`excursions`: An R package for estimation of excursion and contour sets for latent Gaussian models.

David Bolin

October 15, 2013

Abstract The R package `excursions` contains methods for calculating probabilistic excursion sets, contour credible regions, and simultaneous confidence bands for latent Gaussian stochastic processes and fields. This document contains an introduction to the methods used in the package and some worked out examples with simulated data that shows how the package can be used.

1 Introduction

The ability to find regions where a stochastic process exceeds a certain level or is significantly different from some reference level is important in several areas of application. Examples in geosciences include studies of air pollution, where one is interested in testing if, and where, the pollution level exceeds some given limit value set by some regulatory agency (Cameletti et al., 2013), temperature (Furrer et al., 2007), precipitation (Sain et al., 2011), and vegetation (Eklundh and Olsson, 2003, Bolin et al., 2009), and similar problems can be found in a wide range of scientific fields including brain imaging (Marchini and Presanis, 2003) and astrophysics (Beaky et al., 1992).

The problem of finding regions of this kind is closely related to the problem of multiple hypothesis testing, and there exists a large literature on different types of correction methods that are used to handle the so-called mass significance problem that occurs when multiple tests are done simultaneously (see e.g. Marchini and Presanis, 2003, for an overview). Recently, Bolin and Lindgren (2013) approached this problem not as a testing problem but by providing definitions for the regions of interest directly in terms of distributional properties. Bolin and Lindgren (2013) also provided methods for estimating these sets in practice for the class of latent Gaussian models, which is a large model class that is widely used in applications (see e.g. Rue et al., 2009).

This work summarizes the methods of Bolin and Lindgren (2013) and contains an introduction to the R (R Core Team, 2013) package `excursions`, which is an R implementation of the methods described in Bolin and Lindgren (2013). The package is available on CRAN (`http://CRAN.R-project.org/package=excursions`) and can be installed in R by typing

```
install.packages("excursions")
```

The structure of the paper is as follows. Section 2 contains an introduction to the definitions and methods that are needed. Section 3 contains an introduction to the methods in `excursions`. Section 4 contains a step-by-step tutorial on how to use the methods in practice for a simple latent Gaussian model. Finally, Section 5 contains some more realistic examples of when the package can be used.

2 Methods

Let $X(s)$ be a stochastic process, or random field, defined on some domain of interest, Ω, which we assume is open with a well-defined area $|\Omega| < \infty$. Let u be some level of interest and α be an error probability. There are four related regions that are of interest.

The positive level u excursion set with probability α, $E^+_{u,\alpha}(X)$, is defined as the largest set so that with probability $1-\alpha$ the level u is exceeded at all locations in the set. Similarly, the negative u excursion set with probability α, $E^-_{u,\alpha}(X)$, is defined as the largest set so that with probability $1-\alpha$ the process is below the level u at all locations in the set. A related set is the level u contour credibility region, $E^c_{u,\alpha}(X)$, which is defined as the smallest set such that with probability $1-\alpha$ *all* level u crossings of X are in the set. Finally, the level u contour avoiding set, $E_{u,\alpha}(X)$, is defined as the complement of $E^c_{u,\alpha}(X)$.

As a tool for visualizing and calculating the sets above in practice, Bolin and Lindgren (2013) introduced the excursion functions. These are compact ways of showing the excursion sets simultaneously for all values of *alpha*. For a level u, the positive and negative excursion functions, contour avoidance function, and the contour function are defined as

$$F^+_u(\mathbf{s}) = 1 - \inf\{\alpha; \mathbf{s} \in E^+_{u,\alpha}\}, \qquad F^-_u(\mathbf{s}) = 1 - \inf\{\alpha; \mathbf{s} \in E^-_{u,\alpha}\},$$
$$F_u(\mathbf{s}) = 1 - \inf\{\alpha; \mathbf{s} \in E_{u,\alpha}\}, \qquad F^c_u(\mathbf{s}) = \sup\{\alpha; \mathbf{s} \in E^c_{u,\alpha}\}.$$

The functions take values between zero and one and each set $E^\bullet_{u,\alpha}$ can be retrieved as the $1-\alpha$ excursion set of the function $F^\bullet_u(s)$.

We now summarize the method that is used to calculate these sets for latent Gaussian models. Let $\mathbf{x}$ be the latent Gaussian process evaluated at the locations of interest, $\mathbf{y}$ be a vector of observations of the field, and $\boldsymbol{\theta}$ be the vector of model parameters. The distribution that should be used when calculating the excursion set is then the posterior distribution for $\mathbf{x}$,

$$\pi(\mathbf{x} \mid \mathbf{y}) = \int \pi(\mathbf{x} \mid \mathbf{y}, \boldsymbol{\theta}) \pi(\boldsymbol{\theta} \mid \mathbf{y}) d\boldsymbol{\theta}. \tag{1}$$

Bolin and Lindgren (2013) proposed a method based on a sequential integration method in combination with a parametric family for the possible sets. This method is used for calculating the sets for Gaussian distributions, and is then combined with one of five different strategies for handling the latent Gaussian structure. Thus, we first describe the method for handling the Gaussian case and then turn to the full latent Gaussian structure. We only describe the method for calculating a positive excursion, and given this method, the extension to calculating the other sets is trivial.

Bolin and Lindgren (2013) discusses several different parametric families for the excursion sets. The simplest choice, which is the one that is used in the `excursions` package is given by $D(\rho) = \{\mathbf{s}; \mathsf{P}(x(\mathbf{s}) > \gamma) \geq 1-\rho\}$. This parametric family is based only on the marginal distributions, which are easy to calculate.

The other component that is needed is a sequential integration method for calculating gaussian probabilities $P(D \subseteq A^+_\gamma(\mathbf{x}))$, which can be written as

$$\frac{|\mathbf{Q}|^{1/2}}{(2\pi)^{d/2}} \int_{\mathbf{a} \leq \mathbf{x}} \exp\left(-\frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x}\right) \mathrm{d}\mathbf{x}, \tag{2}$$

where $\mathbf{a}$ is a vector depending on the mean value of $\mathbf{x}$, the domain D, and threshold γ. To take advantage of the possible sparsity of $\mathbf{Q}$ if a Markovian model is used, the integral is rewritten as

$$\int_{a_1}^\infty \pi(x_1|\mathbf{x}_{2:d}) \int_{a_2}^\infty \pi(x_2|\mathbf{x}_{3:d}) \cdots \int_{a_{d-1}}^\infty \pi(x_{d-1}|x_d) \int_{a_d}^\infty \pi(x_d)\, \mathrm{d}\mathbf{x} \tag{3}$$

where, if the problem has a Markov structure, $x_i|\mathbf{x}_{i+1:d}$ only depends on a few of the elements in $x_{i+1:d}$ given by the Markov structure. The integral is calculated using a sequential importance sampler by starting with the integral of the last component $\pi(f_d)$ and then moving backward in the indices, see Bolin and Lindgren (2013) for further details.

Given the parametric family and the sequential integration method, the excursion set for a Gaussian process is found using the following method.

