mathcal{S}$, from an HPP within the domain D. If no pair of points is less than some threshold distance $d0$ apart, we accept the realization. Otherwise, for each close pair of points, we have an independent Bernoulli trial to decide whether to reject or accept the pair. If *any* pairs of points are rejected, the entire point pattern is rejected. Letting $0 \leq p \leq 1$ denote the probability of acceptance of a pair of points and $n(\mathcal{S}, d)$ the number of close pairs of points in the point pattern $\mathcal{S}$, the realization $\mathcal{S}$ is accepted based on a Bernoulli trial with probability $p^{n(\mathcal{S},d_0)}$. When $p = 1$, $\mathcal{S}$ will be accepted with probability 1 and thus, represents an HPP. When $p = 0$, this results in a hard-core process such that $\mathcal{S}$ is rejected when $n(\mathcal{S}, d_0) > 0$. Therefore, for $0 < p < 1$, the Strauss process spans the range between complete spatial randomness and the hard-core process. Note that both increases in d_0 and decreases in p will decrease the acceptance probability of candidate realizations of $\mathcal{S}$, making the simulation procedure outlined above increasingly inefficient.

An alternative approach for simulating from a more general Gibbs process uses a Markov chain Monte Carlo (MCMC) algorithm. In the Bayesian framework, MCMC algorithms are often used to obtain samples from the desired posterior distribution in conducting model inference. Importantly, MCMC iterative procedures can also be used as a simulation algorithm where the MCMC algorithm is based on a spatial birth-death process [10, 16, 101] The algorithm begins with an initial point pattern and after a series of iterations where points are added (births) and removed (deaths), the algorithm will tend to an equilibrium state. Once approximate equilibrium is reached, any instantaneous snapshot of the spatial patterns of points can be assumed to follow the probability law of a Gibbs process. That is, given $Q(\mathcal{S})$ (Section 2.5) the joint location density for a particular point pattern realization, $\mathcal{S}$, is proportional to $e^{-Q(\mathcal{S})}$. To obtain a collection of realizations of a Gibbs process, one continues iterating through the birth-death process where the number of iterations between retained snapshots of the spatial patterns of points is large enough such that dependence between observations is negligible.

We offer a few more details on the MCMC approach for simulating Gibbs processes. The MCMC algorithm consists of random births and deaths of points, ultimately reaching an equilibrium state. Assume $\mathcal{S}$ is the initialized point pattern for the birth-death process. Additionally, assume a repulsive Gibbs process where d_0 denotes the threshold distance as discussed above. Then, for any small window of time, Δt, each point, independently, has probability $m\Delta t$ of mortality (i.e. being removed from the point pattern). Additionally, letting b denote the birth rate per unit area and per unit time, for some small spatial region of area Δa, a new point is "born", independently, with probability $bp^k\Delta t\Delta a$, where p is the probability of acceptance and k is the number of points in the current point pattern $\mathcal{S}$ that lie closer than the threshold d_0 of the proposed birth point. Again, $p = 0$ implies a hard-core process. Iterating between random births and deaths in this fashion creates the sequence of spatial point patterns. Note that once equilibrium is reached, this doesn't mean that realizations of the point pattern will consist of the same set of points. Rather, the random locations of points will follow the same specified probability law of the Gibbs process.

As opposed to the long-run approximations from the birth-death MCMC procedure, *perfect simulation* provides an alternative method for simulating from Gibbs

processes [163, 164]. The benefit of perfect simulation is that it solves the problem of having to assess when approximate equilibrium of the target distribution has been reach. Perfect simulation is a Markov chain simulation algorithm such that the exact equilibrium is attained when the algorithm completes; it is therefore able to return perfect simulations from the target distribution. Perfect simulation based on coupling from the past (CFTP) repeatedly generates upper and lower Markov chains starting increasingly further back in time until the pair of chains merge at time 0. Then, the chains return a *perfect* simulation from the specified target distribution. While CFTP is the most common perfect sampler, other perfect samplers exist in the literature [24, 138].

The general approach for perfect simulation using CFTP assumes that the state space is finite, has an ordering such that the sampler is monotone, and that there are unique minimal and maximal elements in which the lower and upper processes are started, respectively [163]. Then, starting the two chains at any time $t \leq 0$ in any arbitrary state, the chain produced by the sampler is contained between the two chains. [163] showed that for sufficiently large t, under weak ergodicity conditions, the two chains will coalesce and yield a simulation from the target distribution.

[109] and [95] show how perfect simulation using CFTP algorithms can be used to conduct perfect simulation for point processes. Whereas the original simulation work of [163] assumed the target distribution to be attractive, [109] generalized the approach for repulsive processes. In particular, they generalized the algorithm for an area-interaction point process to obtain perfect simulations from its target equilibrium distribution of a spatial birth-death process.

[110] outline a dominated CFTP for pairwise interaction point processes, where "dominating" processes are used for generating the upper and lower processes of the CFTP algorithm. The dominating processes are generated backward in time and are used to generate the upper and lower processes. The upper and lower processes are generated forward in time starting at $t \leq 0$ using birth-death processes until the upper and lower processes at time 0 are equal. In addition, they propose an algorithm for doing perfect simulation for spatial point patterns using the Metropolis-Hastings algorithm.

Pairwise interaction processes are also addressed in [25], where the dominated CFTP perfect simulation and path sampling [78] are combined for likelihood based inference and non-parametric Bayesian inference. [95] extend the perfect simulation algorithm using a two component Gibbs sampler for a bivariate point process model in order to obtain exact samples from a mixture model. They investigate the algorithm in terms of computational feasibility through simulation and find that exact samples from the target distribution can be obtained as long as the rate of the underlying Poisson processes is small or moderate. Perfect simulation algorithms are further utilized for simulating from multivariate discrete and continuous target distributions by [137].

Finally, as we briefly mentioned in Section 2.5, the determinantal point process is another example of an inhibition process. Simulation for this process is challenging and is presented in full detail in [119]. Such simulation makes it a natural candidate for model fitting using approximate Bayesian computation (ABC) as discussed in [184]. (See Section 4.3 below, as well).

The `spatstat` package in R includes functions for simulating from Gibbs processes using various simulation algorithms. Versions of these simulation algorithms include a Metropolis-Hastings algorithm, which reaches approximate convergence to the underlying Gibbs process, as well as perfect simulation, which guarantees convergence but often at the expense of large computation time. In general, as the

strength of inhibition in the Gibbs process increases, the computation time needed to reach convergence also increases, thus, requiring more advanced algorithms. *Simulated tempering* [82, 131] is one such approach for processes with strong inhibition where auxiliary variables are introduced into the algorithm to decrease autocorrelation. The `spatstat` package, using a Metropolis-Hastings algorithm, provides a method for simulating point patterns using tempering. Details of the simulated tempering algorithm as they pertain to simulation and inference for hard-core Gibbs processes can be found in [133].

4.1.6. Marked point patterns

Marked point patterns offer rich opportunities to think about the process(es) generating the joint distribution of the locations and marks of the point pattern and to specify models that suitably capture these patterns. A marked point pattern is a collection of observations in the form $(\mathbf{s}, m)$ where $\mathbf{s}$ represents the random spatial location of the point and m the mark. As discussed in Section 2.6, marks can be continuous, where the support of the marks, M is a subset of $\mathbb{R}^1$, or discrete, where M is a set of labels, l, $l = 1, 2, \ldots, L$. For continuous marks, the marked point pattern is equivalent to a point pattern over $D \times M$. Given a model for the intensity function $\lambda(\mathbf{s}, m)$, $(\mathbf{s}, m) \in D \times M$, such as an NHPP, Cox process, or Gibbs process, the marked point pattern can be simulated using the respective procedure specified above.

For discrete marks, referred to here as labels, it makes sense to first decompose the joint distribution of $(\mathbf{s}, m)$ into the product of a conditional and marginal. From a Bayesian modeling perspective, the joint distribution of $(\mathbf{s}, m) \in D \times M$ can be specified as either $[\mathcal{S}|\text{label} = l]P(\text{label} = l)$ or $P(L(\mathbf{s}) = l|\mathbf{s} \in \mathcal{S})[\mathcal{S}]$ (Section 2.6). The former arises from specification of a marginal distribution for the marks along with a conditional distribution for the point pattern given the mark. The latter assumes a point pattern model for the locations and then a conditional distribution for the mark at each location in the pattern; the mark is viewed as a *response* at the location. As discussed in Section 2.6, these two joint modeling approaches are incompatible; one must make a process-based decision as to which specification to choose. As a result, the simulation approaches differ for each of the specifications.

We first consider the $[\mathcal{S}|\text{label} = l]P(\text{label} = l)$ approach. Here, the marginal distribution of discrete marks could be modeled using a discrete distribution with L categories, such as a multinomial distribution with probability p_l of being in category l. Then, the conditional distribution $[\mathcal{S}|\text{label} = l]$, could be modeled using any of the aforementioned processes. For example, with processes specified using intensities, each $\lambda_l(\mathbf{s})$ could be defined as an NHPP such that $\log\lambda_l(\mathbf{s}) = \mathbf{x}^T(\mathbf{s})\boldsymbol{\gamma}_l$ where $\boldsymbol{\gamma}_l$ denotes label-specific coefficients. To simulate, first draw a random mark, l, from the multinomial distribution with categories, $1, 2, \ldots, L$. Then, given the mark, simulate a location from the process $\lambda_l(\mathbf{s})$. Together, this yields the realization $(\mathbf{s}, L(\mathbf{s}) = l)$. Now, this poses the question of how many points to simulate? For each of the $l = 1, 2, \ldots, L$ point processes, first compute $\lambda_{l,max} = \underset{\mathbf{s}\in D}{max}\ \lambda_l(\mathbf{s})$. Then, sample $N_{max} \sim Po(\sum_{l=1}^{L} p_l\lambda_{l,max})$. Now, for $i = 1, 2, \ldots, N_{max}$, simulate a random mark, m_i from the multinomial distribution. Then, given the mark $m_i = l$, follow the two step procedure of sampling from an NHPP by simulating uniformly over D, and retaining the point, $\mathbf{s}_i$, with probability $p = \frac{\lambda_l(\mathbf{s}_i)}{\lambda_{l,max}}$. The collection of retained points yields the marked point pattern.

The alternative process model decomposes the joint likelihood into the product $P(L(\mathbf{s}) = l|\mathbf{s} \in \mathcal{S})[\mathcal{S}]$. Here, we have a marginal process model for a realization of a point pattern of locations, and a conditional distribution for the mark at a location in the realized pattern. Therefore, we need only specify one process, for instance, with $\lambda(\mathbf{s})$ for $\mathbf{s} \in D$ and a discrete distribution for labels, $l = 1, 2, \ldots, L$ which depends upon location. The point process can be modeled using any of the models discussed. The conditional distribution for marks could follow a multinomial regression model with location-specific covariates, $\mathbf{x}(\mathbf{s})$. For example, the distribution of a mark (e.g., species label) could be driven by a set of environmental conditions, such that for a given location (environmental conditions), the distribution reflects the relative chances for the species at that location. Given the set of simulated locations $\mathcal{S}$, the only additional step is sampling from a conditional labeling distribution for each $\mathbf{s} \in \mathcal{S}$ to obtain the collection of marked points.

4.1.7. Other considerations for constructing and simulating point patterns

The procedures outlined above yield realizations of spatial point patterns under various explicit model specifications. Other considerations for the construction and simulation of point patterns include irregularly shaped spatial domains, domains of higher dimension, as well as possible process enrichment through thinning procedures, displacement, censoring, and superposition. In this section we give a brief overview of these ideas, highlighting how they might be used in practice to better specify the underlying process and mechanism for data collection.

Irregular spatial domains

In the procedures outlined above, we assumed the spatial domain was the unit square, which enabled us to sample locations by independently drawing from two standard uniform distributions. In practice, the spatial domain can take any shape (and be of any dimension). Here, we offer a few comments regarding simulating point patterns for general domains. It is trivial to extend the sampling to rectangular spatial domains with non unit-length widths and heights by applying appropriate shifts and/or scalings of the uniform distributions above.

For irregularly shaped domains, e.g., state or county boundaries, lakes, national forests, even after shifts, scalings, or rotations, a realization of the spatial point process can not be obtained by drawing from two independent uniform distributions. In this case, an efficient approach for simulating the point pattern entails embedding the irregular domain of interest, D, into a rectangular domain, D^* such that $D \subset D^*$. Then, under any of the process models outlined above, we can simulate points under D^* and use rejection sampling to retain only those in $D \cap D^*$. Note, here that we reject each point with $p = 1$ if it is outside D, retain it with $p = 1$ if it is in D. While D^* can be defined to be any rectangle such that $D \subset D^*$, in practice it should be specified as small as possible to obtain maximum efficiency in sampling points (i.e., retaining a high percentage of points.) Under an NHPP model, for example, $N^*_{max} \sim Po(\lambda_{max}|D^*|)$, and points are located uniformly over D^*. Points that are not contained in D are rejected, and those in D are retained according to their specified probability using an appropriate Bernoulli trial.

