

## Session 8.2: Introduction to SPDE model with R-INLA

Bayesian modelling for Spatial and Spatio-temporal data, Imperial College London

# Learning Objectives

At the end of this session you should be able to:

- understand the basics of the Stochastic Partial Differential Equation (**SPDE**) approach;
- implement the SPDE approach using the R-INLA package;
- perform spatial prediction and mapping.

The topics treated in this lecture can be found in:

- Sections 6.5:6.8 of the book **Spatial and Spatio-Temporal Bayesian models with R-INLA**
- Chapters 8 and 9 of the book **Geospatial Health Data: Modeling and Visualization with R-INLA and Shiny**  
<https://www.paulamoraga.com/book-geospatial/index.html>

# Outline

1. Basics of the SPDE approach
2. The SPDE approach with R-INLA
3. Spatial prediction using SPDE approach

# The SPDE approach

# A common model for geostatistical (noisy) data

- Usually the following (mixed-effects) model is assumed

$$y(\mathbf{s}) = \mu(\mathbf{s}) + \xi(\mathbf{s}) + \epsilon(\mathbf{s})$$

where

- $\mu(\mathbf{s})$  is the so-called **large scale** component, including linear or non linear of covariates.
- $\xi(\mathbf{s})$  is a zero mean latent **Gaussian spatial process** commonly assumed to be stationary and isotropic with covariance function  $Cov(\xi(\mathbf{s}_i), \xi(\mathbf{s}_j))$  which depends only on the distance between the locations.
- $\epsilon(\mathbf{s})$  represents the Gaussian **measurement error** (independent from  $\xi(\mathbf{s})$ ) and its variance ( $\sigma_\epsilon^2$ ) is usually known as **nugget effect**.

# A common model for geostatistical (noisy) data

- Usually the following (mixed-effects) model is assumed

$$y(\mathbf{s}) = \mu(\mathbf{s}) + \xi(\mathbf{s}) + \epsilon(\mathbf{s})$$

where

- $\mu(\mathbf{s})$  is the so-called **large scale** component, including linear or non linear of covariates.
- $\xi(\mathbf{s})$  is a zero mean latent **Gaussian spatial process** commonly assumed to be stationary and isotropic with covariance function  $Cov(\xi(\mathbf{s}_i), \xi(\mathbf{s}_j))$  which depends only on the distance between the locations.
- $\epsilon(\mathbf{s})$  represents the Gaussian **measurement error** (independent from  $\xi(\mathbf{s})$ ) and its variance ( $\sigma_\epsilon^2$ ) is usually known as **nugget effect**.

Given the data from  $n$  locations  $(\mathbf{s}_1, \dots, \mathbf{s}_n)$  we have:

$$\begin{aligned} y(\mathbf{s}_i) \mid \mu(\mathbf{s}_i), \xi(\mathbf{s}_i), \sigma_\epsilon^2 &\sim \text{Normal}(\mu(\mathbf{s}_i) + \xi(\mathbf{s}_i), \sigma_\epsilon^2) \\ \xi &\sim \text{GF}(\mathbf{0}, \Sigma) \end{aligned}$$

where  $\Sigma$  is a **dense** matrix defined by a spatial covariance function  $\mathcal{C}(\|\mathbf{s}_i - \mathbf{s}_j\|)$ .

- Several functions are available for the **spatial covariance function** (e.g. exponential, Matérn, spherical, etc.) parameterized by some parameters, e.g. spatial variance, range (Banerjee, Carlin, and Gelfand, 2014).

# Matérn covariance function

## Definition

## The effect of range

The **Matérn covariance function** is defined by

$$\text{Cov}(\xi(\mathbf{s}_i), \xi(\mathbf{s}_j)) = \text{Cov}(\xi_i, \xi_j) = \frac{\sigma^2}{\Gamma(\lambda)2^{\lambda-1}}(\kappa\|\mathbf{s}_i - \mathbf{s}_j\|)^{\lambda}K_{\lambda}(\kappa\|\mathbf{s}_i - \mathbf{s}_j\|)$$

where:

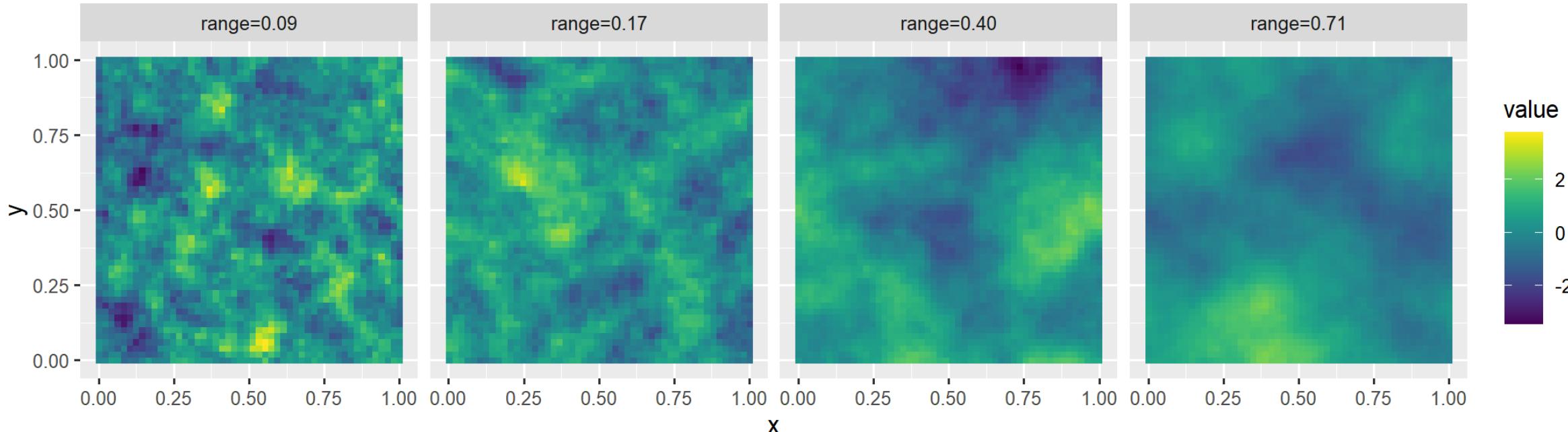
- $\|\mathbf{s}_i - \mathbf{s}_j\|$  is the Euclidean distance between two generic locations  $\mathbf{s}_i, \mathbf{s}_j \in \mathbb{R}^2$ ,
- $\sigma^2$  is the marginal variance of the GF,
- $K_{\lambda}$  denotes the modified Bessel function of second kind and order  $\lambda > 0$ , which measures the degree of **smoothness** of the process (it is usually kept fixed due to poor identifiability),
- $\kappa > 0$  is a **scale parameter** related to the **range**  $r$ , i.e. the distance at which the spatial correlation becomes almost null. Typically, the empirically derived definition for the range is  $r = \frac{\sqrt{8\lambda}}{\kappa}$  (Lindgren, Rue, and Lindström, 2011), with  $r$  corresponding to the distance at which the spatial correlation is close to 0.1, for each  $\lambda \geq 1/2$ .
- The Matérn family is a very flexible class of covariance functions able to cover a wide range of spatial fields.

# Matérn covariance function

Definition

The effect of range

Simulation of different GFs with  $\sigma = 1$  and different ranges:



# Model based approach for estimation

- In the Bayesian framework, the classical approach for model estimation is Markov chain Monte Carlo methods (MCMC) considering that the likelihood function is a multivariate Gaussian distribution (see for example the `spBayes` and `spTimer` R packages).
- This requires to compute the Cholesky factorization of the dense covariance matrix  $\Sigma$  which is computationally very expensive. This is known as **big  $n$  problem**.
- The **stochastic partial differential equation (SPDE) approach** (Lindgren, Rue, and Lindström, 2011) is an alternative to the use of MCMC: it represents the continuous spatial process  $\xi(s)$  with Matérn covariance function using a discretely indexed spatial random process (i.e., a Gaussian Markov Random Field - GMRF), which is characterized by a sparse precision matrix and enjoys computational benefits in terms of fast inference.