1. Find the reordering given by the order the nodes are added to $D(\rho)$ when ρ is increased.
2. Sequentially add nodes to the set D according to the ordering given above and in each step update the conditional probability $P(D \subseteq A_\gamma^+(\mathbf{x}) \mid \mathbf{y}, \boldsymbol{\theta})$. Stop as soon as this probability falls below $1-\alpha$.
3. $E_{\gamma,\alpha}^+$ is given by the last set D for which $P(D \subseteq A_\gamma^+(\mathbf{x}) \mid \mathbf{y}, \boldsymbol{\theta}) \geq 1-\alpha$.

Note that if we set $\alpha = 1$ and save the conditional probability $P(D \subseteq A_\gamma^+(\mathbf{x}) \mid \mathbf{y}, \boldsymbol{\theta})$ in each iteration, we obtain an estimate of the excursion function $F_\gamma^+(\mathbf{s})$.

We now have the method for handling the Gaussian case, which is the basis for the five methods for handling the latent Gaussian case described below.

`EB` **(Empirical Bayes)** The `EB` method is the simplest method for handling the latent Gaussian structure. One then calculates the excursion set under the conditional posterior $\pi(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta}_0)$ where $\boldsymbol{\theta}_0$ for example is the maximum a posteriori estimate or the maximum likelihood estimate of $\boldsymbol{\theta}$. $\pi(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta}_0)$ is Gaussian if the likelihood of the problem is Gaussian, and in other situations $\pi(\mathbf{x}|\mathbf{y})$ is approximated with a Gaussian distribution $\tilde{\pi}_G(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta}_0)$ that is used in the calculation of the excursion set. The Gaussian approximation is achieved by matching the modal configuration and the curvature at the mode, as described in Rue et al. (2009).

`QC` **(Quantile Correction)** The `QC` method is based on using the `EB` method with a modification based on the marginal posteriors. For each i, $\mathsf{P}(x_i > a_i|\mathbf{y})$ is calculated and the limits a_i in (3) are replaced with $\tilde{a}_i = \sigma_i \Phi^{-1}(1 - \mathsf{P}(x_i > a_i|\mathbf{y}))$, where σ_i is the marginal standard deviation for $x_i|\mathbf{y}, \boldsymbol{\theta}_0$ and Φ denotes the standard Gaussian distribution function. One then has $\mathsf{P}(x_i > \tilde{a}_i|\mathbf{y}, \boldsymbol{\theta}_0) = \mathsf{P}(x_i > a_i|\mathbf{y})$, which means that the QC method is exact if the x_i's are independent. For non-Gaussian likelihoods, a Gaussian approximation $\tilde{\pi}_G(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta}_0)$ is again used as in the `EB` method.

`NI` **(Numerical Integration)** In the `NI` method, one numerically approximates the excursion function as $F_\gamma^+(\mathbf{s}) = \sum_{k=1}^K \lambda_k F_k(\mathbf{s})$ where $F_k(\mathbf{s})$ is the excursion function calculated for the conditional posterior $\pi(\mathbf{x} \mid \mathbf{y}, \boldsymbol{\theta}_k)$ (or $\tilde{\pi}_G(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta}_0)$ for non-Gaussian likelihoods) for a fixed parameter configuration $\boldsymbol{\theta}_k$. The configurations $\boldsymbol{\theta}_k$ are chosen as in the INLA method and the weights λ_k are chosen proportional to $\pi(\boldsymbol{\theta}_k \mid \mathbf{y})$. Finally, the excursion set for a fixed α is retrieved as the excursion set $A_\alpha^+(F_\gamma^+)$ of the excursion function.

`NIQC` **(Numerical integration with Quantile Corrections)** The `NIQC` and the `iNIQC` (improved NIQC) methods are based on combining the QC method and the NI method. The NI method is used for calculating the excursion function, but for each parameter configuration, the QC method is used to modify the integration limits while calculating $F_k(\mathbf{s})$. These methods are thus, only required for models with non-Gaussian likelihoods, where Gaussian approximations $\tilde{\pi}_G(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta}_0)$ are used. For each configuration $\boldsymbol{\theta}_k$, the modified limits $\tilde{a}_i$ are calculated so that $\mathsf{P}(z_i > \tilde{a}_i|\mathbf{y}, \boldsymbol{\theta}_0) = \mathsf{P}(x_i > a_i|\mathbf{y})$.

`iNIQC` **(improved NIQC)** The `iNIQC` method is equivalent to the `NIQC` method except that the modified limits $\tilde{a}_i$ are calculated so that $\mathsf{P}(z_i > \tilde{a}_i|\mathbf{y}, \boldsymbol{\theta}_0) = \mathsf{P}(x_i > a_i|\mathbf{y}, \theta_i)$. The `iNIQC` method is more computationally demanding since one has to calculate the marginal quantiles for each configuration but should also perform better in practice for models with non-Gaussian likelihoods.

3 Implementation

The R (R Core Team, 2013) package `excursions` is an R interface to C functions that estimates excursion and contour sets using the methods described above. The C functions use methods from a number of C and Fortran libraries, such as `BLAS` (Dongarra et al., 1990), `LAPACK` (Anderson et al., 1999), and `CHOLMOD` (Chen et al., 2008) for efficient matrix manipulations together with function from the GNU Scientific library (Galassi and Gough, 2006) and several different reordering methods. Notably, the `CAMD` library (Amestoy et al., 1996, 2004) is used for constrained approximate minimum degree orderings.

The R interface contains two main function for calculating excursion and contour sets, `excursions` and `excursions.inla`. `excursions` is the main function that calls the C functions that preforms the calculations and `excursions.inla` is an interface function that can be used for latent Gaussian models that have been estimated using the `INLA` package (Rue et al., 2009).

As for the different strategies for handling latent Gaussian structures, `excursions` has direct support for `EB` and `QC` while `excursions.inla` has support for all strategies described in Section 2. In general, `EB` and `QC` are suitable for models where the posterior distribution is close to a Gaussian distribution. The methods are equivalent if the posterior is Gaussian and in other cases `QC` is more accurate than `EB`. The `NI` method is, in general, more accurate than the `QC` method, and for non-Gaussian likelihoods, the `NIQC` and `iNIQC` methods can be used for improved results. In general the accuracy and computational cost of the methods are as follows

$$\begin{aligned} \text{accuracy: } & \texttt{EB} < \texttt{QC} < \texttt{NI} < \texttt{NIQC} < \texttt{iNIQC} \\ \text{comp. cost: } & \texttt{EB} \approx \texttt{QC} < \texttt{NI} \approx \texttt{NIQC} < \texttt{iNIQC}. \end{aligned}$$

If the main purpose of the analysis is to construct excursion or contour sets for low values of α, we recommend using the `QC` method for problems with Gaussian likelihoods and the `NIQC` method for problems with non-Gaussian likelihoods. The increase in accuracy of the improved NIQC method is often small compared to the added computational cost.