Thinned processes

Thinned processes are obtained through two surfaces, namely, the intensity associated with the point process, $\lambda(\mathbf{s})$ along with a thinning surface, $p(\mathbf{s})$, with realized values $0 \leq p(\mathbf{s}) \leq 1$. Here, $\lambda(\mathbf{s})$ might be specified using any of the point process models discussed above. The thinning process $p(\mathbf{s})$ can take various forms. A simple thinning process, known as p-thinning assumes $p(\mathbf{s}) = p$ for all $\mathbf{s} \in D$ such that each point in the spatial point pattern is retained independently with probability p. Other versions might view $p(\mathbf{s})$ as say a population density surface so that the thinning corresponds to a population density thinning. That is, if $\lambda(\mathbf{s})$ is an intensity associated with the incidence of locations of a particular disease, then $p(\mathbf{s})\lambda(\mathbf{s})$ provides a population (at risk) adjustment to the intensity surface. A more stochastic thinning process could view $p(\mathbf{s})$ arising as a cdf transformation of a Gaussian process.

With regard to simulating thinned processes, a spatial point pattern is first simulated according to $\lambda(\mathbf{s})$. The thinning process is applied at the second stage of the simulation, analogous to the second stage in the NHPP simulation procedure, where each point simulated in the first stage is retained independently with associated probability $p(\mathbf{s})$. The collection of retained points yields the simulated thinned spatial point pattern [54].

Simple p-thinning has also been proposed as a tool for model validation with spatial point patterns (Section 3.2). The thinning is used to create training and test datasets [121]. Recall that in traditional model validation, a training set, say, consisting of 80% of the data is randomly partitioned from the test set consisting of the remaining 20%. In spatial point pattern analyses, p-thinning entails using a Bernoulli random variable with probability p of being retained for each data point in $\mathcal{S}$, yielding the training point pattern $\mathcal{S}_{train}$. The random collection of points not retained creates the test point pattern, $\mathcal{S}_{test}$. Under proper rescaling by the ratio $\frac{1-p}{p}$ to the fitted intensity surface obtained from training the modeling using $\mathcal{S}_{train}$, model cross-validation can be conducted using $\mathcal{S}_{test}$.

We note that this p-thinning procedure can be applied to any spatial point process model with a conditionally independent location density, e.g., an NHPP, a LGCP, even more general Cox processes, with care. The resulting induced intensity will agree with that of the underlying process of the spatial point pattern up to a scaling. With, e.g., pairwise dependence between locations to prescribe the joint location density, as in a Gibbs process, p-thinning can not be applied. Such thinning will reduce the size of the point pattern and, as a result, change the interpoint distance structure.

Again, richer specifications of $p(\mathbf{s})$ are also common and often take a more process-based approach for point process modeling and simulation. For example, assume that a given spatial domain provides a suitable habitat for a particular species and the distribution of this species is defined according to a spatial process with intensity $\lambda(\mathbf{s})$. In addition, based on some biotic or abiotic environmental conditions, e.g., soil moisture or a high abundance of predator species, there may be subregions within the spatial domain that are less likely to contain the particular species. The thinning process, $p(\mathbf{s})$, offers an additional stochastic modification that is independent of the primary process and can be employed to capture such deviations to the primary spatial process. Therefore, given realizations from the primary process, $\lambda(\mathbf{s})$, using the simulation approaches outlined above, each point is retained according to a realization from the thinning process with respective probabilities $p(\mathbf{s})$. The *operating* intensity for the observed point pattern becomes $p(\mathbf{s})\lambda(\mathbf{s})$. (See Section 5.1 for an example in this regard.)

In practice, we note that sampling from and applying the thinning process, $p(\mathbf{s})$ may also require a disjoint partition of the spatial domain and a collection of representative points. For example, in the thinning by population density example above, the population density surface will be at areal unit scale, e.g., block or census tracts.

Superposition

The superposition, or aggregation, of two or more component spatial point processes offers a natural way of enriching the specification of a point process. For instance, let $\mathcal{S}_1$ and $\mathcal{S}_2$ each denote a spatial point pattern with intensity functions, $\lambda_1(\mathbf{s})$ and $\lambda_2(\mathbf{s})$, respectively. Then, their superposition $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$ also forms a spatial point pattern. Simulating from a superposition spatial point process entails simulating from each of the component process using the procedures outlined above and combining them onto one underlying space. Realizations of a marked spatial point pattern where the marks are discrete labels can be easily simulated by a superposition point process. Here, letting $\lambda_l(\mathbf{s})$ define the intensity of the process for label l where $l = 1, 2, \ldots, L$, a superposition point process could be obtained by $\bigcup_{l=1}^{L} \mathcal{S}_l$.

Various specifications of the component spatial point processes offer useful flexibility in building a marginal point pattern from the superposition point process. As a special case, when each component process is a Poisson point process, the resulting marginal superposition point process is also a Poisson process with intensity $\lambda(\mathbf{s}) = \sum_{l=1}^{L} \lambda_l(\mathbf{s})$. An early version of this considers a point pattern of disease incidence which resulted from a *background* intensity surface, essentially the baseline process, with an overlay of an intensity for points during a high risk season or a disease outbreak setting [107]. If $\lambda_B(\mathbf{s})$ is the baseline intensity and $\lambda_E(\mathbf{s})$ is the elevated risk intensity, then an interesting ratio here would be $\frac{\lambda_E(\mathbf{s})}{\lambda_B(\mathbf{s})+\lambda_E(\mathbf{s})}$.

Going further, [54] shows that the superpositioned point process of a homogeneous Poisson process and a Poisson cluster process has second-order properties that are equivalent to pure Poisson cluster processes. However, interestingly, the nearest neighbor properties between the Poisson cluster process and the superpositioned process remain quite different. Similarly, the superposition of inhibition processes, such as two Gibbs processes, can also yield point processes with complex second-order and nearest neighbor properties. While each Gibbs process will itself exhibit inhibition between points, within a specified radius, the superposition point process need not retain minimum distance behavior between points or number of points.

Lastly, shot noise process realizations can also naturally be developed by additive superposition. That is, the shot noise process is the superposition of the realizations of each of the random kernels. Recall the shot noise process $\lambda(\mathbf{s}) = e^{\mathbf{x}^T(\mathbf{s})\boldsymbol{\gamma}}\lambda_0(\mathbf{s})$ where $\lambda_0(\mathbf{s})$ is the shot process such that $\lambda_0(\mathbf{s}) = \sum_{\mathbf{s}_i \in \mathcal{S}_{shot}} f(\mathbf{s}-\mathbf{s}_i)m(\mathbf{s}_i)$ and $\mathcal{S}_{shot}$ is the realization of the shot locations. Let $\lambda_{\mathbf{s}_i}(\mathbf{s})$ denote the random kernel for shot $\mathbf{s}_i \in \mathcal{S}$ such that $\lambda_{\mathbf{s}_i}(\mathbf{s}) = e^{\mathbf{x}^T(\mathbf{s})\boldsymbol{\gamma}} f(\mathbf{s}-\mathbf{s}_i)m(\mathbf{s}_i)$. Realizations from each random kernel, $\lambda_{\mathbf{s}_i}(\mathbf{s})$, can be easily simulated, from which their superposition yields the realization of the shot noise process. Lastly, the shot noise process can be defined by the summation $\lambda(\mathbf{s}) = \sum_{\mathbf{s}_i \in \mathcal{S}_{shot}} \lambda_{\mathbf{s}_i}(\mathbf{s})$.

Displacement

Displacement is a point process operation that entails random relocations of the point pattern, obtained by randomly moving points of the point process to other

locations within the defined domain. For each iteration of the operation, each point $\mathbf{s}_i \in \mathcal{S}$ is randomly and independently displaced, such that $\mathbf{s}_i \rightarrow \mathbf{s}_i + \mathbf{h}_i$ and the collection of displaced points, $\mathbf{s}_i + \mathbf{h}_i$ yields the new realization of the point process. There are many ways of specifying the methods of displacement, such as rectangular or radial displacement. In simulation, care needs to be taken to control for the edges of the domain to ensure that displaced points remain in D. Toroidial boundaries are one choice, such that when points disappear off the right edge, they re-appear at the left. Adding a buffer zone is another choice, such that the point pattern is imbedded in a larger region and the maximum distance of the rectangular or radial shift is less than the size of the buffer [36, 197].

Most commonly, displacement is used in testing hypotheses pertaining to the underlying processes of observed point patterns. Monte Carlo randomization tests, for example, can be used to test claims regarding distributional assumptions of a point pattern by comparing values of a test statistic, such as the K function, $K(d)$, for some distance, d, to assess clustering. The test statistic is first computed using the observed point pattern and compared to values of the statistic computed using a random and independent sequence of displaced point patterns. We note here that the function $rshift()$ in the R package `spatstat` conducts random shifts of point patterns that could be used in this capacity.

Another interesting application of displacement would be for a superposition of component point patterns. For example, consider the superposition of locations of male adults and yearlings of a species of birds, or two different species of trees across a forest with continuous marks denoting the diameter at breast height. There may be scientific insight to understanding not only the distribution of both adults and yearlings or the different tree species and sizes, but also in the their interaction. The continuous marks could be treated as a third dimension of the point pattern where the domain is defined as $D \times M$, $M \in \mathbb{R}^1$. In this particular case, random and independent displacements of the points could be used to test whether or not there is clustering or inhibition between species of tree (or adults and yearlings) and whether or not these patterns are similar for different sizes of trees.

Censoring

Consider spatial point patterns as observational data in ecological studies, such as a collection of locations where a particular bird call was heard and identified, or the locations of observed grass species across a field site. In each of these examples, based on the sampling effort of the individuals who are collecting the data or the rate of missed detections, both of which could vary across space, the observed spatial point pattern would be a censored, degraded, or partially-realized version of the true point process. When censoring is homogenous across the spatial region, the observed point pattern would resemble that of a p-thinned point pattern as discussed above. Imperfect detection might be an example of a constant censoring process such that the ability to detect a species is the same across the entire spatial region. On the other hand, censoring that might vary across space could be the result of varying sampling effort across a region. For regions within the domain with zero sampling effort, $p = 0$. Nonhomogeneous censoring would mimic that of a $p(\mathbf{s})$-thinned process.

Section 5.1 below describes censoring in the context of presence-only data for species distributions. For a given bounded region D, there is a finite set of species locations which can be considered as a point pattern, i.e., the full census of individuals in D. However, in practice, money and time considerations imply that

sampling effort over D will be sparse. The observed point pattern of locations will be a degraded/censored point pattern of the full census of locations due to the fact that some regions have never been sampled. The *operating* intensity for those regions is censored to 0.

As another example, the Breeding Bird Survey (BBS) is a massive citizen science survey database of birds seen and identified across North America[3]. It is common for birders to revisit the same locations, resulting in high sampling effort in regions such as nature areas or parks. Therefore, the observed spatial point pattern of a particular species of interest might exhibit clustering of points in these high traffic areas and few to no points in other areas. From an inferential standpoint for spatial point pattern analysis, the effects of both homogenous and nonhomogeneous censoring can be dramatic, as they could lead to vastly underestimating abundance of a species or misinformation with regard to its spatial distribution. To simulate a censored point pattern, the procedures outlined above for simulating p-thinning and $p(\mathbf{s})$-thinning can be applied.

Measurement error in locations

In many applications, the observed point pattern $\mathcal{S}$ may not represent the collection of *true* locations of the points. For example, a forest inventory analysis may rely on remotely sensed data from, say, light detection and ranging technology [LiDAR, 124], which yields spatial location of a species of tree in a forest. Due to measurement error of LIDAR, whether known or unknown, the observed point pattern may differ from the true locations of the trees. These discrepancies are often referred to as map positional error in geographic information systems (GIS) applications. In practice, ignoring measurement error could have important impacts on model inference, especially since these data products often transcend map projections and scalings.

From a modeling perspective, we can account for this uncertainty by defining the model hierarchically and adding a measurement error model to the data level. This would suggest looking at the observed location as $\mathbf{s}_{obs} = \mathbf{s}_{true} + \boldsymbol{\epsilon}(\mathbf{s})$ where $\boldsymbol{\epsilon}(\mathbf{s})$ is a bivariate noise process capturing the error in measurement. Here, the collection of points $\mathbf{s}_i \in \mathcal{S}$ is treated as one realization. [18] develop models that address map positional error in order to infer the true location of feature coordinates. (See also [45]). In addition, they consider multiple realizations of the point pattern from different sources with possibly varying accuracy, and propose model averaging strategies to improve estimation of the true locations of the points. Model-based inference also yields uncertainty quantification where model averaging yields predictions with smaller uncertainty than any individual model. The hierarchical form of the model can be naturally handled within the Bayesian framework; both model fitting and simulation are easily attainable under general specification of the latent process generating the *true* point pattern along with a measurement error specification.

4.2. Computation strategies for Bayesian model fitting

Following Section 3.1, our generic strategy for Bayesian model fitting requires simulation of point patterns under a given model along with fitting the given model

[3] https://www.pwrc.usgs.gov/bbs/

to a dataset. For us, the broad range of Bayesian inference described in Chapter 3 proceeds subsequent to the model fitting. In this section we attend to the computational approaches for such model fitting.

4.2.1. General comments on model fitting

We begin with a few words regarding general fitting of spatial point process models. A broad tool is the minimum contrast method as advocated by Diggle [54]. This is essentially a method of moments idea. First, a summary function, typically the K function (Section 2.2), is computed from the observed point pattern. Second, the theoretical expected value of this summary statistic under the point process model is derived (if possible, as an algebraic expression involving the parameters of the model) or estimated from simulations of the model. Then the model is fitted by finding the optimal parameter values for the model to give the closest match between the theoretical and empirical curves. With a single parameter, the criterion becomes $\int(\hat{K}(d) - K(d))^2 dd$. With additional parameters, often powers are introduced.