# Introduction to the SPDE approach

# The SPDE approach: main references

Seminal paper

JSS paper

SPDE book

Other references

- Lindgren, F., Rue, H. and Lindstrom, J. (2011), An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73: 423-498. <https://doi.org/10.1111/j.1467-9868.2011.00777.x>

Journal of the  
Royal Statistical Society

*J. R. Statist. Soc. B* (2011)  
73, Part 4, pp. 423–498



## An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach

Finn Lindgren and Håvard Rue

*Norwegian University of Science and Technology, Trondheim, Norway*

and Johan Lindström

*Lund University, Sweden*

# The SPDE approach: main references

Seminal paper

JSS paper

SPDE book

Other references

- Lindgren, F., & Rue, H. (2015). Bayesian Spatial Modelling with R-INLA. *Journal of Statistical Software*, 63(19), 1–25. <https://doi.org/10.18637/jss.v063.i19>



---

*Journal of Statistical Software*

January 2015, Volume 63, Issue 19.

<http://www.jstatsoft.org/>

---

## Bayesian Spatial Modelling with R-INLA

**Finn Lindgren**  
University of Bath

**Håvard Rue**  
Norwegian University of  
Science and Technology

# The SPDE approach: main references

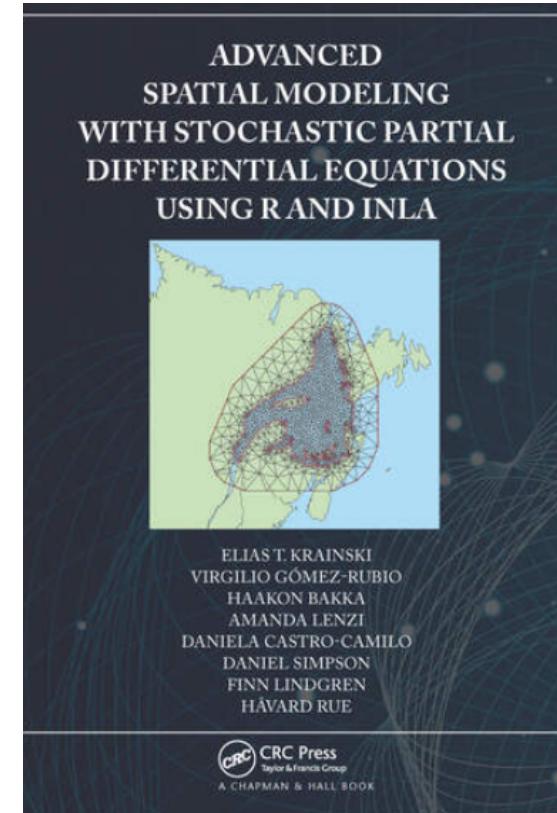
Seminal paper

JSS paper

SPDE book

Other references

- E.T. Krainski, V. Gómez-Rubio, H. Bakka, A. Lenzi, D. Castro-Camilo, D. Simpson, F. Lindgren and H. Rue (2019) Advanced Spatial Modeling with Stochastic Partial Differential Equations Using R and INLA, CRC Press.
- <https://becarioprecario.bitbucket.io/spde-gitbook/>



# The SPDE approach: main references

Seminal paper

JSS paper

SPDE book

Other references

- Lindgren F., Bolin D., Rue H., **The SPDE approach for Gaussian and non-Gaussian fields: 10 years and still running**, Spatial Statistics, Volume 50, 2022, <https://doi.org/10.1016/j.spasta.2022.100599>
- Bakka H, Rue H, Fuglstad GA, et al. **Spatial modeling with R-INLA: A review**. WIREs Comput Stat. 2018; 10:e1443. <https://doi.org/10.1002/wics.1443>
- Moraga P, **Geospatial Health Data. Modeling and Visualization with R-INLA and Shiny**, 2020, CRC press, [www.paulamoraga.com/book-geospatial-info/](http://www.paulamoraga.com/book-geospatial-info/)

# The SPDE approach for Matérn GF

- The starting point is the **linear fractional stochastic partial differential equation (SPDE)**

$$(\kappa^2 - \Delta)^{\alpha/2}(\tau\xi(\mathbf{s})) = \mathcal{W}(\mathbf{s})$$

where  $\mathbf{s} \in \mathbb{R}^d$ ,  $\Delta$  is the Laplacian operator,  $\alpha > 0$  is the **smoothness** term,  $\kappa > 0$  is the scale parameter,  $\tau$  controls the variance and  $\mathcal{W}(\mathbf{s})$  is a Gaussian spatial white noise process (with unit variance).