The package also contains a function `excursions.simconf` for calculating simultaneous confidence bands for Gaussian processes and a function `excursions.int` for calculating Gaussian integrals.

3.1 `excursions`

A basic call to the function `excursions` looks like

```
excursions(alpha,u,mu,Q,type)
```

Here, `alpha` is the error probability and `u` is the excursion or contour level. `mu` and `Q` are the mean vector and precision matrix for the joint distribution. `type` determines what type of region that is considered: '>' for positive excursion regions, '<' for negative excursion regions, '!=' for contour avoiding regions, and '=' for contour credibility regions.

The default strategy that is used to handle the possible latent Gaussian structure is the `EB` method, and the QC method can be used instead by using the call

```
excursions(alpha,u,mu,Q,type,method='QC',rho)
```

If QC is used, the argument `rho` must be provided, which should contain a vector with point-wise marginal probabilities: $P(X > u)$ for positive excursions and contour regions, and $P(X < u)$ for negative excursions. In the situation when $\pi(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta})$ is Gaussian but $\pi(\mathbf{x}|\mathbf{y})$ is not, the marginal probabilities should be calculated under the distribution $\pi(\mathbf{x}|\mathbf{y})$ and `mu` and `Q` should be chosen as the mean and precision for the distribution $\pi(\mathbf{x}|\mathbf{y}, \hat{\boldsymbol{\theta}})$ where $\hat{\boldsymbol{\theta}}$ is the MAP or ML estimate of the parameters.

There are a number of other optional arguments that one can want to provide in certain situations. `n.iter` specifies the number or iterations in the MC sampler that is used for approximating probabilities. The default value is 10000. `Q.chol` is used to provide the Cholesky factor of the precision matrix. Note that it is not necessary to provide this unless you are using a fixed reordering, provided using the `reo` argument, since the Cholesky factor has to be recomputed for the reordered nodes. `vars` is used to provide the marginal variances, i.e. the diagonal elements of $\mathtt{Q}^{-1}$. `reo` is used to provide a reordering that defines the parametric family used when calculating the probabilities. If `reo` is provided, all other arguments are assumed to be reordered using this reordering. This argument is mainly intended for internal usage and should not be needed for basic usage of the function. `ind` is used to provide indices of the nodes that should be analyzed, this argument is useful if only a part of the full distribution should be considered. `max.size` is used to proved an upper bound for the number of nodes that should be included in the set of interest. This argument is mainly intended for internal usage and should not be needed for basic usage of the function. `max.threads` decides the number of computational threads the program can use, the default argument is 0 which makes the program use the maximum number of threads allowed by the system. Setting argument `keep` to `TRUE` makes the program save the binary I/O files used for passing information between R and C. Finally, calling the function with the argument `verbose=TRUE` sets the function to verbose mode.

The function returns a list containing the following quantities: the excursion set, contour credible region, or contour avoiding set, `D`, the corresponding excursion function calculated for values up to alpha, `F`, the estimated standard deviations of F estimate, `Fe`, the marginal excursion probabilities, `rho`, the reordering used for excursion function, `reo`, the inverse reordering, `ireo`, and the marginal variances, `vars`.

3.2 excursions.inla

A basic call to the function `excursions.inla` looks like

excursions.inla(result.inla,alpha,u,type,ind,method)

Here, `result.inla` is the result object from the INLA call. The arguments `alpha`, `u`, and `type` are the same as for `excursions`.

The argument `ind` is also the same as for `excursions`, i.e. it is used to specify which part of the joint distribution that one is interested in. The difference is that `ind` was optional in `excursions` but is required in `excursions.inla`. The reason for this is that the distribution calculated by INLA often is a joint distribution of the linear predictor, possible covariates, the measurements, and other things, and the user therefore needs to specify which parts of the distribution that are relevant to the analysis.

The argument `method` specifies which strategy that should be used to handle the latent Gaussian structure, this argument could either be `'EB'` or `'QC'` for `excursions` but `excursions.inla` can handle all possible strategies: `'EB'`, `'QC'`, `'NI'`, `'NIQC'`, and `'iNIQC'`.

The function has two optional arguments: `n.iter` specifies the number or iterations in the MC sampler that is used for approximating probabilities, with the default value 10000, and `verbose` can be set to `TRUE` for verbose mode.

The function returns a list containing the following objects: The excursion function calculated for all values up to `alpha`, `F`, the marginal excursion or contour probabilities, `rho`, and the posterior mean of the distribution, `mean`.

3.3 `excursions.simconf`

The function `excursions.simconf` is used for calculating simultaneous confidence bands for Gaussian processes $X(s)$. The function returns upper and lower bounds $a(s)$ and $b(s)$ such that $\mathsf{P}(a(s) < X(s) < b(s)) = 1 - \alpha$. Currently, the method only supports the EB method and uses the following parametric family for the bounds:

A basic call to the function looks like

```
excursions.simconf(alpha, mu, Q)
```

where `alpha` is the coverage probability, `mu` is the mean value vector for the process, and `Q` is the precision matrix for the process.

The function has a few optional arguments similarly to those of `excursions`. `n.iter` specifies the number or iterations in the MC sampler that is used for approximating probabilities. The default value is 10000. `Q.chol` is used to provide the Cholesky factor of the precision matrix. `vars` is used to provide the marginal variances, i.e. the diagonal elements of $\mathtt{Q}^{-1}$. `ind` is used to provide indices of the nodes that should be analyzed, this argument is useful if only a part of the full distribution should be considered. `max.threads` decides the number of computational threads the program can use, the default argument is 0 which makes the program use the maximum number of threads allowed by the system. Finally, calling the function with the argument `verbose=TRUE` sets the function to verbose mode.

The function returns a list with the lower limit `a` and the upper limit `b`.

3.4 `excursions.int`

The function `excursions.int` is a utility function that is used for calculating Gaussian integrals

$$\int_{\mathbf{a}}^{\mathbf{b}} \frac{|\mathbf{Q}|^{1/2}}{(2\pi)^{n/2}} \exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \mathbf{Q}(\mathbf{x}-\boldsymbol{\mu}))\,\mathrm{d}\mathbf{x} \tag{4}$$

using the sequential importance sampling method described in Section 2. An important feature of the function is that the integration can be stopped as soon as the value of the integral in the sequential integration goes below some given value $1-\alpha$. Using this option saves computation time if one only is interested in the exact value of the integral given that it is larger than $1-\alpha$.

A basic call to the function looks like

```
excursions.simconf(mu, Q,a,b)
```

where `mu` is the mean value vector, `Q` is the precision matrix, `a` is the lower limit, and `b` is the upper limit. The function has optional arguments `n.iter`, `Q.chol`, `ind`, `verbose`, and `max.threads` with the same meaning as those in `excursions.simconf`. Finally, the argument `alpha` can be used to define α in the limit $1-\alpha$ for the integral.

The function returns a list with two elements, `P` contains the estimated value of the integral and `Pe` contains an estimate of the uncertainty of `P`.