More generally, for a model with parameter vector $\boldsymbol{\theta}$, the minimum contrast estimator arises as the value of $\boldsymbol{\theta}$ which minimizes $\sum_d |\hat{K}_d(\mathcal{S})^a - K_{\boldsymbol{\theta}}(d)^a|^b$ where $K_{\boldsymbol{\theta}}$ is the theoretical K function, $\hat{K}_d(\mathcal{S})$ is the empirical estimator for the K function (Section 2.2), and a and b are user-specified parameters [54]. The summation replaces the integral by using a set of radii d for the K function.

From our perspective, likelihood-based methods are more attractive and have recently become more common practice [12]. For instance, with the homogeneous Poisson process (HPP), we have a closed form likelihood and the MLE is straightforward. For the nonhomogeneous process (NHPP), the Berman-Turner device [22] is attractive. It connects the NHPP log likelihood to a weighted Poisson regression log likelihood using quadrature to do numerical integration. It is also well suited for Bayesian model fitting of the NHPP and is detailed in Section 4.2.2 below. As mentioned in Section 2.3.1, for the log Gaussian Cox process (LGCP), we have a stochastic integral in the likelihood which can never be obtained explicitly. Customarily, we resort to numerical integration using so-called "representative points" as remarked in Section 2.3.1. Since we need likelihood evaluation in Bayesian model fitting, it is employed there as well (Section 4.2.2 below). Very recently, [87] proposed a fitting strategy that avoids numerical integration of the stochastic integral.

For Neyman-Scott clustering processes, likelihood-based inference is computationally very demanding and is not straightforward to implement. Hence, the method of minimum contrast is the usual choice; implementation with a Cauchy distribution kernel is offered in `spatstat`. For Markov and Gibbs processes, a common approach is to employ the pseudo-maximum likelihood using the Papangelou conditional intensity. That is, the pseudo-likelihood is expressed through $\Pi_i \lambda(\mathbf{s}_i|\mathcal{S}/\mathbf{s}_i)$. Again, the `spatstat` package [12] is a very useful piece of software which offers likelihood fitting for many spatial point pattern models.

4.2.2. Bayesian computational strategies for log Gaussian Cox processes

Bayesian inference for Cox processes is natural since the observed point pattern depends on the latent random process. In the case of the LGCP, the point pattern depends on the latent Gaussian process. Using Bayes' Theorem, we can derive the inverse relationship and obtain inference for the latent random process given the

observed point pattern. [94] offer a nice overview of Bayesian inference for point processes, focusing attention on Poisson processes, doubly stochastic processes (e.g., the LGCP), as well as cluster processes.

In this section we begin with a brief review of Bayesian model fitting for the HPP. We then continue with the extension of Bayesian inference to the NHPP using the Berman-Turner device [22]. Building on these process specifications, we address Bayesian modeling fitting of the LGCP. In particular, due to its wide usage, we introduce a collection of Markov chain Monte Carlo methods for obtaining Bayesian inference for the LGCP. Then, we discuss integrated nested Laplacian approximation (INLA) as a computationally efficient approach for obtaining approximate Bayesian inference.

For what follows, as usual, let D denote the spatial domain and $\mathcal{S}$ the collection of observed spatial points, $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n$ such that $\mathbf{s}_i \in D$ for all $i = 1, 2, \ldots, n$. The intensity function of the HPP is $\lambda(\mathbf{s}) \equiv \lambda$ over the entire domain, D. Bayesian posterior inference for the parameter λ requires specification of the likelihood function and prior distribution. Here, it is convenient and computationally efficient to assign a conjugate Gamma prior distribution to λ such that $\lambda \sim Gamma(\alpha, \beta)$ where α is the shape parameter and β is the rate parameter. The posterior distribution of λ given the data, $\mathcal{S}$ can be written

$$\begin{aligned}[\lambda|\mathcal{S}] &\propto L(\lambda; \mathcal{S})[\lambda] \\ &\propto e^{-\lambda(|D|+\beta)}\lambda^{n+\alpha-1}\end{aligned}$$

where $|D|$ denotes the area of D. This posterior follows a Gamma distribution with shape $n+\alpha$ and rate $|D|+\beta$. As such, direct samples from the posterior distribution can be obtained efficiently.

In the context of the NHPP and the LGCP, our likelihood function takes of the form

$$L((\lambda(\mathbf{s}), \mathbf{s} \in D); \mathcal{S}) = e^{-\lambda(D)} \prod_{i=1}^{n} \lambda(\mathbf{s}_i)$$

where $\lambda(D) = \int_D \lambda(\mathbf{s}) d\mathbf{s}$. For NHPPs, this is a regular integral, whereas for the LGCP, this integral is stochastic, requiring further computational considerations. Except for in the simplest forms where the true intensity $\lambda(\mathbf{s})$ varies as a tiled surface across D, numerical integration will yield an approximation to $\lambda(D)$. The Berman-Turner device [22], using numerical quadrature, offers a simple and efficient approach for numerical integration. Here, we discuss it in the context of spatial point patterns and as a method of approximating $\int_D \lambda(\mathbf{s}) d\mathbf{s}$. We assume the intensity function of the NHPP is specified with parameter vector $\boldsymbol{\gamma}$. For example, the intensity could take a log-linear form where $\log\lambda_{\boldsymbol{\gamma}}(\mathbf{s}) = \mathbf{x}(\mathbf{s})'\boldsymbol{\gamma}$ with $\mathbf{x}(\mathbf{s})$ denoting a vector of covariates for location $\mathbf{s}$ and $\boldsymbol{\gamma}$ a vector of coefficients needing to be estimated.

To begin, we will approximate the integral using the weighted sum $\sum_{j=1}^{m} w_j \lambda_{\boldsymbol{\gamma}}(\mathbf{u}_j)$ where the $w_j > 0$ are the quadrature weights that sum to $|D|$ and $\mathbf{u}_j \in D$ for $j = 1, 2, \ldots, m$ are the quadrature points. We can approximate the log likelihood of the NHPP using these quadrature weights and points as

$$\log L(\boldsymbol{\gamma}; \mathcal{S}) \approx \sum_{i=1}^{n} \log\lambda_{\boldsymbol{\gamma}}(\mathbf{s}_i) - \sum_{j=1}^{m} w_j \lambda_{\boldsymbol{\gamma}}(\mathbf{u}_j).$$

A natural and convenient choice for quadrature points includes both the observation points $\mathbf{s}_i \in \mathcal{S}$ along with other randomly sampled points within D. Additional

comments with regard to the quadrature points and weights are given below. Letting 1_j denote an indicator variable such that $1_j = 1$ if point $\mathbf{u}_j$ is a point in $\mathcal{S}$ and 0 otherwise, we can write the log likelihood as

$$\log L(\boldsymbol{\gamma}; \mathcal{S}) \approx \sum_{j=1}^{m} 1_j \log \lambda_{\boldsymbol{\gamma}}(\mathbf{u}_j) - w_j \lambda_{\boldsymbol{\gamma}}(\mathbf{u}_j).$$

By defining $y_j = 1_j / w_j$, we can conveniently rewrite this as

$$\log L(\boldsymbol{\gamma}; \mathcal{S}) \approx \sum_{y=1}^{m} (y_j \log \lambda_{\boldsymbol{\gamma}}(\mathbf{u}_j) - \lambda_{\boldsymbol{\gamma}}(\mathbf{u}_j)) w_j,$$

which now takes the form of a weighted log likelihood of independent Poisson variables $Y_j \sim Po(\lambda_{\boldsymbol{\gamma}}(\mathbf{u}_j))$. Standard generalized linear model software can be employed to obtain approximate MLE estimates under this specification.

In obtaining inference within the Bayesian framework, we can utilize this approximation of the log likelihood in a Metropolis-Hastings algorithm for sampling from the posterior distribution of $\boldsymbol{\gamma}$. In practice, note that inference in either paradigm requires evaluation of $\lambda_{\boldsymbol{\gamma}}(\mathbf{u}_j)$ for all $\mathbf{u}_j$. That is, for all observed points in $\mathcal{S}$ as well as the quadrature points. Under a loglinear speciation of the NHPP where $\log \lambda_{\boldsymbol{\gamma}}(\mathbf{s}) = \mathbf{x}(\mathbf{s})^T \boldsymbol{\gamma}$, this requires obtaining the covariates $\mathbf{x}(\mathbf{u}_j)$ for all $j = 1, 2, \ldots, m$. Now, sampling from the posterior distribution of $\boldsymbol{\gamma}$ given $\mathcal{S}$ will require a Metropolis-Hastings algorithm. Assuming a vector form for $\boldsymbol{\gamma}$, these could be updated individually or as a block. In either case, the standard Metropolis-Hastings ratio comprised of likelihood, prior, and proposal can be employed where the approximate likelihood is used instead of the true likelihood.

One final consideration when using numerical quadrature in practice is the choice of quadrature points and weights. In general, numerical integration provides more accurate approximations to an integral when the resolution of the discretized surface is high. Discretizing the spatial region using a fine resolution grid is one possible choice to ensure adequate spatial coverage of the domain. In such a case, weights can then be computed as the ratio of the area of each grid cell relative to $|D|$. For an NHPP specified above with location-specific covariates, the resolution of the grid need only be as fine as the scale at which the covariates are observed. In ecological applications, for example, variables such as soil moisture or canopy cover may only be observable at a specified scale, e.g., hectare scale, in which case the resolution of the tiled surface of $\lambda_{\boldsymbol{\gamma}}(\mathbf{s})$ will only be at the hectare scale.

The discussion of quadrature points and weights above with regard to numerical integration provides a nice motivation for the specifications required for Bayesian inference for the LGCP. Recall that the extension from NHPPs to LGCPs arises from adding a Gaussian process to the intensity function such that $\lambda(\mathbf{s})$ is now stochastic. For example, we might write $\log \lambda(\mathbf{s}) = \mathbf{x}(\mathbf{s})^T \boldsymbol{\gamma} + z(\mathbf{s})$ where $z(\mathbf{s})$ is a Gaussian process. In the literature, these point processes are often referred to as *doubly* stochastic processes.

As noted above, the random, infinite dimensional $\lambda(\mathbf{s})$ yields a stochastic integral in the likelihood that cannot be evaluated explicitly, and thus, exact Bayesian inference is intractable. Discretization approaches have been proposed [e.g., 20, 165] in order to obtain a tractable expression of the likelihood where the continuous surface is approximated by a piecewise constant function over a collection of grid cells that form a disjoint partition of the spatial domain. This is analogous to step functions in one dimension. As the resolution of the grid used to approximate the

continuous surface increase, [206] showed that the posterior density converges to the true posterior. In practice however, the choice of grid reflects the trade-off between the accuracy of the approximation and its computational complexity, recognizing that inference can be sensitive to the discretization scheme that is applied.

Recall the Berman-Turner device above, which relies on being able to evaluate $\lambda(\mathbf{s})$ at all quadrature points. Under the LGCP specification, $\lambda(\mathbf{s})$ is now random so direct application of numerical integration techniques is not possible. Here, in order to complete the numerical integration, we will need to condition on a realization of the Gaussian process at a set of representative points. For example, if we obtain a realization of the Gaussian process at all quadrature locations, $\mathbf{u}_j, j = 1, 2, \ldots, m$, we can evaluate the approximate log likelihood using the approach above. Note, however, that in addition to the possible hyperparameters in the mean and covariance function of the Gaussian process, the realization of the Gaussian process at the representative points is now also part of the joint posterior distribution. Common specification of the Gaussian process assumes a mean $-\sigma^2/2$ (this choice is made in order that, for $\lambda(\mathbf{s}) = e^{\mathbf{x}(\mathbf{s})^T\boldsymbol{\gamma}}e^{z(\mathbf{s})}$, we have $Ee^{z(\mathbf{s})} = 1$), variance σ^2, and correlation specified using an appropriate covariance kernel (e.g., Matérn covariance function), with parameter(s) $\boldsymbol{\phi}$. Collectively, our MCMC sampling algorithm requires iterative sampling of the parameters $\boldsymbol{\gamma}$, σ^2, $\boldsymbol{\phi}$, as well as obtaining the needed realizations of the Gaussian process, $z(\mathbf{s})$.

While this may seem straightforward, sampling from the Gaussian process can pose many computational challenges. First, using standard geostatistical models with spatial random effects, such as Gaussian processes, MCMC algorithms are much more computational efficient when the models are specified marginally [16]. That is, when the Gaussian process is integrated out. Even when inference is obtained conditionally, a linear model form with normal errors enables conjugate, direct sampling of the spatial random effects. Here, under a LGCP, we do not have closed form conditional distributions for posterior sampling and cannot marginalize over the Gaussian process since we require realizations for numerical integration.

In general, Markov chain Monte Carlo methods are common for fitting LGCPs. MCMC can be computationally challenging, and often require precise tuning in order to obtain proper mixing and convergence. Some advanced sampling algorithms that mitigate these challenges are discussed in detail below. As an alternative to MCMC, approximate model fitting and inference for LGCPs can be obtained using integrated nested Laplace approximation (INLA) [176]. In the remainder of this section, we will visit some of the common and more recent approaches proposed in the literature for Bayesian model inference of LGCPs.

Elliptical slice sampling

[144] proposed the elliptical slice sampler as an efficient way to obtain samples of a multivariate latent Gaussian random variable in a generalized linear model when the likelihood function prohibits the posterior distribution of the latent random variable to be sampled directly. The elliptical slice sampler is suitable for obtaining samples from a posterior distribution that is proportional to the product of a multivariate Gaussian distribution and a likelihood function that connects the Gaussian prior to the data. Sampling from the posterior distribution of the random variable $z(\mathbf{s})$ of the LGCP is an example of this form, where the log intensity is specified as $\log\lambda(\mathbf{s}) = \mathbf{x}^T(\mathbf{s})\boldsymbol{\gamma} + z(\mathbf{s})$ and $z(\mathbf{s})$ is modeled as a Gaussian process.