- Whittle in 1954 showed that the exact and stationary solution to this SPDE is the stationary Gaussian field  $\xi(\mathbf{s})$  with Matérn covariance function.
- Lindgren, Rue, and Lindström (2011) represent the solution of the SPDE using the finite element method (this is possible only for some values of the smoothness parameter).
- In  $\mathbb{R}^2$  the link between the SPDE parameters  $\tau, \alpha, \kappa$  and the Matérn function parameters  $\sigma^2, \lambda, \kappa$  is given by

$$\lambda = \alpha - 1 \quad \sigma^2 = \frac{\Gamma(\lambda)}{\Gamma(\alpha)(4\pi)\kappa^{2\lambda}\tau^2}$$

- In R-INLA the default value for the smoothness parameter is  $\alpha = 2 \rightarrow \lambda = 1$ . Consequently

$$r = \sqrt{8\lambda}/\kappa = \sqrt{8}/\kappa \text{ (see slide 6)} \quad \sigma^2 = 1/(4\pi\kappa^2\tau^2)$$

# Piecewise linear approximation

General result

1D    2D

- The solution to the SPDE can be approximated through a **basis function representation** defined on a **triangulation** of the spatial domain

$$\xi(s) = \sum_{g=1}^G \varphi_g(s) \tilde{\xi}_g$$

where  $G$  is the total number of triangulation vertices,  $\{\varphi_g\}$  is the set of (deterministic) basis functions and  $\{\tilde{\xi}_g\}$  are zero mean Gaussian distributed weights.

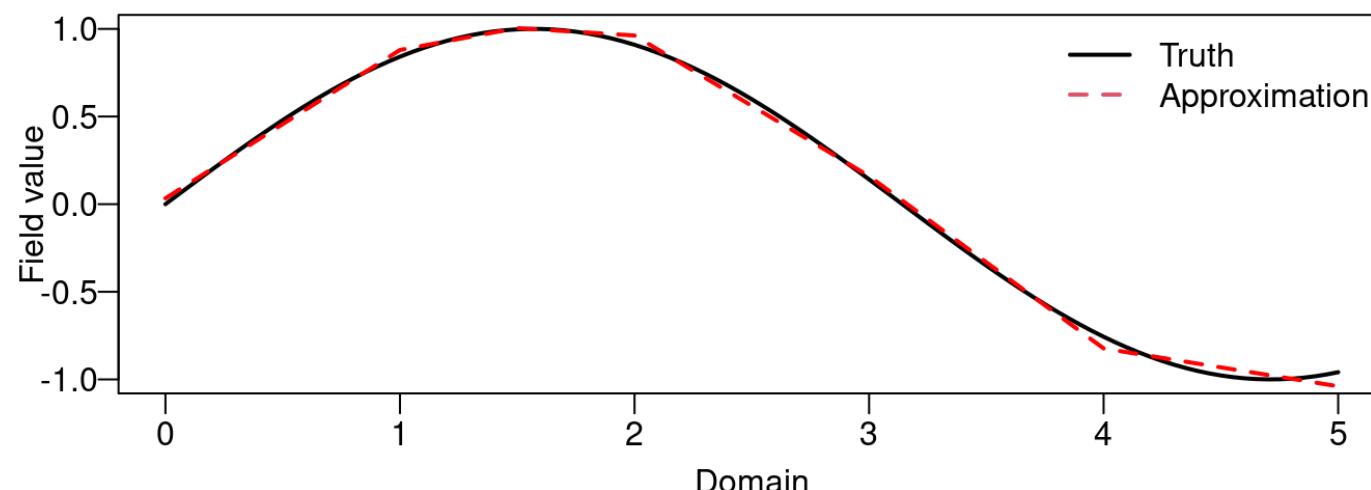
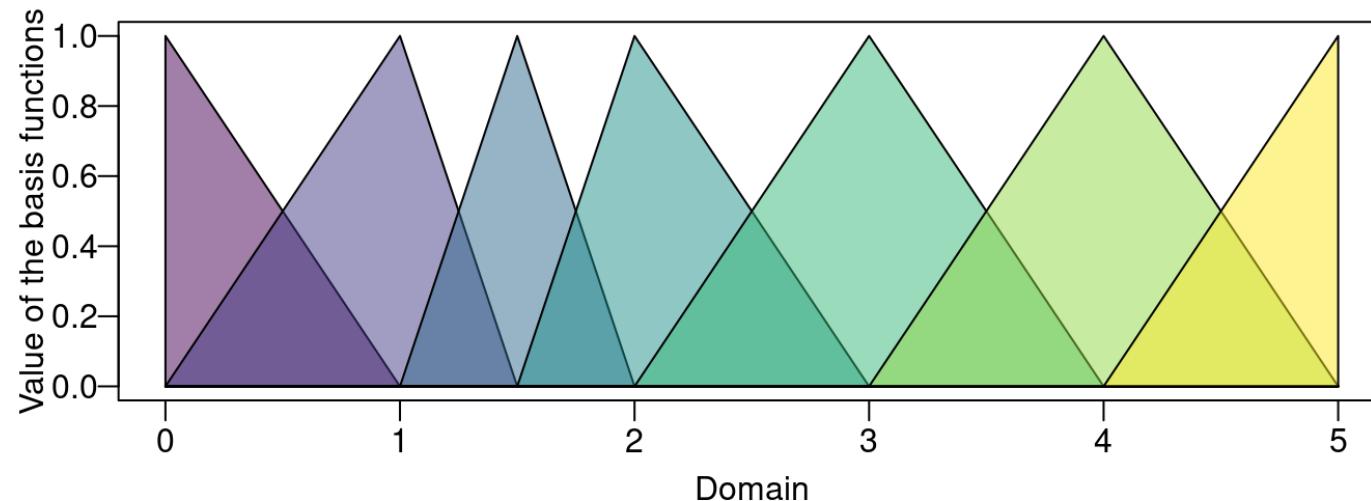
- In order to obtain a Markov structure, the basis functions are chosen to be **piecewise linear** in each triangle, i.e.  $\varphi_g$  is 1 at vertex  $g$  and 0 at all other vertices.