4 A tutorial using a simple latent Gaussian model

In this section, the different ways the package can be used are illustrated using a simple one-dimensional problem. The usage of the functions are only dependent on that we have some joint

distribution we want to calculate excursion sets for, and not how this distribution is obtained. Thus, the added difficulty when looking at more complicated problems will be in the specification of the joint distribution and not in how the excursions functions are used. A few more complicated spatial problems are shown in Section 5.

The model we use is a partially observed AR(1) process with a covariate for the mean and Gaussian measurement noise. We assume the measurement equation

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \boldsymbol{\varepsilon} \tag{5}$$

where $\mathbf{A}$ is a matrix that determines which nodes of the latent field $\mathbf{x}$ that are observed and $\boldsymbol{\varepsilon} \sim \mathsf{N}(\mathbf{0}, \tau_e^{-1}\mathbf{I})$ is Gaussian measurement noise with a precision parameter τ_e. The latent process is defined as

$$\mathbf{x} = \mathbf{b}\boldsymbol{\beta} + \mathbf{z} \tag{6}$$

where $\mathbf{b}$ is a covariate for the mean and $\mathbf{z}$ is an AR(1) process defined as

$$\begin{aligned} z_1 &\sim \mathsf{N}(0, (\tau(1-\rho^2)^{-1}) \\ z_i &= \rho z_{i-1} + \varepsilon_i, \quad i = 1, \ldots, n, \end{aligned}$$

where $\varepsilon_i \sim \mathsf{N}(0, \tau^{-1})$ and $|\rho| < 1$.

4.1 Generating the data

We start by loading the library and define the parameters for the model.

```
> library(excursions)
> rho = 0.9
> tau = 20
> tau.e = 1
```

Next, we calculate the precision matrix for the joint distribution of the AR process, define a piecewise linear covariate, `mu`, for the mean, and sample the process:

```
> n = 500
> x = 1:n
> mu = 10*((x<n/2)*(x-n/2) + (x>=n/2)*(n/2-x)+n/4)/n
> Q = tau*sparseMatrix(i=c(1:n, 2:n), j=c(1:n, 1:(n-1)),
+                      x=c(1,rep(1+rho^2, n-2),1, rep(-rho, n-1)),
+                      dims=c(n, n), symmetric=TRUE)
> X = mu+solve(chol(Q), rnorm(n))
```

We measure the sampled process at `n.obs` random locations under Gaussian measurement noise. The measurement locations are drawn at random from the set of nodes where the process is defined.

```
> n.obs = 200
> obs.loc = sample(1:n,n.obs)
> A = sparseMatrix(i=1:n.obs, j=obs.loc, x=rep(1, n.obs), dims=c(n.obs, n))
> Y = as.vector(A %*% X + rnorm(n.obs)/sqrt(tau.e))
```

We now want to reconstruct the latent process `x` given the measurements `Y`. Assuming known model parameters, the posterior distribution for the latent AR process given the measurements is Gaussian with mean `mu.post` and a precision matrix `Q.post`

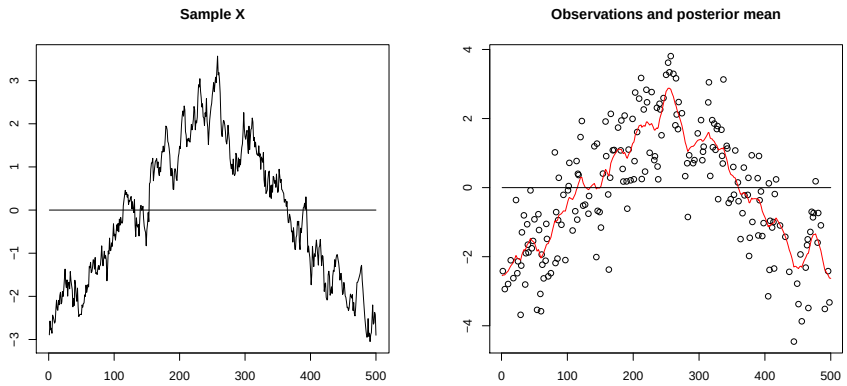

Figure 1: The process `X`, the observations `Y`, and the posterior mean.

```
> Q.post = (Q + (t(A)%*%A)*tau.e)
> mu.post = as.vector(mu + solve(Q.post,(t(A)%*%(Y-A%*%mu))*tau.e))
```

Figure 1 shows the sample `X`, the measurements `Y`, and the posterior mean `mu.post`.

```
> plot(X,type="l",main='Sample X',xlab="",ylab="")
> lines(rep(0,n))

> plot(obs.loc,Y,main='Observations and posterior mean',xlab="",ylab="")
> lines(mu.post, col=2)
> lines(rep(0,n))
```

4.2 Estimation for known model parameters

We now calculate the positive excursion function for the level 0.

```
> res.e = excursions(alpha=0.99, u=0, mu=mu.post, Q=Q.post, type='>')
```

We should have used `alpha=1` to calculate the function for all nodes. The reason for instead using `alpha=0.99` is that we know that many nodes will have probabilities close to zero, and by using `alpha=0.99`, the function is only calculated for the values up to 0.99. This way, the function is not calculated for any of the nodes that are not included in the 0.99 excursion set but instead set to zero for these nodes. Thus, the values for all nodes where the function is between 0 and 0.01 are truncated to 0. This saves computation time and since one might not be that interested in the exact form of the 0.99 excursion set, the error in the function is negligible.

The level 0 excursion set with probability `alpha` can now be obtained as `res.e$F>1-alpha` for any value of `alpha` (up to the value we used in the call to `excursions`). Alternatively, if one only is interested in the excursion set for a given value of `alpha` one can use that value directly in the `excusions` call:

```
> E1 = res.e$F>1-0.05
> res2 = excursions(alpha=0.05, u=0, mu=mu.post, Q=Q.post, type='>')
> E2 = res2$D
> sum(abs(E1 - E2))

[1] 0
```

The sets `E1` and `E2` should be equal in the code above, but can differ in a few nodes because of the Monte Carlo error in the probability calculations. If one is worried that this error is too large, the number of Monte Carlo samples can be increased by using the `n.iter` argument:

```
> res2 = excursions(alpha=0.05, u=0, mu=mu.post, Q=Q.post, type='>',n.iter=20000)
```

Only calculating the `alpha=0.05` excursion set is much faster than calculating the entire excursion function; however, the gain is lost as soon as an excursion set for any higher value of `alpha` is needed since one then would have to recalculate the excursion function for all values up to that value.