Let $\mathbf{z}$ denote a realization of the Gaussian process at a collection of representative spatial locations defining a lattice over the spatial region. These points, representing grid cells, should be specified at a sufficiently fine resolution such that with low probability, two points in the observed point pattern are in the same grid cell and the error in using a piecewise constant approximation for the continuous intensity surface is negligible. For a finite set of locations, $\mathbf{z}$ follows a multivariate normal distribution with covariance Σ capturing the spatial correlation across locations. In specifying the algorithm, we will assume $\mathbf{z}$ is mean $\mathbf{0}$, however this can easily be generalized to non-zero mean distributions by a shift through a change of variables.

Let $\mathbf{z}$ denote the current value of the process realization at the specified set of locations. Step one of the algorithm entails sampling from the prior distribution of $\mathbf{z}$ to define the *ellipse* for the algorithm. More precisely, $\boldsymbol{\nu} \sim MVN(\mathbf{0}, \Sigma)$. Then, the threshold for the log likelihood used in the accept/reject step of the algorithm is obtained by sampling $u \sim Unif[0, 1]$. Now, to draw the initial sample, $\mathbf{z}_c$, we first sample $\theta \sim Unif[0, 2\pi]$ and define the bracket $[\theta_{min}, \theta_{max}] = [\theta - 2\pi, \theta]$. Then, the candidate value, $\mathbf{z}_c$, is obtained by setting $\mathbf{z}_c = \mathbf{z}cos\theta + \boldsymbol{\nu}sin\theta$. This draw of $\mathbf{z}_c$ is accepted as a sample from the posterior distribution if $\log L(\mathbf{z}_c; \mathcal{S}, \boldsymbol{\gamma}) - logL(\mathbf{z}; \mathcal{S}, \boldsymbol{\gamma}) > \log u$ where $\log L(\mathbf{z}_c; \mathcal{S}, \boldsymbol{\gamma})$ and $\log L(\mathbf{z}; \mathcal{S}, \boldsymbol{\gamma})$ are the log likelihoods evaluated at $\mathbf{z}_c$ and $\mathbf{z}$, respectively, given the data, $\mathcal{S}$, and other parameters, $\boldsymbol{\gamma}$. If $\mathbf{z}_c$ is rejected, we shrink the bracket above such that if $\theta < 0$, $\theta_{min} = \theta$, or if $\theta \geq 0$, $\theta_{max} = \theta$. A new value of θ is then sampled such that $\theta \sim Unif[\theta_{min}, \theta_{max}]$ and we obtain our new candidate value of $\mathbf{z}$ as above. We continue through this procedure of shrinking the bracket and obtaining new draws of θ and candidate values of $\mathbf{z}$ until we accept the draw based on the log likelihoods and threshold draw. Therefore, each iteration of the MCMC algorithm results in a unique sample of $\mathbf{z}$.

Computationally, the most expensive part of the elliptical slice sampler outlined above for obtaining realizations of the Gaussian process in the LGCP model is in sampling $\boldsymbol{\nu}$. For m representative points, this cost is $\mathcal{O}(m^3)$. Conditional on the realization of $\mathbf{z}$, and, thus, the intensity surface $\lambda(\mathbf{s})$, the observations of the point pattern are independent and computing the log likelihood potentially costs $\mathcal{O}(m + n)$ if the representative points and the observed points are distinct. If the observed points are *projected* to the nearest representative points then the cost becomes $\mathcal{O}(m)$.

This sampler can easily be imbedded into an MCMC algorithm where other model parameters are updated using traditional Gibbs sampling or Metropolis-Hastings algorithms. One benefit of the elliptical slice sampler is that it allows $\mathbf{z}$ to be sampled in block. This is an advantage in terms of mixing over algorithms that sample each component separately, especially when the random effects $\mathbf{z}$ exhibit strong dependence. Additionally, the step-size within the algorithm is controlled by the bracket $[\theta_{min}, \theta_{max}]$. Whereas other algorithms require tuning to obtain appropriate step sizes, the elliptical slice sampler properly controls the step-size within each iteration of the algorithm to more efficiently draw likely candidate values of the latent Gaussian random variable.

Metropolis-Adjusted Langevin algorithm

A second MCMC method for inference for the LGCP uses the Metropolis-Adjusted Langevin algorithm (MALA) [142]. Traditional Metropolis-Hastings algorithms tend to perform poorly in terms of mixing and convergence when proposal distributions

and tuning parameters are poorly chosen, and the problem is compounded when parameters and latent processes are highly dependent [84]. The Metropolis-Hastings algorithm with a Langevin-type proposal offers an efficient approach when the gradient of the transition density can be written down explicitly. MALA simulates from the posterior distribution based on a linear transformation of the variable(s) of interest. Proposals of the new states for the (transformed) parameters or random variables are obtained using the gradient of the target posterior density and they are accepted or rejected using the Metropolis-Hastings algorithm [171, 174]. For traditional Metropolis-Hastings algorithms, the target acceptance rate for a random walk proposal and Gaussian target distribution has been shown to be 0.234 [170]. [171, 173] found 0.57 to be the optimal acceptance rate for MALA and [6] developed an algorithm to find the tuning parameter to achieve this optimal rate.

Let $\mathcal{U}$ denote the set of grid cells in the discretization such that $\mathbf{u}_j, j = 1, 2, \ldots, m$ are the grid cell centroids and $|\mathbf{u}_j|$ is the area of grid cell j. Here, $\lambda(\mathbf{u}_j)$ is the piecewise constant intensity for grid cell j. Let $\mathbf{z} = (z(\mathbf{u}_1), z(\mathbf{u}_2), \ldots, z(\mathbf{u}_m))$ denote the realization of the Gaussian process at the set of m locations. Now, we are interested in obtaining samples from $\pi(\mathbf{z}|\mathcal{S}, \boldsymbol{\theta})$ where $\mathcal{S}$ is the observed spatial point pattern and $\boldsymbol{\theta}$ are all other parameters. Let $\mathbf{z} \sim MVN(-\frac{\sigma^2}{2}\mathbf{1}, \Sigma)$ where Σ is a spatial covariance matrix with variance σ^2 and spatial dependence specified according to a spatial covariance function. The transformation of $\mathbf{z}$ is such that $\tilde{\mathbf{z}} = \Sigma^{-1/2}(\mathbf{z} + \frac{\sigma^2}{2}\mathbf{1})$. Now, MALA is used to obtain samples from $\pi(\tilde{\mathbf{z}}|\mathcal{S}, \boldsymbol{\theta}) \propto \pi(\mathcal{S}|\mathbf{z}, \boldsymbol{\theta})\pi(\tilde{\mathbf{z}})$.

The Langevin-type proposal distribution for $\tilde{\mathbf{z}}$ is

$$p(\tilde{\mathbf{z}}_{cand}|\tilde{\mathbf{z}}) \sim MVN(\tilde{\mathbf{z}} + \frac{1}{2}\nabla \log(\pi(\tilde{\mathbf{z}}|\mathcal{S}, \boldsymbol{\theta})), h^2\mathbf{I})$$

where $\nabla \log(\pi(\tilde{\mathbf{z}}|\mathcal{S}, \boldsymbol{\theta}))$ is the gradient of the posterior density of interest, h is a tuning parameter, and $\mathbf{I}$ is the identity matrix. We note that modifications to the proposal covariance matrix have been proposed and are discussed below.

Now,

$$\log(\pi(\tilde{\mathbf{z}}|\mathcal{S}, \boldsymbol{\theta})) = constant - \frac{1}{2}||\tilde{\mathbf{z}}|| + \sum_{\mathbf{u}_j \in \mathcal{U}} (n_{\mathbf{u}_j}\mathbf{z}(\mathbf{u}_j) - |\mathbf{u}_j| \exp(\mathbf{x}^T(\mathbf{u}_j)\boldsymbol{\gamma} + z(\mathbf{u}_j)))$$

where $n_{\mathbf{u}_j}$ is the number of observed points in grid cell j. The Discrete Fourier Transform (DFT) can then be used to compute the gradient, $\nabla \log(\pi(\tilde{\mathbf{z}}|\mathcal{S}, \boldsymbol{\theta}))$.

See [142, 143] for further discussion of MCMC with Langevin-Hastings updates for LGCPs as well as applications to environmental spatial point patterns. Modifications to the specification of the proposal covariance matrix that further improve convergence of the MCMC algorithm have been proposed by [84]. They demonstrate the improvements of these methods through exhaustive simulations and applications for LGCPs as well as other models with intractable likelihoods. In particular, these modified specifications can increase efficiency with increases in both the dimension of the parameter space and correlation in the latent spatial processes.

The modifications to the proposal distribution of the MALA algorithm arise from using an approximation of the Fisher information matrix evaluated at the maximum likelihood estimate of the transformed process $\tilde{\mathbf{z}}$. Convergence can be further improved by modifying the tuning parameter h within the chain using an adaptive MCMC algorithm [172]. Details and an application using the modified proposals for the MALA algorithm are given in [54]. Lastly, code for fitting LGCPs using the algorithm outlined above is provided in [142]. The `lgcp` package in R [196] also includes functions for fitting LGCPs using MALA.

Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) [126, 146–148] algorithms pose an alternative to Metropolis-Hastings algorithms for Bayesian inference of LGCPs. In general, in order to learn about a high dimensional density, we must efficiently explore the parameter space for $\boldsymbol{\theta}$, and can not stay near the mode(s) since there is very little mass concentrated at these locations. Of course, there are also regions, away from the modes, where there is little mass. So, we need the informal concept of a "typical set." Such a set is a high mass set within the parameter space. That is, random walk Metropolis is a "guess and check" style approach where proposals are drawn from a specified distribution. High acceptance is not favorable for exploring the parameter space as it is too local, and low acceptance means the candidates are being drawn too far from the mass of the density.

This suggests that information about $p(\boldsymbol{\theta})$ and the gradient, or rather, a moving direction, should be leveraged to make efficient proposals with regard to exploring the density. Whereas MALA suggests that the direction should be back towards the mode, in high dimension, the mode contains very little mass. Instead, the goal here is to choose a direction to move to (or stay in) the foregoing typical set. This approach for a probabilistic specification with (*mode*, *gradient*, *typical set*), is analogous to a physical system with (*planet*, *gravitational field*, *orbit*). For a physical system, staying in orbit requires momentum to be added or subtracted in order to appropriately offset the gravitational pull. The HMC approach moves from probability densities to energies associated with the physical system, where latent momentum is added for each component of $\boldsymbol{\theta}$. The energy is specified through the *Hamiltonian* and Hamilton's equations are used to control the behavior of the associated physical system.

Here, we describe the simplest version of the HMC approach. First, for $\boldsymbol{\theta}$ of p-dimensions, the latent joint distribution is specified in $2p$-dimensions. We introduce p auxiliary parameters, $\boldsymbol{\phi}$, an associated vector of momentums. We set $\pi(\boldsymbol{\theta}, \boldsymbol{\phi}) = e^{-H(\boldsymbol{\theta},\boldsymbol{\phi})}$ where H is the Hamiltonian, or *energy*, at $(\boldsymbol{\theta}, \boldsymbol{\phi})$ The Hamiltonian is defined $H(\boldsymbol{\theta}, \boldsymbol{\phi}) = P(\boldsymbol{\theta}) + K(\boldsymbol{\phi}, \boldsymbol{\theta})$, where $P(\boldsymbol{\theta})$ is the potential energy and $K(\boldsymbol{\phi}, \boldsymbol{\theta})$ is the kinetic energy. In the density space, this is written $\pi(\boldsymbol{\theta}, \boldsymbol{\phi}) \propto e^{-P(\boldsymbol{\theta})} \times e^{-K(\boldsymbol{\phi},\boldsymbol{\theta})}$, which can be viewed as a conditional times a marginal form. The potential energy function is already determined through the density function for $\boldsymbol{\theta}$ which we wish to sample. So, we are left to choose the kinetic energy function, equivalently, the (conditional) density for the momentums. Every choice of kinetic energy, $K(\boldsymbol{\phi}, \boldsymbol{\theta})$, yields a new Hamiltonian transition that will interact differently with a given target distribution for $\boldsymbol{\theta}$. So, careful tuning is required to "stay in orbit'.' The Hamiltonian equations provide an approach to achieve this equilibrium, since Hamilton's equations maintain energy, equivalently, orbit. That is, they maintain a level set for $H(\boldsymbol{\theta}, \boldsymbol{\phi})$.

In implementation, the algorithm proceeds by drawing from $[\boldsymbol{\phi}|\boldsymbol{\theta}]$. The simplest choice is so-called *Euclidean-Gaussian* kinetic energy, $K(\boldsymbol{\phi}|\boldsymbol{\theta}) \propto \boldsymbol{\phi}^T M^{-1} \boldsymbol{\phi}$ implying a multivariate normal for the momentums, i.e., $\boldsymbol{\phi} \sim MVN(\mathbf{0}, M)$ Theory shows that a good choice for M is $\Sigma_{\boldsymbol{\theta}}$, which can be learned from the MCMC samples, bringing in some conditional dependence. A draw from $[\boldsymbol{\phi}|\boldsymbol{\theta}]$ implies a draw, h, from $[H(\boldsymbol{\theta}, \boldsymbol{\phi})|\boldsymbol{\theta}]$, which yields a fixed energy.

Then, to maintain the energy/orbit, we explore the level set determined by h through random trajectories over the level set. The level set is the set of $(\boldsymbol{\theta}^*, \boldsymbol{\phi}^*)$ such that $H(\boldsymbol{\theta}^*, \boldsymbol{\phi}^*) = h$. This set is determined by solutions to Hamilton's equa-

tions which don't exist in closed form. Therefore, the main obstruction to implementing the Hamiltonian Monte Carlo method is generating the Hamiltonian trajectories themselves. Aside from a few trivial examples, we cannot solve Hamilton's equations exactly and any implementation must instead solve them numerically.

In practice, the proposal step in HMC is more costly than the traditional proposal draws in usual MCMC, making the computational cost of a single iteration higher. The benefit, however, is a more efficient exploration of the space for the high dimensional distribution, making it more efficient overall. The HMC algorithm is built into the STAN software packages [34], which is easily accessible and adaptable to a wide range of models. As with all MCMC algorithms, HMC can still struggle to capture isolated local maxima that do not fall in the typical set. See [84] for additional details and modifications to the proposal covariance matrix that further improve convergence of the MCMC algorithm.

Integrated nested Laplace approximation

The integrated nested Laplace approximation (INLA) approach to Bayesian inference for LGCP models circumvents the challenges of programming, fine-tuning, and running computationally challenging MCMC algorithms [102, 104]. INLA has become a popular method for parameter inference and model fitted for the class of latent Gaussian models, of which LGCPs are a special case [176]. The main aim of INLA is to approximate the posterior distributions; it is not a sampling scheme like the algorithms discussed above. For the LGCP, INLA will approximate, marginally, the posterior distribution of the latent field as well as the model parameters. The approximate (marginal) posterior distributions can then be used to obtain estimates of means, standard deviations, etc.

The speed of model inference using INLA relies on approximating the continuous Gaussian process with a discrete Gaussian Markov random field (GMRF). Recall the LGCP specification where the intensity function of the spatial point pattern is written as $\log\lambda(\mathbf{s}) = \mathbf{x}^T(\mathbf{s})\boldsymbol{\gamma} + z(\mathbf{s})$ and $z(\mathbf{s})$ is a Gaussian process. INLA requires the Gaussian process, which is capturing the spatial dependence in the process not accounted for the covariates, to be approximated by a GMRF. The GMRF is specified through its precision matrix Q (the inverse of the covariance matrix, i.e., $\Sigma^{-1} = \mathrm{Q}$) and when sparse, results in a large reduction in computation time. For example, Q may be specified as a precision matrix of a GMRF with first- or second-order neighbor structure. This differs from the spatial covariance matrix of the Gaussian process, $z(\mathbf{s})$, which is specified by a covariance function, such as a Matérn covariance function, resulting in a dense matrix.

Below we provide some details of the Laplace approximation, first generally, and then as used in obtaining posterior inference. Laplace approximations are used to approximate $\int_{-\infty}^{\infty} g(\mathbf{v})d\mathbf{v}$ where $g(\cdot) > 0$. To begin, write $h(\mathbf{v}) = \log g(\mathbf{v})$ and expand $h(\mathbf{v})$ in a Taylor series as $h(\mathbf{v}) \approx h(\mathbf{v}^*) + (\mathbf{v} - \mathbf{v}^*)^T \nabla h(\mathbf{v})|_{\mathbf{v}=\mathbf{v}^*} - (\mathbf{v} - \mathbf{v}^*)^T H(\mathbf{v}^*)(\mathbf{v} - \mathbf{v}^*)$. Here, $\mathbf{v}^*$ is the mode and H is the Hessian for h. Then, plugging these values in and integrating out the d-dim multivariate normal for $\mathbf{v}$ yields the approximation to the integral, $(2\pi)^{d/2}|H(\mathbf{v}^*)|^{\cdot 5} e^{h(\mathbf{v}^*)}$.

In practice, using the Laplace approximation, and thus, INLA, to obtain posterior inference, begins with writing the joint posterior distribution $f(\boldsymbol{\theta}|\mathbf{y}) \propto e^{\log(f(\mathbf{y}|\boldsymbol{\theta})f(\boldsymbol{\theta}))}$. The Laplace approximation entails using a second-order expansion of $\log(f(\mathbf{y}|\boldsymbol{\theta})f(\boldsymbol{\theta}))$ to create what is essentially a multivariate normal density approximation for the

posterior. In a hierarchical setting this can be extended to

$$\Pi_i f(y_i|z_i, \boldsymbol{\theta}_y) f(\mathbf{z}|\boldsymbol{\theta}_z) f(\boldsymbol{\theta})$$

where, typically $\mathbf{z}$ are latent variables and $f(\mathbf{z}|\boldsymbol{\theta}_\mathbf{z})$ is $MVN(\mathbf{0}, \Sigma_z)$. Now, the posterior of interest is $f(\boldsymbol{\theta}, \mathbf{z}|\mathbf{y}) \propto \Pi_i f(y_i|z_i, \boldsymbol{\theta}_y) f(\mathbf{z}|\boldsymbol{\theta}_z) f(\boldsymbol{\theta})$. This approach requires that we can assume $\boldsymbol{\theta}$ is low dimensional in order to obtain posteriors for each component of $\boldsymbol{\theta}$. More importantly, $\boldsymbol{\theta}$ must be low dimensional to obtain $f(\mathbf{z}|\mathbf{y}) = \int f(\mathbf{z}|\boldsymbol{\theta}, \mathbf{y}) f(\boldsymbol{\theta}|\mathbf{y}) d\boldsymbol{\theta}$.

The first step is to obtain the Laplace approximation for the full conditional, $f(\mathbf{z}|\boldsymbol{\theta}, \mathbf{y})$ for a given $\boldsymbol{\theta}$, which requires first computing the mode of the full conditional, $\mathbf{z}^*$, and Hessian. Let $\tilde{f}(\mathbf{z}|\boldsymbol{\theta}, \mathbf{y})$ denote this approximation. Next, $f(\boldsymbol{\theta}|\mathbf{y}) \propto \frac{f(\mathbf{y}, \mathbf{z}, \boldsymbol{\theta})}{f(\mathbf{z}|\boldsymbol{\theta}, \mathbf{y})}$ regardless of $\mathbf{z}$. Then, our approximation to the posterior of $\boldsymbol{\theta}$ can be written $\tilde{f}(\boldsymbol{\theta}|\mathbf{y}) \propto \frac{f(\mathbf{y}, \mathbf{z}, \boldsymbol{\theta})}{\tilde{f}(\mathbf{z}|\boldsymbol{\theta}, \mathbf{y})}|_{\mathbf{z}=\mathbf{z}^*}$. Lastly, $\tilde{f}(\mathbf{z}|\mathbf{y}) = \int \tilde{f}(\mathbf{z}|\boldsymbol{\theta}, \mathbf{y}) \tilde{f}(\boldsymbol{\theta}|\mathbf{y}) d\boldsymbol{\theta}$, which can be approximate using a grid over $\boldsymbol{\theta}$-space and replacing the integral by a sum approximation. From this, it is easy to see the computational challenge when the dimension of $\boldsymbol{\theta}$ gets large, say, more than 8 to 10.

The `inla` package in R contains functions for fitting spatial point process models using INLA under various model specifications. For example, the (log)intensity function can be specified with a variety of location-specific covariates and/or spatially structured random effects observed on different spatial scales. Some of these model specifications are discussed in detail in [102]. Multiple alternatives also exist within the `inla` packages for modeling the spatial dependence in the latent GMRF, such as first- and second-order random walks or conditional autoregressive models. Extensions that allow for an irregular grid for the GMRF and neighborhood dependence structure [125, 186] are also available.

Comparisons between MCMC and INLA approaches to inference for LGCP models can be found in [167, 196]. In general, there are advantages and disadvantages to each of these approaches. MCMC, and, thus, fully Bayesian approaches benefit from producing full Bayesian inference; namely, full joint posterior distributions which can be used to obtain parameter inference (means, credible intervals) and to obtain full posterior predictive distributions. In addition, inference is obtained for the Gaussian process with full a covariance matrix, rather than from a process approximated by a GMRF. Overwhelmingly, computational efficiency in obtaining inference is the main benefit of INLA. In particular, model comparison and validation greatly benefit from the rapid computation time of INLA, making it a convenient method for assessing a collection of candidate models. As above, INLA will struggle, however, when the dimension of $\boldsymbol{\theta}$ grows large in a point process model because of the challenge of the above numerical integration over $\boldsymbol{\theta}$.

4.2.3. Application: Modeling earthquake epicenters with a LGCP

We illustrate some of the above concepts using a dataset of earthquake epicenter locations. The earthquake data comes from USGS Earthquake Hazards Program, which is publicly available[4]. The data in this analysis consist of 1035 earthquakes of magnitude ≥ 2.5 occurring in 2017 with epicenters in northern Oklahoma and southern Kansas, as depicted in Figure 4.1.

The intensity function for the LGCP is specified as

$$\log\lambda(\mathbf{s}) = \mu + z(\mathbf{s}).$$

[4]https://earthquake.usgs.gov/earthquakes/

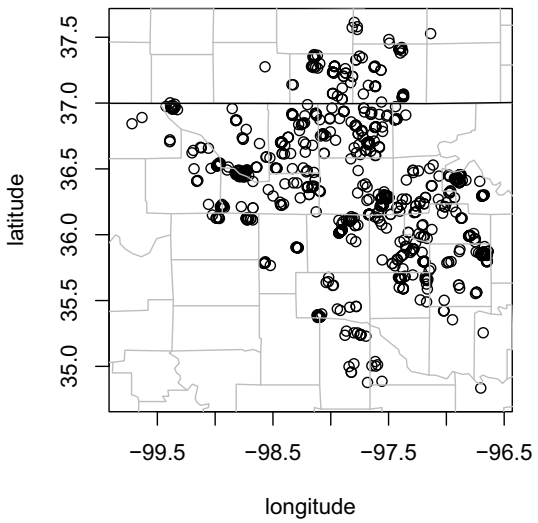

FIG 4.1. *Epicenters of earthquakes with magnitude greater than 2.5 during 2017 across northern Oklahoma and southern Kansas.*

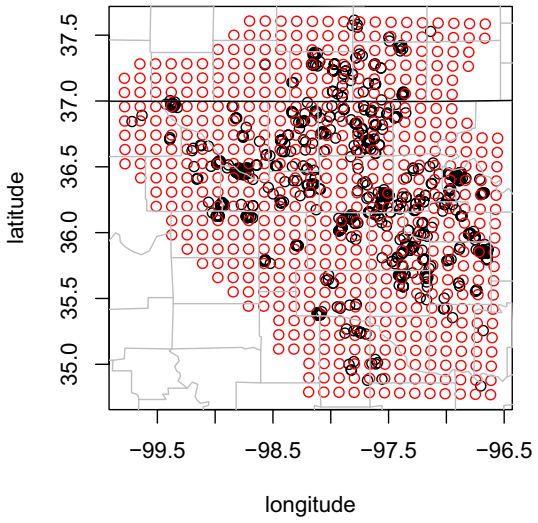

FIG 4.2. *The grid of representative points used for model fitting and inference. The points were located on a 12×12 km grid using UTM coordinates, however they are projected here to latitude and longitude.*

That is, we have no regressors but $z(\mathbf{s})$ is a Gaussian process with mean $-\sigma^2/2$ and covariance specified by the exponential covariance function where $cov(z(\mathbf{s}), z(\mathbf{s}')) = \sigma^2 \exp^{-||\mathbf{s}-\mathbf{s}'||/\phi}$, where σ^2 is the variance and ϕ is the range parameter.

A collection of representative points was used for model fitting and inference as discussed in Section 4.1.3. The locations of the earthquake epicenters were projected from latitude and longitude to UTM coordinates. Distances between points were computed using Euclidian distance of the UTM coordinates, given in kilometers (km). The maximum distance between epicenters was 390 km. The locations of the representative points were defined on a 12×12 km grid as shown in Figure 4.2. A total of 529 representative points were used for model fitting and inference,

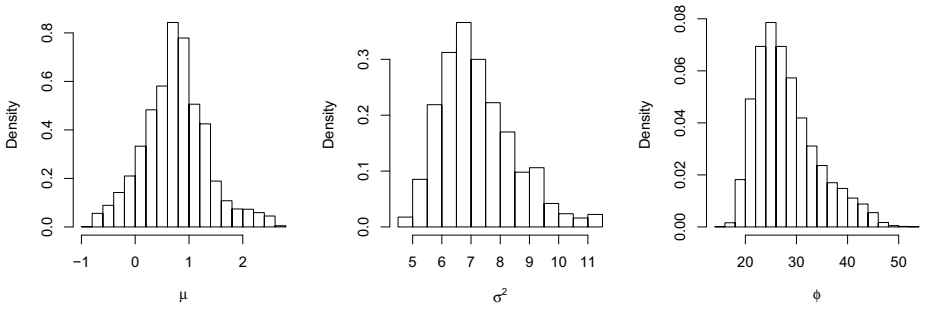

FIG 4.3. *Posterior distributions of the parameters* μ, σ^2, *and* ϕ.

as outlined in Section 4.2.2. Specifically, realizations of the Gaussian process were obtained at these representative points and used for numerical integration when evaluating the likelihood.