We can also calculate the level 0 contour credibility function by changing `type='>'` to `type='='`:

```
> res.c = excursions(alpha=0.99, u=0, mu=mu.post, Q=Q.post, type='=')
```

Both the excursion function and the contour credibility function are shown in Figure 2, together with some corresponding excursion and contour sets.

```
> plot(res.e$rho, type="l",xlab="",ylab="")
> lines(res.e$F, col=2)

> plot(res.c$F, type="l",col=2,xlab="",ylab="")

> plot(rep(0,n),type="l",main='alpha=0.05 Excursion sets',xlab="",ylab="")
> barplot(res.e$rho>0.95,space=0,border=FALSE,add=T,offset=-0.5)
> barplot(res.e$F>0.95,space=0,border=FALSE,col=2,add=T,offset=-0.5)

> plot(rep(0,n),type="l",main='alpha=0.05 contour set',xlab="",ylab="")
> barplot(res.c$F>0.95,space=0,border=FALSE,col=2,add=T,offset=-0.5)
```

Note that the sets shown in the lower panels of the figure are obtained as excursion sets of the functions in the upper panels. Also note that the excursion function to the left always is less than or equal to the marginal probabilities, and the corresponding α excursion set (the red set in the lower left panel) is therefore always contained in the set where the marginal probabilities is at least $1 - \alpha$ (the grey set in the lower left panel).

We can also calculate a simultaneous confidence band for the process as follows

```
> conf = excursions.simconf(alpha = 0.1, mu=mu.post,Q=Q.post)
```

The resulting confidence band is shown in the left panel of Figure 3.

```
> plot(obs.loc,Y,xlab="",ylab="")
> lines(X)
> lines(mu.post,col=2)
> lines(conf$a,col=4)
> lines(conf$b,col=4)
```

4.3 Accounting for parameter uncertainty

Above, we assumed that the parameters were known when making the predictions and calculating the excursion sets. In practice, the parameters are rarely known and instead have to be estimated from the data before making the predictions. Accounting for the parameter uncertainty affects the excursion sets, and in this section, we show how the excursion sets can be calculated while accounting for the uncertainty.

First, we need to estimate the parameters. This can be done using the INLA method as follows

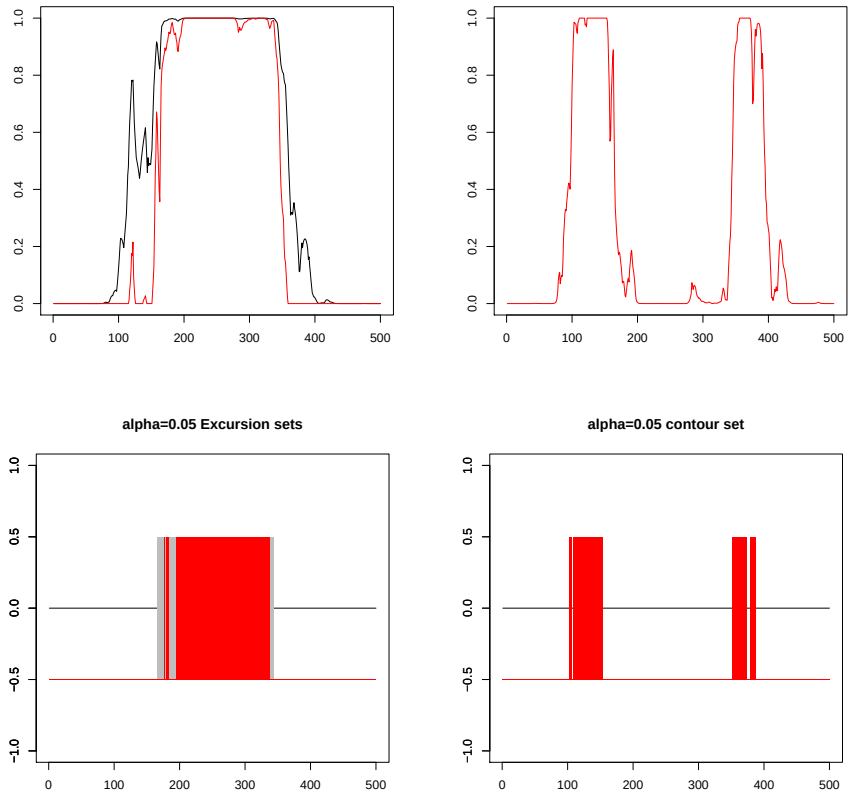

Figure 2: In the top left panel, the excursion function (red) and the marginal probabilities (black) are shown, the contour function is shown in the top right panel). The corresponding excursion and contour sets are shown in the lower panels.

```
> library(INLA)
> ef = list(c(list(ar=x),list(cov=mu)))
> s.obs = inla.stack(data=list(y=Y), A=list(A), effects=ef, tag="obs")
> s.pre = inla.stack(data=list(y=NA), A=list(1), effects=ef,tag="pred")
> stack = inla.stack(s.obs,s.pre)
> formula = y ~ -1 + cov + f(ar,model="ar1")
> result = inla(formula=formula, family="normal",
+                 data = inla.stack.data(stack),
+                 control.predictor=list(A=inla.stack.A(stack),
+                         compute=TRUE,cdf=c(0)),
+                 control.compute = list(config = TRUE))
> iprd <- inla.stack.index(stack, "pred")$data
```

It is outside the scope of this work to go into details on the INLA method and the different options in the R-INLA framework, we instead refer to `www.r-inla.org` for tutorials and examples. However, one detail in the INLA call is important for the excursion method and this is the the argument `control.compute = list(config = TRUE)`. This argument is not needed for the actual estimation, but we include this argument in order to save some quantities to the output that we will need later. We can plot the resulting linear predictor to see how different it is from

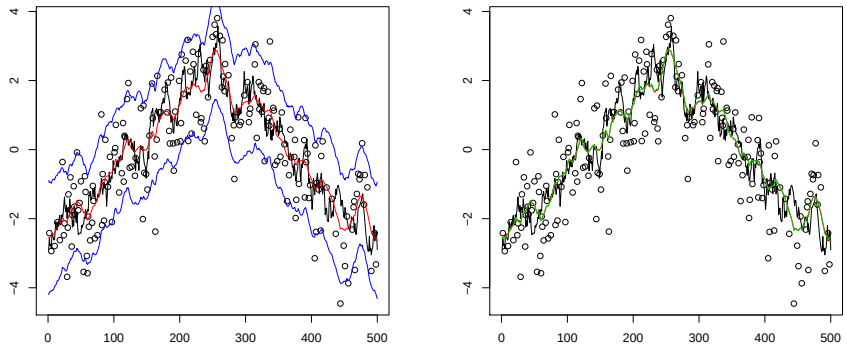

Figure 3: The observations (circles), the true latent process (black), and the posterior mean assuming known parameters (red). The left panel shows the 0.9 simultaneous confidence band assuming known parameters (blue) and the right panel shows the posterior mean calculated with INLA (green).

the posterior mean we previously used, see Figure 3.

```
> plot(obs.loc,Y,xlab="",ylab="")
> lines(X)
> lines(mu.post,col=2)
> lines(result$summary.linear.predictor$mean[iprd],col=3)
```

We can now calculate the excursion function accounting for the parameter uncertainty. Because we added `cdf=c(0)` to the INLA call, we can extract an estimate of the marginal probabilities for exceeding the level 0 as follows

```
> p.marginal = 1- result$summary.linear.predictor$"0 cdf"
```

If `cdf=c(0)` was not used in the INLA call, or if one later decides that excursions for some other level are needed, the marginal probabilities for any level `u` can be estimated as

```
> p.m <- sapply(iprd, function(i) 1- inla.pmarginal(u,
+                 result$marginals.fitted.values[[i]]))
```