Inference was implemented within a Bayesian framework. Independent prior distributions were assigned to each of the three parameters, μ, σ^2, and ϕ. Specifically, a noninformative prior was assigned to μ such that $\mu \sim N(0, 100^2)$. For the covariance parameters, $\sigma^2 \sim IG(2, 2)$ and $\phi \sim Unif(8, 60)$. The parametrization of the inverse gamma distribution is highly noninformative; σ^2 has non-finite variance. The specification of the uniform distribution is based on the spatial domain of the data, and controls the effective range to be between approximately 24 and 180 km. These values were chosen such that the effective range is at least two times the resolution of the grid of representative points and less than approximately half the maximum distance over the domain.

Model fitting was carried out using a MCMC algorithm. Posterior samples were obtained for each of the three parameters using Metropolis-Hastings algorithms. Samples of the Gaussian process, $\mathbf{z}(\mathbf{s})$, at the set of representative points were obtained using the elliptical slice sampler [144, and Section 4.2.2]. The chain was run for 100,000 iterations. The first half of the chain was disregarded as burn-in and every 10th iteration post burn-in was retained for posterior inference in order to reduce dependence in the retained samples. Marginal posterior distributions for the parameters μ, σ^2, and ϕ are shown in Figure 4.3.

The posterior mean intensity surface is shown in Figure 4.4, highlighting areas of increased earthquake activity. Figure 4.5 shows two simulated realizations of the spatial point pattern using samples from the posterior distribution of the model parameter given the data following the procedure outlined in Section 4.1.3. By generating a collection of realizations of the point pattern using the posterior distribution of the model parameters, we also obtained samples from the posterior predictive distribution of $N(D)$, the number of points in the domain. The posterior mean of $N(D)$ was 1041.2 and the 90% credible interval was (987.5, 1109.1). Recall that the true number of earthquakes in the dataset was 1035, and, thus, our credible interval captures the true number of events.

The numbers of observed events for the three counties, Blaine County (Figure 4.5, green), Garfield County (Figure 4.5, red), and Oklahoma County (Figure 4.5, blue), were 15, 92, and 54, respectively. We obtained posterior predictive distributions of $N(B)$, $N(G)$, and $N(O)$, the marginal distribution of the number of events in Blaine, Garfield, and Oklahoma County. The posterior means and 90% credible

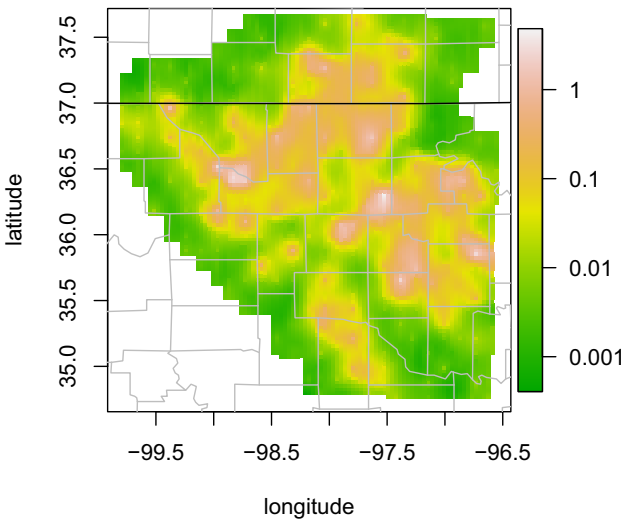

FIG 4.4. *Posterior mean intensity surface.*

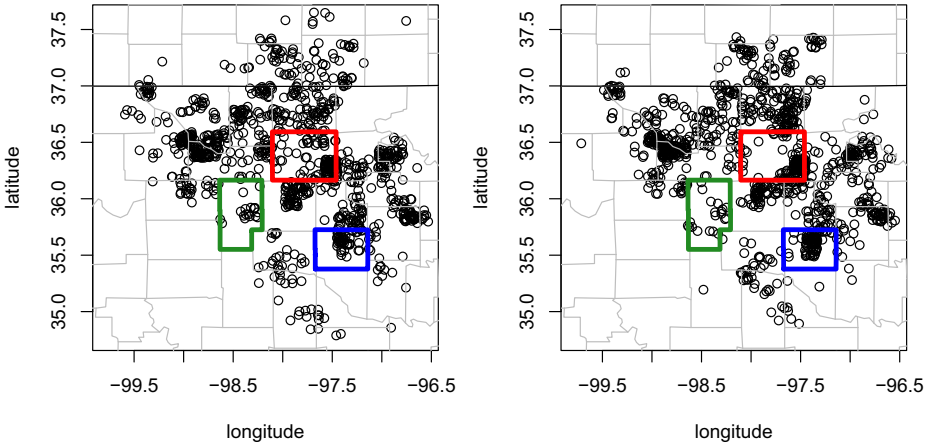

FIG 4.5. *Two realizations of the spatial point pattern from the posterior distribution. Blaine County (green), Garfield County (red), and Oklahoma County (blue) are also highlighted.*

intervals, from the LGCP model, for these random variables are given in Table 4.1. Each credible interval captured the true number of events for the county. In addition, we obtained posterior predictive distributions of $N(B)$, $N(G)$, and $N(O)$ conditional on the number of events in Oklahoma. For each conditional distribution, we set the number of events in Oklahoma equal to 930, which was the number of observed events in the state. The posterior mean and 90% credible intervals for $N(B)$, $N(G)$, and $N(O)$ are also given in Table 4.1. The marginal and conditional means for each county are similar. However, these intervals indicate that the conditional distributions have smaller variance, as expected given a fixed total.

To assess the predictive performance of the model, we created training and test spatial point patterns using p-thinning, discussed in Section 4.1.7. We thinned the observed point pattern using $p(\mathbf{s}) = 0.5$ for all $\mathbf{s} \in \mathcal{S}$, which resulted in a training

TABLE 4.1
Posterior mean (90% CI) of the posterior predictive distributions of $N(B)$, $N(G)$, and $N(O)$ from the LGCP model.

	Observed	Marginal	Conditional
$N(B)$	15	16.6 (8.0 26.1)	15.8 (8.0, 25.0)
$N(G)$	92	92.9 (71.8 115.1)	91. 5 (69.9, 111.0)
$N(O)$	54	53.2 (37.0, 69.0)	54.5 (39.0, 68.1)

dataset of 500 points and a test dataset of 535 points. We refitted the LGCP model to the training dataset, along with an HPP model for comparison. The intensity of the HPP model was specified as $\log\lambda(\mathbf{s}) = \mu$ for all $\mathbf{s} \in D$. The same prior distributions were assigned to each of the parameters; $\mu \sim N(0, 100^2)$ for the HPP. Both models were fitted using MCMC.

Figure 4.6 illustrates the posterior predictive distributions of residuals, R_{pred} as outlined in Section 3.1. For the three counties, Blaine, Garfield, and Oklahoma, we obtain samples from the posterior predictive distribution of the residuals under the two models. For Blaine county, this is computed as $R_{pred}(B) = N_{obs}(B) - N_{pred}(B)$. The distributions for the other two counties were obtained analogously. Figure 4.6 shows boxplots of the predictive distribution under each model for the three counties. The predictive distributions of the residuals under the LGCP are all centered at approximately 0, whereas the predictive distributions under the HPP are far from 0. The HPP model overestimates the number of events in Blaine County while underestimating the number of events in both Garfield and Oklahoma County.

We also compared the LGCP and HPP models based on empirical coverage. For radial distances 6kms, 12kms, and 20kms, we randomly generated 400 points within the domain and defined circular regions. Then, using the test point pattern, we computed the true number of events within each circular region. We obtained the predictive distribution of the number of events within each circular region for each model and computed the 90% credible interval of the residual, R_{pred}. Empirical coverage was computed as the proportion of regions having 90% credible intervals of R_{pred} containing 0. The empirical coverage for the HPP model was 0.915, 0.585, and 0.350 for the three radial distances, indicating that the HPP is unable to capture the clustering behavior in the data for moderate or large regions. Under the LGCP model, empirical coverage was 0.945, 0.880, and 0.895, much closer to the nominal level for all distances.

Figure 4.7 consider the G function, showing four estimates. One is the empirical $G(d)$ function with edge correction (Section 2.2). The second arises as the estimated $G(d)$ function under the HPP. That is, we insert the posterior mean for λ into the explicit expression for $G(d)$. The third and fourth show two estimators arising under the LGCP model. Formally, the G function is only applicable to stationary processes (again, Section 2.2) but since our model is "marginally" stationary, we compute it for the LGCP. The one in blue was obtained by applying the empirical estimator of $G(d)$ in Section 2.2 to posterior point pattern samples under the model and averaging the resulting functions. The one with a dashed line is the fully model-based estimator arising from using the formal Bayesian edge correction presented in Section 3.2.3. The two versions of the estimator are indistinguishable, suggesting that the more computationally demanding edge-corrected version may not be worth the effort to obtain. More importantly, due to the clustering in earthquake locations (departure from complete spatial randomness), the LGCP function is above the HPP function for all distances. Finally, we see that the LGCP G function lies

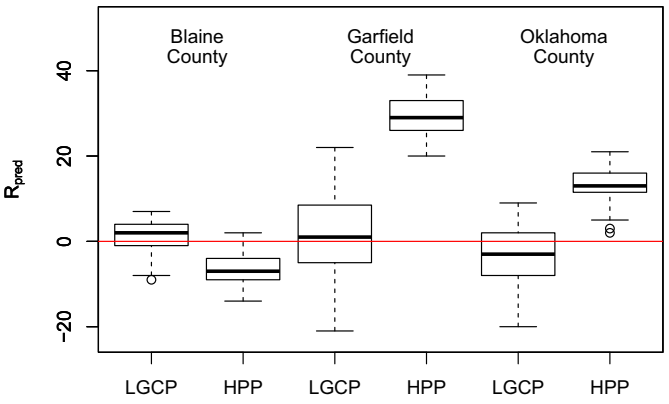

FIG 4.6. *Posterior predictive distributions of residuals R_{pred} for the LGCP and HPP models for the three counties, Blaine, Garfield, and Oklahoma, shown in Figure 4.5.*

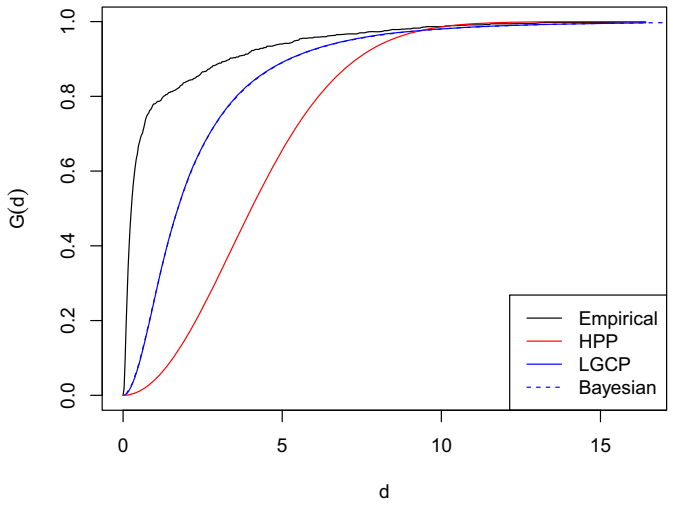

FIG 4.7. *The empirical G function as well as the estimated G function under the HPP and LGCP models. Also shown is the estimated G function with Bayesian edge correction as computed in Section 3.2.3.*

below the customary empirical G function. This does not criticize the LGCP model since the latter is conditionally nonstationary; comparison between the curves is not meaningful. The empirical G function only criticizes the HPP as a choice of stationary model.

4.2.4. Bayesian fitting of Gibbs processes

There are two primary strategies in the literature for fitting Bayesian models to Gibbs processes. One proceeds by using pseudo-maximum likelihood, a fairly easy but approximate approach. The second extends to enable fully Bayesian inference for Gibbs processes using data augmentation strategies. We briefly review both approaches here.

Pseudo-likelihood approach for Gibbs processes

To begin, we offer a maximum pseudo-likelihood alternative to likelihood based inference for the Gibbs process as a way to avoid the challenges of the intractable likelihood function [10, 27, 30, 88, 112, 179]. As outlined in Section 4.2.1, the pseudo-likelihood can be written as the product of the Papangelou conditional intensities. The main advantage of using the conditional intensity function is that it is easy to compute and the pseudo-likelihood can be maximized. Maximum pseudo-likelihood estimates are obtained by maximizing

$$PL(\boldsymbol{\theta};\mathcal{S}) = e^{-\int_D \lambda_{\boldsymbol{\theta}}(\mathbf{u}|\mathcal{S})d\mathbf{u}} \prod_{\mathbf{s}_i \in \mathcal{S}} \lambda_{\boldsymbol{\theta}}(\mathbf{s}_i|\mathcal{S}).$$

For a nonhomogeneous Gibbs process, we might define the conditional intensity as

$$\lambda(\mathbf{s}|\mathcal{S}) = \exp\left(-\left(\mathbf{x}^T(\mathbf{s})\boldsymbol{\gamma} + \sum_{\mathbf{s}_i \in \mathcal{S}} ||\mathbf{s}-\mathbf{s}_i||^2/\tau^2\right)\right)$$

where $\mathbf{x}(\mathbf{s})$ are location specific covariates, $\boldsymbol{\gamma}$ is a vector of coefficients, $||\mathbf{s}-\mathbf{s}_i||^2$ is the squared distance between $\mathbf{s}$ and $\mathbf{s}_i$, and τ^2 controls the amount of inhibition. Except for the most simple specifications of the Gibbs process, the integral $\int_D \lambda_{\boldsymbol{\theta}}(\mathbf{u}|\mathcal{S})d\mathbf{u}$ will have to be approximated using numerical integration. In addition, maximizing the pseudo-likelihood will require having $\mathbf{x}(\mathbf{s})$ available at all representative points used in the numerical integration.