We need these marginal probabilities to use the QC method to calculate excursion functions while accounting for the parameter uncertainty. We also need the mean and precision for the linear predictor given the parameter estimates, these can be extracted from the INLA output as

```
> map.i = which(sapply(1:result$misc$configs$nconfig,function(i)
+                          result$misc$configs$config[[i]]$log.posterior) %in% 0)
> mu.pred=result$misc$configs$config[[map.i]]$mean
> Q.pred=result$misc$configs$config[[map.i]]$Q
>
```

What this statement does is to find the MAP parameter configuration, which has the scaled log-posterior value equal to zero, and extract the corresponding mean and precision. `mu.pred` and the other quantities are evaluated for the joint distribution of the observations and the predictors. We are only interested in the predictor part of the distribution while calculating the excursion set, and we therefore use the `ind` argument to indicate the part of the joint distribution that we are interested in. Calculating the excursion set is done as follows,

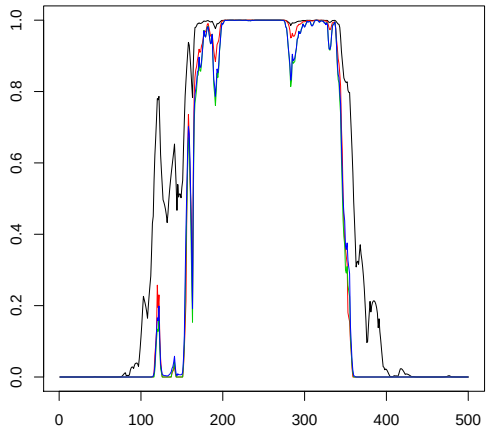

Figure 4: The marginal probabilities (black), the EB excursion function (red), the QC excursion function (green), and the NI excursion function (blue).

```
> res.eb = excursions(alpha=0.99, u=0, mu=mu.pred, Q=Q.pred, ind=iprd,type='>')
```

This gives us the excursion function calculated using the EB method, to use the QC method instead, we add the marginal probabilities and change the method:

```
> res.qc = excursions(alpha=0.99, u=0, mu=mu.pred, Q=Q.pred, method='QC',
+                                          rho = p.marginal, ind=iprd,type='>')
```

Having to manually extract the mean, precision, and marginal probabilities from the INLA output is somewhat cumbersome and prone to errors. A better alternative is therefore to use the function `excursions.inla` which takes the INLA output directly as an argument and extracts all relevant parts automatically. All what we must know is what nodes of the joint distribution that we are interested in and supply these using the `ind` argument.

```
> res.qc2 = excursions.inla(result,ind=iprd,alpha=0.99,u=0,method='QC',type='>')
```

Using `excursions.inla` we can also test the more advanced methods for handling the latent Gaussian structure by changing the `method` argument. In this simple example, the conditional posterior is Gaussian so we do not need the NIQC method or the improved NIQC method; however, we can test the NI method

```
> res.ni = excursions.inla(result,ind=iprd,alpha=0.99,u=0,method='NI',type='>')
```

We plot the different excursion functions as follows, see Figure 4.

```
> plot(p.marginal[iprd],type="l",xlab="",ylab="")
> lines(res.eb$F[iprd],col=2)
> lines(res.qc$F[iprd],col=3)
> lines(res.ni$F,col=4)
```

Calculating the NI excursion function is much more computationally demanding than calculating the QC excursion function, and since the two functions are quite similar in this simple example, there is likely no need for the increased accuracy of the NI excursion function.

5 More complicated case studies

In this section, we present a few examples when the package is used on more complicated models. As previously mentioned, the added difficulty when looking at more complicated problems will be in the specification of the joint distribution and not in how the excursions functions are used. The actual call to the functions in this package are equivalent to those in section 4.

5.1 Latent Gaussian Matérn fields

In this example, we assume the following latent Gaussian model

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \boldsymbol{\varepsilon} \tag{7}$$

where $\boldsymbol{\varepsilon}$ is mean-zero Gaussian measurement noise with some variance σ_e^2 and $\mathbf{X}$ is a Gaussian Matérn field. The matrix $\mathbf{A}$ links the measurement locations to the nodes of the field. We start by loading the required libraries, the fields library is only used for making the figures.

```
> library(excursions)
> library(INLA)
> library(fields)
```

Next, we define the range parameter κ^2, the measurement noise variance σ_e^2, and create a mesh for the SPDE representation of the Matern field

```
> kappa=sqrt(8)/2
> sigma.e=0.1
> x=seq(from=0,to=10,length.out=40)
> lattice=inla.mesh.lattice(x=x,y=x)
> mesh=inla.mesh.create(lattice=lattice, extend=FALSE, refine=FALSE)
```

We now draw 1000 observation locations at random and construct the corresponding $\mathbf{A}$ matrix, we then create the SPDE object and sample the model

```
> n.obs=1000
> obs.loc = cbind(runif(n.obs)*diff(range(x))+min(x), runif(n.obs)*diff(range(x))+min(x))
> A=inla.spde.make.A(mesh, loc=obs.loc)
> spde = inla.spde2.matern(mesh, alpha=2)
> sample = inla.qsample(n=1, Q=inla.spde.precision(spde, theta=c(0,log(kappa))))
> obs = as.vector(A %*% sample + rnorm(n.obs)*sigma.e)
```

Given the observations, we estimate the model parameters using INLA

```
> ef = list(list(field=1:mesh$n))
> s.obs = inla.stack(data=list(y=obs), A=list(A), effects=ef,tag='obs')
> s.pred= inla.stack(data=list(y=NA), A=list(1), effects=ef,tag='pred')
> stack = inla.stack(s.obs,s.pred)
> iprd <- inla.stack.index(stack, 'pred')$data
> formula = y ~ -1 + f(field, model=spde)
> result = inla(formula, data=inla.stack.data(stack),
+                           family="gaussian",only.hyperparam = FALSE,
+                           control.predictor=list(A=inla.stack.A(stack),compute=TRUE),
+                           control.compute = list(config = TRUE))
```

The result object now contains all relevant information for calculating the excursion sets. We calculate the level 0 positive excursion function using the EB and QC methods. We set $\alpha = 0.95$ and therefore only calculate the excursion functions up to that value, which saves computation time.

```
> res.eb = excursions.inla(result,alpha=0.95,method='EB',u=0,type='>',ind=iprd)
> res.qc = excursions.inla(result,alpha=0.95,method='QC',u=0,type='>',ind=iprd)
```

The results are plotted using the code below, see Figure 5 for the results.

```
> proj = inla.mesh.projector(mesh)
> image.plot(proj$x,proj$y,inla.mesh.project(proj,res.eb$mean),xlab="",ylab="")
> title("Mean")

> image.plot(proj$x,proj$y,inla.mesh.project(proj,res.eb$rho),xlab="",ylab="")
> title("Marginal probabilities")

> image.plot(proj$x,proj$y,inla.mesh.project(proj,res.eb$F),xlab="",ylab="")
> title("EB Excursion function")

> image.plot(proj$x,proj$y,inla.mesh.project(proj,res.qc$F),xlab="",ylab="")
> title("QC Excursion function")
```

Now, lets calculate the level 0 contour avoiding function

```
> res.c = excursions.inla(result,alpha=0.95,method='QC',u=0,type='!=',ind=iprd)
```

A useful way of representing the information contained in this function is to plot the product of the function and the sign of the posterior mean:

```
> col = tim.colors(21)
> col[11] = "#FFFFFF"
> image.plot(proj$x,proj$y,inla.mesh.project(proj,sign(res.c$mean)*res.c$F),
+          xlab="",ylab="",col=col)
```

The result is shown in Figure 6.