The disadvantage of this approach is that for small sample sizes, the pseudo-likelihood has been shown to be statistically inefficient; the maximum pseudo-likelihood estimates have greater bias than maximum likelihood estimates [13]. Fitting Gibbs processes by maximizing the pseudo-likelihood function can be conducted in `spatstat` given that the conditional intensity is defined to be log-linear in the parameters. See [10, chapter 13] for details of the various forms of first- and second-order terms available in `spatstat` for the conditional intensity function.

An auxiliary variable method for Bayesian fitting of Gibbs processes

The normalizing constant for Gibbs processes is a function of the model parameters, meaning it cannot be avoided in likelihood-based inference (e.g., MCMC with Metropolis-Hastings algorithm). Due to the statistical inefficiencies of maximum pseudo-likelihood, there is reason to explore more complex algorithms for fitting Gibbs process models. In a sequence of papers [25, 26, 140], efficient, fully Bayesian model fitting strategies using auxiliary variables are proposed which avoid having to compute the normalizing constant in the Metropolis-Hastings ratio. This approach is a data augmentation strategy; such approaches have been developed extensively for Bayesian inference and iterative samplings algorithms for a variety of model structures.

Letting $\boldsymbol{\theta}$ denote the parameters of the model and $\mathcal{S}$ the data, we seek to obtain samples from $[\boldsymbol{\theta}|\mathcal{S}] \propto [\mathcal{S}|\boldsymbol{\theta}][\boldsymbol{\theta}]$. The challenge here is $[\mathcal{S}|\boldsymbol{\theta}] = g_{\boldsymbol{\theta}}(\mathcal{S})/C_{\boldsymbol{\theta}}$ is only known up to a normalizing constant $C_{\boldsymbol{\theta}}$, which is a function of $\boldsymbol{\theta}$. Whereas $C_{\boldsymbol{\theta}}$ drops out when writing the Gibbs process using Papangelou conditional intensities, it needs to be evaluated in the traditional Metropolis-Hastings algorithm.

Introduce the auxiliary variable $\mathcal{U}$ with density $[\mathcal{U}|\boldsymbol{\theta},\mathcal{S}]$ that takes the same support as the data, $\mathcal{S}$. Then the Metropolis-Hastings ratio takes the form

$$m(\boldsymbol{\theta}',\mathcal{U}';\boldsymbol{\theta},\mathcal{U}) = min\left(1, \frac{[\mathcal{U}'|\boldsymbol{\theta}',\mathcal{S}]g_{\boldsymbol{\theta}'}(\mathcal{S})[\boldsymbol{\theta}']C_{\boldsymbol{\theta}}[\boldsymbol{\theta},\mathcal{U}|\boldsymbol{\theta}',\mathcal{U}']}{[\mathcal{U}|\boldsymbol{\theta},\mathcal{S}]g_{\boldsymbol{\theta}}(\mathcal{S})[\boldsymbol{\theta}]C_{\boldsymbol{\theta}'}[\boldsymbol{\theta}',\mathcal{U}'|\boldsymbol{\theta},\mathcal{U}]}\right)$$

Now, partitioning the joint proposal as $[\boldsymbol{\theta}',\mathcal{U}'|\boldsymbol{\theta},\mathcal{U}] = [\mathcal{U}'|\boldsymbol{\theta}',\boldsymbol{\theta},\mathcal{U}][\boldsymbol{\theta}'|\boldsymbol{\theta},\mathcal{U}]$, we can let $[\mathcal{U}'|\boldsymbol{\theta}',\boldsymbol{\theta},\mathcal{U}] = g_{\boldsymbol{\theta}'}(\mathcal{U}')/C_{\boldsymbol{\theta}'}$ and simplify $[\boldsymbol{\theta}'|\boldsymbol{\theta},\mathcal{U}]$ to $[\boldsymbol{\theta}'|\boldsymbol{\theta}]$. Plugging these in we find that the former Metropolis-Hastings ratio is now

$$m(\boldsymbol{\theta}',\mathcal{U}';\boldsymbol{\theta},\mathcal{U}) = min\left(1, \frac{[\mathcal{U}'|\boldsymbol{\theta}',\mathcal{S}]g_{\boldsymbol{\theta}}(\mathcal{U})g_{\boldsymbol{\theta}'}(\mathcal{S})[\boldsymbol{\theta}'][\boldsymbol{\theta}|\boldsymbol{\theta}',\mathcal{U}']}{[\mathcal{U}|\boldsymbol{\theta},\mathcal{S}]g_{\boldsymbol{\theta}'}(\mathcal{U})g_{\boldsymbol{\theta}}(\mathcal{S})[\boldsymbol{\theta}][\boldsymbol{\theta}'|\boldsymbol{\theta},\mathcal{U}]}\right)$$

where the normalizing constants have dropped out. A simple choices for $[\mathcal{U}|\boldsymbol{\theta},\mathcal{S}]$ would be an HPP with intensity a function of $\mathcal{S}$, taken as $\frac{N(D)}{|D|}$, where $N(D)$ is the number of points in $\mathcal{S}$. Sampling proposals $\mathcal{U}'$ from $g_{\boldsymbol{\theta}'}(\mathcal{U}')/C_{\boldsymbol{\theta}'}$ is likely inefficient and computationally expensive; [25] suggest using perfect sampling to draw the proposal $\mathcal{U}'$. Sampling $[\boldsymbol{\theta}'|\boldsymbol{\theta}]$ could be done using a traditional random walk.

Bayesian fitting of Neyman Scott processes

Likelihood inference based on MCMC methods is easier for parametric Gibbs point process models than for the class of Cox processes, which includes the Neyman-Scott process [143]. Under a fully Bayesian approach to inference for the Neyman Scott process, the nodes of the parent process are treated as unknown parameters to be estimated along with the parameters of the offspring process. Let $[\mathcal{U}|\lambda]$ denote the density of the parent process of nodes $\mathcal{U}$ with intensity λ for the region D.

Here, we will assume an HPP for the parent process. Then, the offspring process density given the knowledge of the parent locations $\mathcal{U}$, is denoted $[\mathcal{S}|\mathcal{U},\mu,\tau^2]$ where μ and τ^2 are parameters of offspring process. The expected number of offspring for each parent is μ and τ^2, along with the kernel function, controls the dispersal of the offspring process conditional on the parent nodes. This conditional density of the offspring can be written

$$[\mathcal{S}|\mathcal{U},\mu,\tau^2] = exp\left(-|D| - \int_D \tilde{f}(\mathbf{s})ds\right)\prod_{\mathbf{s}\in\mathcal{S}}\tilde{f}(\mathbf{s})$$

where $\tilde{f}(\mathbf{s}) = \delta\sum_{\mathbf{u}\in\mathcal{U}} f(\mathbf{s}-\mathbf{u},\tau^2)$ and the kernel $f(\mathbf{s}-\mathbf{u},\tau^2)$ is a probability density function parameterized by τ^2. For a bivariate Gaussian intensity specification of the offspring process (making this a modified Thomas process), $f(\mathbf{s}-\mathbf{u},\tau^2\mathbf{I})$ is a mean 0 bivariate normal density with variance τ^2. The joint posterior distribution of interest can then be written

$$[\mathcal{U},\lambda,\mu,\tau^2|\mathcal{S}] = [\mathcal{S}|\mathcal{U},\mu,\tau^2][\mathcal{U}|\lambda][\lambda,\mu,\tau^2].$$

An MCMC simulation algorithm can be used to obtain posterior samples. Specifically, samples of the parameters λ, μ, and τ^2 can be obtained through traditional Metropolis-Hastings algorithms. Samples of $\mathcal{U}$, however, require more advanced algorithms for simulation in order to be efficient in practice. Common choices for obtaining updates of $\mathcal{U}$ uses the spatial birth-death algorithm [142, Chapter 11] or its extension, the spatial birth-death-move algorithm, and the coupling-from-the-past algorithm [114, 143, 202]. For multimodal densities, these algorithms might

result in slow mixing and could be incorporated into a simulated tempering procedure [82, 133, 143]. Additional details for Bayesian inference for Neyman-Scott processes and the wider class of Cox processes along with examples can be found in [94, 142, 143, 208]. As a final comment, we suggest approximate Bayesian computation (ABC) as a computationally attractive alternative approach for inference for spatial point processes with point-level dependence. Details with regard to the ABC approach are given in the next section.

4.3. Computational strategies for inhibition and clustering processes

As discussed in Section 2.3, the most widely adopted class of spatial point pattern models is the nonhomogeneous Poisson processes (NHPP) or, more generally, the log Gaussian Cox processes (LGCP) (see [142] and references therein). Such models assume conditionally independent event locations given the process intensity. However, in many applications, we find evidence of clustering or of inhibition, following Section 2.5. Here, we focus on inhibition and refer to associated models as repulsive spatial point processes. Most common in this setting are Gibbs point processes (here, denoted as GPP) [see, e.g., 101, and Section 2.5]. These processes specify the joint location density, up to normalizing constant, in the form of a Gibbs distribution, introducing potentials on cliques of order 1 but also potentials on cliques of higher order, which capture interaction. The most familiar example in the literature is the Strauss process and its extreme version, the hardcore process [106, 193]. An attractive alternative is the determinantal point process (here, denoted as DPP). Though these processes have some history in the mathematics and physics communities, they have only recently come to the attention of the statistical community thanks, most notably, to recent efforts of Jesper Møller and colleagues. See, for instance, [119].

Approximate Bayes computation (ABC)to fit both classes of models is discussed in detail in [184]. Here, we confine ourselves to ABC model fitting for GPPs. In the literature, Markov chain Monte Carlo (MCMC) model fitting has been proposed for GPPs as discussed briefly in Section 4.2.4 and in greater detail in [2, 3, 86, 140]. The algorithms are complex and implementation can often result in poorly behaved chains with concerns regarding posterior convergence. Here, we demonstrate much simpler model fitting using ABC. ABC is particularly promising for GPPs since these processes allow straightforward simulation of point pattern realizations given parameter values. Additionally, such simulation facilitates posterior inference as well as consideration of model adequacy and model comparison, as we have argued in the previous chapters.

ABC methods are now attracting considerable attention [19, 130, 132, 162, 187]. The scope of ABC applications is also increasing, e.g., population genetics [19], multidimensional stochastic differential equations [161], macroparasite populations [58] and the evolution of HIV-AIDS [31]. As for spatial statistical applications, [64] implemented ABC for max-stable processes in order to model spatial extremes and [188] applied ABC with functional summary statistics to fit a cluster and marked spatial point process.

Supplementing the discussion in Section 2.5, the Gibbs point process offers a mechanistic model with interpretable parameters and has been used for modeling repulsive point patterns in environmental science and biology [86, 113, 134, 192]). The main challenge for fitting models using these processes is that likelihoods have intractable normalizing constants which depend on parameters. Hence, likelihood inference is difficult [142] and standard Bayesian inference using Markov chain Monte

Carlo (MCMC)) is not directly available. Maximum pseudo-likelihood estimation was proposed in [28, 30] and [105] (Section 4.2.4). These estimators show poor performance in the presence of strong inhibition [e.g., 100]. In the Bayesian framework, a clever auxiliary variable MCMC strategy was developed by [140] (again, Section 4.4.2) and extended by [145]. However, perfect simulation within the MCMC algorithm is needed along with approximations.

Here, following [184], we show how to implement the ABC algorithm for fitting GPP's based on ABC-MCMC proposed by [132]. We include discussion about how to choose summary statistics, kernel functions, and tune user-specific parameters. Again, the attractiveness of ABC for repulsive point processes rests in the fact that it is straightforward to generate realizations under these point processes given parameter values. This enables the ABC presumption: randomly draw parameters and then randomly draw point patterns given parameters. Further, as has been our theme for the entire monograph, with posterior inference achieved for the model parameters, we can use composition sampling to draw posterior predictive point patterns, enabling posterior inference about features of point patterns realized under the models. In addition, through these posterior samples of point patterns, we can propose model assessment for repulsive point processes, following the discussion in [121] and Section 3.2.

Briefly reviewing, for GPPs, since $c_0(\boldsymbol{\theta})$ cancels out of the Papangelou conditional intensity, the pseudo-likelihood, in log form, $\log PL(\mathcal{S}|\boldsymbol{\theta}) = -\int_D \lambda(u|\mathcal{S},\boldsymbol{\theta})du + \sum_i \log\lambda(\mathbf{s}_i|\mathcal{S},\boldsymbol{\theta})$, has been proposed [30] yielding the maximum pseudo-likelihood estimator. Although the maximum pseudo-likelihood estimator is consistent [see 105], the performance of the maximum pseudo-likelihood estimator is poor in the case of a small number of points and strong repulsion [100].

The pseudo-likelihood can be used for MCMC in the Bayesian framework [e.g., 113]. [140] and [26] proposed an auxiliary variable MCMC method (AVM) where, conveniently, $c_0(\boldsymbol{\theta})$ cancels out of the Hastings ratio. The challenge is to obtain the conditional density of the auxiliary variable. A partially ordered Markov model is used to approximate this density. A similar approach is the exchange algorithm proposed by [145]. Both algorithms require perfect simulation from the likelihood given $\boldsymbol{\theta}$ for each MCMC iteration. Although, perfect simulation is available for GPPs, this step can be computationally burdensome and obtaining a good acceptance rate is difficult.