5.2 Zero-inflated negative binomial point processes

In this example, we estimate excursion sets for a marked point process with a zero inflated negative Binomial likelihood.

We start by loading the libraries we will use and define some parameters for the problem. `fields` and `ggplot2` are only used for making the figures.

```
> library(INLA)
> library(fields)
> library(ggplot2)
> library(excursions)
> a=0.5
> b=2
> n = 25^2
> n.obs = 3000
> kappa=sqrt(8)/4
> size = 1
> p.alpha = 0.1
```

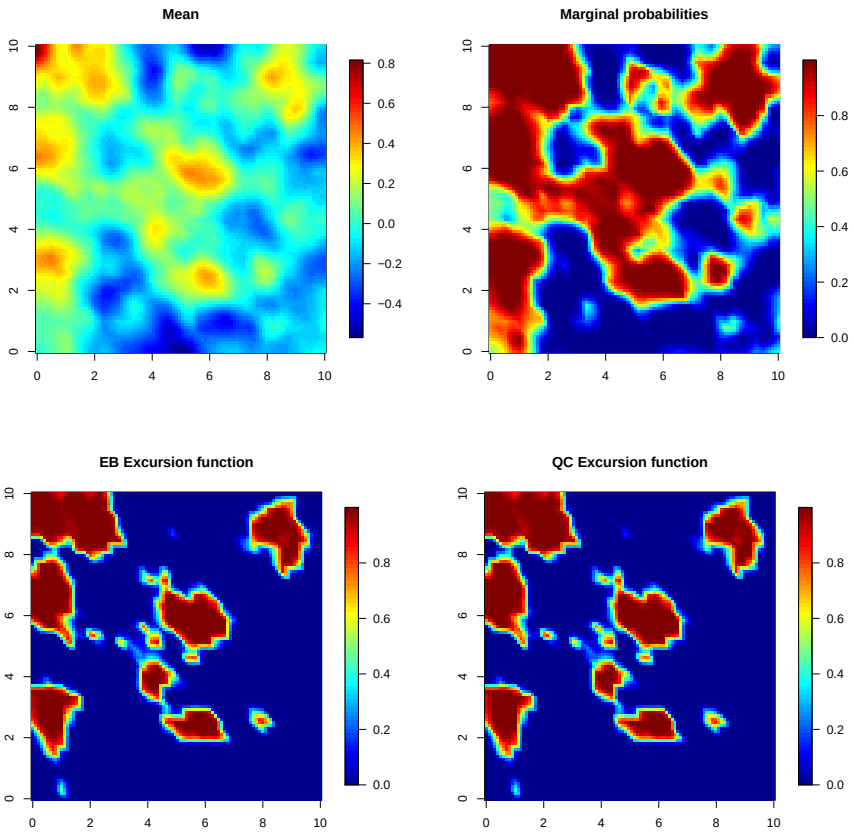

Figure 5: Posterior mean (upper left), marginal excursion probabilties (upper right), EB excursion function (lower left), and QC excursion function (lower right)

Next, we create the latent process and sample it according to the likelihood.

```
> x=seq(from=0,to=10,length.out=sqrt(n))
> lattice=inla.mesh.lattice(x=x,y=x)
> mesh=inla.mesh.create(lattice=lattice, extend=FALSE, refine=FALSE)
> spde = inla.spde2.matern(mesh, alpha=2)
> z = inla.qsample(n=1, Q=inla.spde.precision(spde, theta=c(0,log(kappa))))
> E = rep(1,n)
> E.obs = rep(1,n.obs)
> obs.loc = cbind(runif(n.obs)*diff(range(x))+min(x), runif(n.obs)*diff(range(x))+min(x))
> A=inla.spde.make.A(mesh, loc=obs.loc)
> eta = as.vector(a + b*z)
> mu = E*exp(eta)
> prob = size/(size + mu)
> eta.obs = as.vector(A %*% eta)
> mu.obs = E.obs*exp(eta.obs)
> prob.obs = size/(size + mu.obs)
> obs = rnbinom(n.obs, size=size, prob = prob.obs)
> obs[runif(length(obs),0,1)<p.alpha] = 0
```

Figure 7 contains a plot of the observations, generated using the following commands

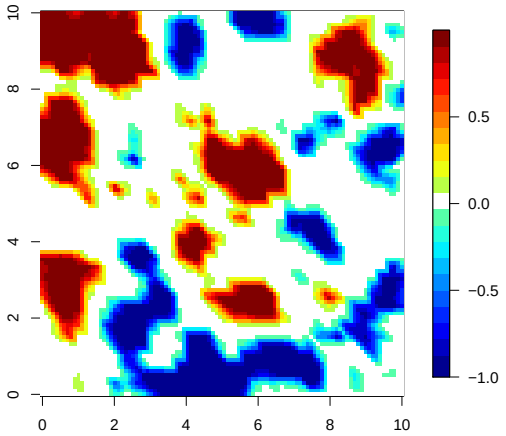

Figure 6: The product of the contour avoiding function and the sign of the posterior mean. White areas in the figure are likely to contain level 0 crossings, the process likely exceeds the level in the red areas, and the process likely is below the level in the blue areas.

```
> df = as.data.frame(cbind(obs.loc,obs))
> names(df) = c("x","y","z")
> cscale = tmp = rev(rainbow(8))
> cscale[1] = "#000000"
>  ggplot(df,aes(x,y,colour=z)) + geom_point() + scale_colour_gradientn(colours=cscale)
```

We now estimate the model parameters using `INLA`.

```
> ef = list(list(field=1:mesh$n))
> s.obs = inla.stack(data=list(y=obs,E=E.obs), A=list(A), effects=ef ,tag="obs")
> s.pred= inla.stack(data=list(y=NA,E=E), A=list(1), effects= ef,tag="pred")
> stack=inla.stack(s.obs,s.pred)
> iprd <- inla.stack.index(stack, 'pred')$data
> formula = y ~ -1 + f(field, model=spde)
> result = inla(formula, family = "zeroinflatednbinomial1",
+                                 data=inla.stack.data(stack),
+                                 control.compute = list(config = TRUE),
+                                 control.predictor=list(A=inla.stack.A(stack),
+                                 compute=TRUE,cdf=c(1)))
```

Plot the resulting fit of the latent field, see Figure 8.

```
> eta.fit = result$summary.fitted.values$mean[iprd]
> proj = inla.mesh.projector(mesh)
> image.plot(proj$x,proj$y,inla.mesh.project(proj,eta),xlab="",ylab="")
> title("eta")

> image.plot(proj$x,proj$y,inla.mesh.project(proj,eta.fit),xlab="",ylab="")
> title("eta fit")
```

Next, we calculate the excursion function using the `NIQC` method.