More recently, [123] proposed the double MCMC algorithm which does not require perfect simulation from the likelihood. It only requires simulation from the Markov transition kernel and is faster than the AVM and exchange algorithms but convergence to the stationary distribution is not guaranteed. [86] implement this algorithm, with an application, for a class of GPP models.

For ABC, we need to simulate realizations of a GPP given parameter values. This is usually based on a birth-and-death algorithm [e.g., 81, 101, 142]. An alternative simulation algorithm to generate the point pattern is "dominated coupling from the past" [111] as implemented by [24] and [25]. This algorithm can be called as a default setting in `spatstat` [12].

4.3.1. Approximate Bayesian Computation for repulsive point processes

Let $\mathcal{S}_{obs}$ be the observed point pattern and $\mathcal{S}^*$ be a simulated point pattern. For a Bayesian model of the form $\pi(\mathcal{S}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$, ABC consists of three steps: (1) generate $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta})$, (2) generate $\mathcal{S}^* \sim \pi(\mathcal{S}|\boldsymbol{\theta})$, (3) compare summary statistics for the

generated $\mathcal{S}*$, $\mathbf{T}(\mathcal{S}^*)$, with those of the observed data, $\mathbf{T}(\mathcal{S}_{obs})$, and accept $\boldsymbol{\theta}$ if $\Psi(\mathbf{T}(\mathcal{S}), \mathbf{T}(\mathcal{S}_{obs})) < \epsilon$ for a selected *kernel(distance)* measure Ψ. Accepted $\boldsymbol{\theta}$ are samples from the approximate posterior distribution, $\pi_\epsilon(\boldsymbol{\theta}|\mathbf{T}(\mathcal{S}_{obs}))$. Approximation error relative to the exact posterior distribution $\pi(\boldsymbol{\theta}|\mathcal{S}_{obs})$ comes from the choice of $\mathbf{T}(\cdot)$, Ψ, and ϵ. If $\mathbf{T}(\cdot)$ is a sufficient statistic for $\boldsymbol{\theta}$, then $\pi(\boldsymbol{\theta}|\mathbf{T}(\mathcal{S}_{obs})) = \pi(\boldsymbol{\theta}|\mathcal{S}_{obs})$ and, given Ψ, the only approximation error is from ϵ. Since sufficient statistics are not usually available, the selection of informative summary statistics $\mathbf{T}(\cdot)$ is critically important. Small values of ϵ are desired but require more simulation of $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta})$ and $\mathcal{S} \sim \pi(\mathcal{S}|\boldsymbol{\theta})$ in order to retain $\mathcal{S}$. Again, with regard to simulation of $\mathcal{S}$ for the GPP, we can utilize perfect simulation (Section 4.1.5).

Summary Statistics

The Strauss process was discussed in Section 2.5. It is a Gibbs point process (GPP) with density often written as [e.g., 142]

$$\pi(\mathcal{S}) = \beta^{N(\mathcal{S})}\gamma^{s_R(\mathcal{S})}/c(\beta,\gamma), \quad \mathcal{S} \subset D \tag{4.1}$$

where $\beta > 0$, $0 \leq \gamma \leq 1$, $N(\mathcal{S})$ is the number of points, and $c(\beta,\gamma)$ is the normalizing constant. Here,

$$s_R(\mathcal{S}) = \sum_{\{\mathbf{s}_i,\mathbf{s}_j\}\subseteq\mathcal{S}\subset D} 1(\|\mathbf{s}_i - \mathbf{s}_j\| \leq R) \tag{4.2}$$

is the number of R-*close* pairs of points in $\mathcal{S}$. Given R, $N(\mathcal{S})$ and $s_R(\mathcal{S})$ are sufficient statistics for (β,γ). γ is an interaction parameter indicating the degree of repulsion. Large values of γ suggest weak repulsion while small values of γ indicate strong repulsion. $\gamma = 0$ provides the hardcore Strauss process which does not allow occurrence of any points within the interaction radius R. $\gamma = 1$ provides a homogeneous Poisson process.

Hence, the appropriate summary statistics would be $\mathbf{T} = (\log N(\mathcal{S}), K_R(\mathcal{S}))$. In practice, R is not known but we can choose a radius R though profile pseudo likelihoods. Alternatively, creating a set of R values yields a set of summary statistics, indexed by R.

In other repulsive point process settings, second order summary statistics would emerge as potential summary statistics because they illuminate clustering or inhibition behavior. For instance, with a stationary point process, the K function with radius d, $K(d)$, the *expected* number of the remaining points in the pattern within distance d from a typical point, is a useful measure. The empirical estimator of K_d given in Section 2.2 provides such a choice. The actual summary statistics would employ a set of d's.

The variance stabilized version could also be considered. This is the L function [29]; $\hat{L}_d(\mathcal{S}) = \sqrt{\hat{K}(d;\mathcal{S})/\pi}$ is often preferred because the fluctuations of the estimated K function increase with increasing d while the root transformation stabilizes these fluctuations [e.g., 101]. Again, a set of D's would be employed.

4.3.2. Explicit specification of an ABC algorithm

The ABC algorithm we present here is based on a semi-automatic approach proposed by [65]. They argue that the optimal choice of $\mathbf{T}(\mathcal{S}_{obs})$ is $E(\boldsymbol{\theta}|\mathcal{S}_{obs})$ and

then discuss how to construct $E(\boldsymbol{\theta}|\mathcal{S}_{obs})$. They consider a linear regression approach to construct the summary statistics through a pilot run. In our setting, we generate L sets of $\{\boldsymbol{\theta}_\ell, \mathcal{S}_\ell\}_{\ell=1}^L$. Then, we implement a linear regression for $E(\boldsymbol{\theta}_\ell|\mathcal{S}_{obs}) = \mathbf{a} + \mathbf{b}\boldsymbol{\eta}(\mathcal{S}_\ell, \mathcal{S}_{obs})$ where $\boldsymbol{\eta}(\mathcal{S}_\ell, \mathcal{S}_{obs})$ is a vector of functions of the summary statistics constructed from the simulated and observed point patterns[5]. Following above, we take $\boldsymbol{\eta}(\mathcal{S}, \mathcal{S}_{obs}) = (\eta_1(\mathcal{S}, \mathcal{S}_{obs}), \boldsymbol{\eta}_2(\mathcal{S}, \mathcal{S}_{obs}))$ where $\eta_1(\mathcal{S}, \mathcal{S}_{obs})$ and the $M \times 1$ vector $\boldsymbol{\eta}_2$ are

$$\begin{aligned} \eta_1(\mathcal{S}, \mathcal{S}_{obs}) &= \log n(\mathcal{S}) - \log n(\mathcal{S}_{obs}), \quad \text{and} \\ \eta_{2,d}(\mathcal{S}, \mathcal{S}_{obs}) &= \left| \sqrt{\hat{K}_d(\mathcal{S})} - \sqrt{\hat{K}_d(\mathcal{S}_{obs})} \right|^2. \end{aligned} \tag{4.3}$$

for $d = 1, 2, \ldots M$.

After obtaining $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$ by least squares, we can calculate $\hat{\boldsymbol{\theta}}^* = \hat{\mathbf{a}} + \hat{\mathbf{b}}\boldsymbol{\eta}(\mathcal{S}^*, \mathcal{S}_{obs})$ for any simulated $\mathcal{S}^*$. We set $\hat{\boldsymbol{\theta}}_{obs} = \hat{\mathbf{a}}$ and specify our distance function for the ABC through $\Psi(\hat{\boldsymbol{\theta}}^*, \hat{\boldsymbol{\theta}}_{obs})$ with Ψ specified below. To facilitate the regression, one can take a log transformation of the parameter vector, e.g., $\boldsymbol{\theta} = (\log\beta, \log\gamma)$ for the Strauss process.

Given the results of the pilot run, the approach proposed by [65] implements the ABC-MCMC algorithm by [132]. ABC-MCMC is a straightforward extension of the standard MCMC framework to ABC; convergence to the approximate posterior distribution, $\pi_\epsilon(\boldsymbol{\theta}|\mathbf{T}(\mathcal{S}_{obs}))$, is guaranteed. Specifically, with t denoting iterations and $q(\cdot|\cdot)$ denoting a proposal density,

1. Let $t = 1$.
2. Generate $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t-1)})$ and $\mathcal{S}^* \sim \pi(\mathcal{S}|\boldsymbol{\theta}^*)$ and calculate $\hat{\boldsymbol{\theta}}^* = \hat{\mathbf{a}} + \hat{\mathbf{b}}\boldsymbol{\eta}(\mathcal{S}^*, \mathcal{S}_{obs})$. Repeat this step until $\Psi(\hat{\boldsymbol{\theta}}^*, \hat{\boldsymbol{\theta}}_{obs}) < \epsilon$ where $\hat{\boldsymbol{\theta}}_{obs} = \hat{\mathbf{a}}$ and $\Psi(\hat{\boldsymbol{\theta}}^*, \hat{\boldsymbol{\theta}}_{obs})$ is defined below.
3. Calculate the acceptance ratio $\alpha = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)q(\boldsymbol{\theta}^{(t-1)}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta}^{(t-1)})q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(t-1)})}\right\}$. If $u < \alpha$ where $u \sim Unif(0,1)$, retain $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^*$, otherwise $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)}$. Return to step 2 and $t \to t+1$.

As a distance measure, we use the component-wise sum of quadratic loss for the log of the parameter vector, i.e., $\Psi(\hat{\boldsymbol{\theta}}_\ell, \hat{\boldsymbol{\theta}}_{obs}) = \sum_j (\hat{\theta}_{\ell,j} - \hat{\theta}_{obs,j})^2 / \hat{\text{var}}(\hat{\theta}_j)$ where $\hat{\text{var}}(\hat{\theta}_j)$ is the sample variance of j-th component of $\hat{\boldsymbol{\theta}}$. To choose an acceptance rate ϵ, through the pilot run, we obtain the empirical percentiles of $\{\Psi(\hat{\boldsymbol{\theta}}_\ell, \hat{\boldsymbol{\theta}}_{obs})\}_{\ell=1}^L$ and then select ϵ according to these percentiles. Step 2 is the most computationally demanding. We need to simulate the proposed point pattern $\mathcal{S}^* \sim \pi(\mathcal{S}|\boldsymbol{\theta}^*)$ until $\Psi(\hat{\boldsymbol{\theta}}^*, \hat{\boldsymbol{\theta}}_{obs}) < \epsilon$.

With regard to choice of number of summary statistics, when M is large more information is obtained but, if too large, overfitting, relative to the number of points in the point pattern, results. As a strategy for this selection, we specify M with equally spaced d's, and implement a lasso [198]. We determine the penalty parameter for the lasso by cross-validation and preserve the regression coefficients corresponding to the optimal penalty by using `glmnet` [71]. A simpler alternative is to fit using several choices of M and assess the sensitivity of posterior inference.

As noted in Section 2.2, in frequentist analysis, the minimum contrast estimator is often used to fit models using the K function. The minimum contrast estimator

[5] [65] implement linear regression for each component of $\boldsymbol{\theta}$. However, with a small number of parameters, we keep the notation as linear regression for multivariate responses.

requires the analytical form for the functional statistics which are not necessarily available for repulsive point processes. ABC does not require analytical expressions for the functional statistics because the approach compares the "estimated" K function for observed and simulated point patterns. However, if analytical forms for the functional summary statistics are available, the minimum contrast estimator or composite likelihood estimators [11, 92] would be available and easy to implement. Furthermore, software for these estimators has already been developed [12].

As a final comment here, [184] compared this proposed ABC-MCMC algorithm with the straightforward exchange algorithm of Murray et al. (2006), mentioned above, for a Strauss point process. They considered inefficiency factors (IF) for parameters, i.e., the ratio of the numerical variance of the estimate from the MCMC samples relative to that from hypothetical uncorrelated samples, using both model fitting approaches. They found that IFs for the exchange algorithm tend to be an order of magnitude greater than those from the ABC-MCMC algorithm. Also, the ABC-MCMC algorithm allows simple parallelization, which is not possible for the exchange algorithm. So, computationally, the ABC-MCMC algorithm can be much faster.

4.3.3. Neyman Scott process and shot noise processes

In a sense, ABC easier for the Neyman Scott and the shot noise processes than for Gibbs processes because simulation of samples is so straightforward. See [188] in this regard. That is, the Neyman Scott processes are defined in a generative fashion (Section 4.1) while the shot noise processes are generated in two straightforward stages. That is, first a realization of an HPP(λ) yields the shot noise intensity, followed by a realization of an NHPP given the shot noise intensity (Section 4.1) Shot noise process realizations can also be developed by additive superposition, i.e., summing up realizations over each of the random kernels. We note that [188] illustrate the use of ABC for a Thomas process.

The question to ask here is, in general, what should be the choice of summary statistics for spatial point patterns? Again, usual measures are the K function or the pair correlation function. For ABC we only require the empirical K function or the empirical partial correlation function. $\hat{K}(d)$ was given in Section 2.2. The empirical correlation function $\hat{g}(d)$ is derived from the relationship, $g(d) = K'(d)/2\pi d$ using either a smoothing spline for $K'(d)$ or else a direct kernel estimate for $g(d)$. Whichever is selected, the infinity of statistics is reduced to a single measure for ABC comparison by a weighted integration, $\int_{d \le d_o} w(d)(\hat{K}(d) - \hat{K}_{obs}(d))^2 dd$ for a suitable maximum distance, d_o. See [188] for further details.