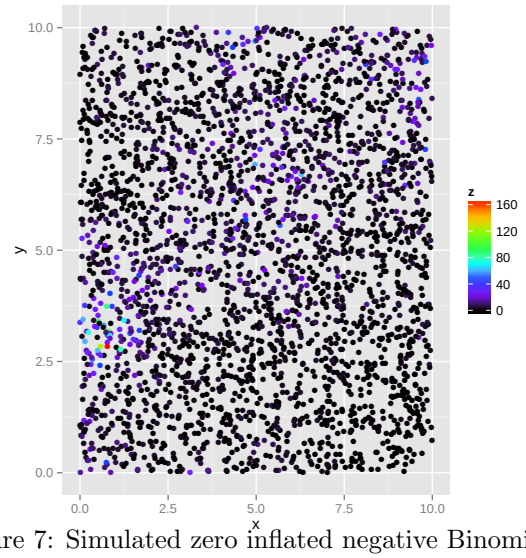

Figure 7: Simulated zero inflated negative Binomial data

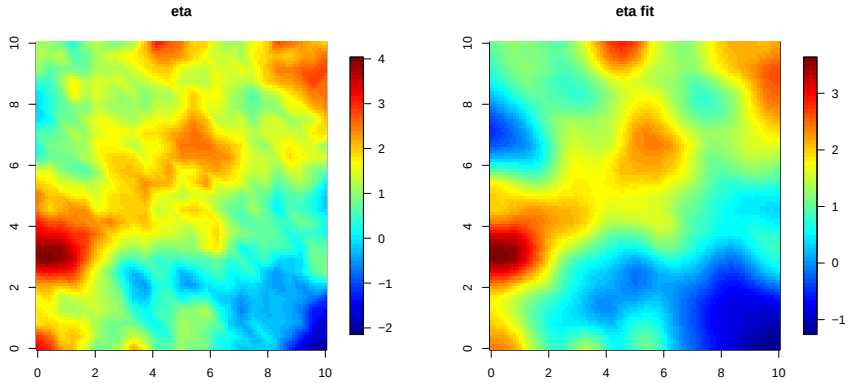

Figure 8: True latent field and estimate

```
> r.niqc =excursions.inla(result,method='NIQC',alpha=0.99, u=1,type='>', ind=iprd)
```

The resulting excursion function is shown in Figure 9, generated as

```
> image.plot(proj$x,proj$y,inla.mesh.project(proj,r.niqc$rho))
> title("marginal P(eta>0)")

> image.plot(proj$x,proj$y,inla.mesh.project(proj,r.niqc$F))
> title("NIQC Excursion function")
```

References

P. Amestoy, T. A. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

P. Amestoy, T. A. Davis, and I. S. Duff. Algorithm 837: Amd, an approximate minimum degree ordering algorithm. *ACM Transactions on Mathematical Software*, 30(3):381–388, 2004.

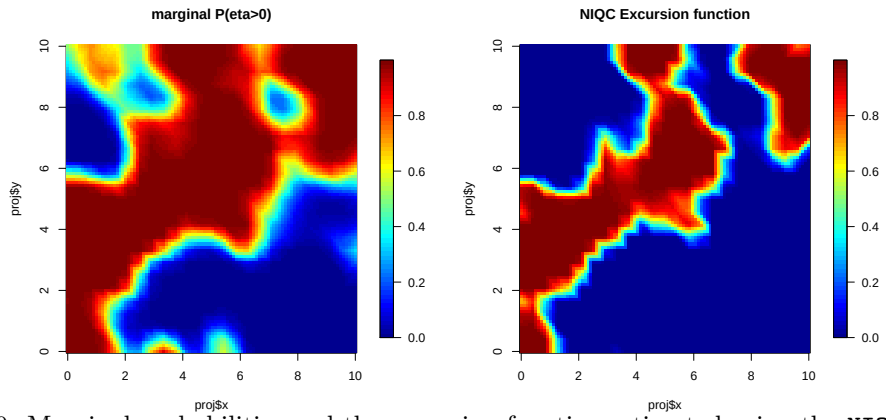

Figure 9: Marginal probabilities and the excursion function estimated using the `NIQC` method.

E. Anderson, Z. Bai, C. Bischof, L.S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, S. Hammarling, A. Greenbaum, A. McKenney, , and D. Sorensen. *LAPACK Users' guide (third ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

M. M. Beaky, R. J. Scherrer, and J. V. Villumsen. Topology of large-scale structure in seeded hot dark matter models. *Astrophys. J.*, 387:443–448, 1992.

D. Bolin and F. Lindgren. Excursion and contour uncertainty regions for latent Gaussian models (in press). *J. Roy. Statist. Soc. Ser. B Stat. Methodol.*, 2013.

D. Bolin, J. Lindström, L. Eklundh, and F. Lindgren. Fast estimation of spatially dependent temporal vegetation trends using Gaussian Markov random fields. *Comput. Statist. and Data Anal.*, 53:2885–2896, 2009.

Michela Cameletti, Finn Lindgren, Daniel Simpson, and Håvard Rue. Spatio-temporal modeling of particulate matter concentration through the SPDE approach. *AStA Advances in Statistical Analysis*, pages 1–23, 2013.

Yanqing Chen, Timothy A Davis, William W Hager, and Sivasankaran Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3):22, 2008.

Jack J Dongarra, Jeremy Du Croz, Sven Hammarling, and Iain S Duff. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software (TOMS)*, 16(1): 1–17, 1990.

L. Eklundh and L. Olsson. Vegetation index trends for the African Sahel 1982–1999. *J. Geophys. Res.*, 30:1430–1433, 2003.

R. Furrer, R. Knutti, S. R. Sain, D. Nychka, and Meehl G. A. Spatial patterns of probabilistic temperature change projections from a multivariate bayesian analysis. *Geophys. Res. Lett.*, 34, 2007.

Mark Galassi and Brian Gough. Gnu scientific library: reference manual, 2006.

J. Marchini and A. Presanis. Comparing methods of analyzing fMRI statistical parametric maps. *NeuroImage*, 22:1203–1213, 2003.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. URL `http://www.R-project.org/`.

H Rue, S Martino, and N Chopin. Approximate Bayesian inference for latent Gaussian models using integrated nested Laplace approximations (with discussion). *J. Roy. Statist. Soc. Ser. B Stat. Methodol.*, 71(2):319–392, 2009.

S. R. Sain, R. Furrer, and N. Cressie. A spatial analysis of multivariate output from regional climate models. *Ann. Appl. Statist.*, 5:150–175, 2011.