# Piecewise linear approximation

General result

1D

2D

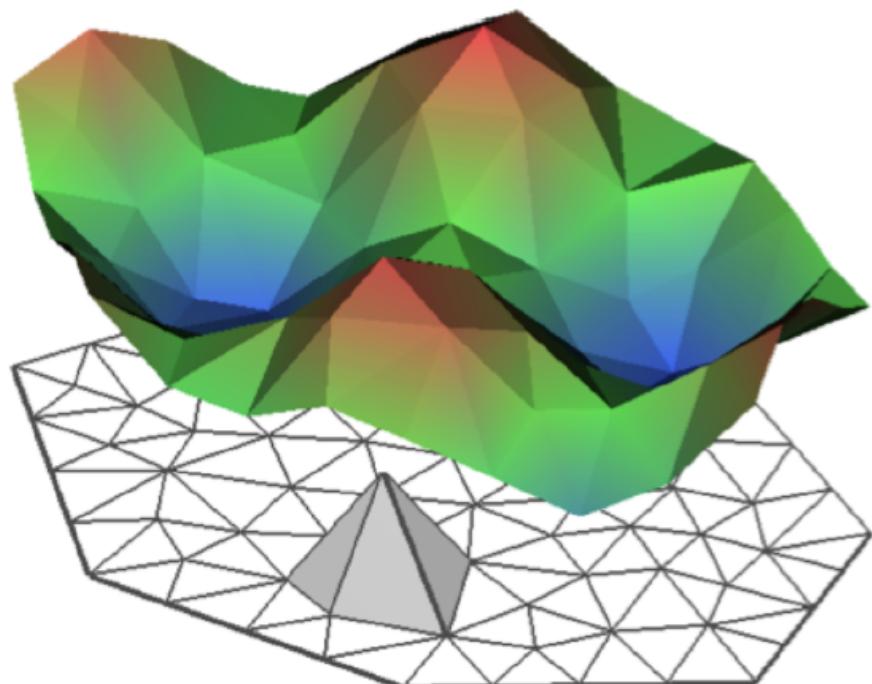
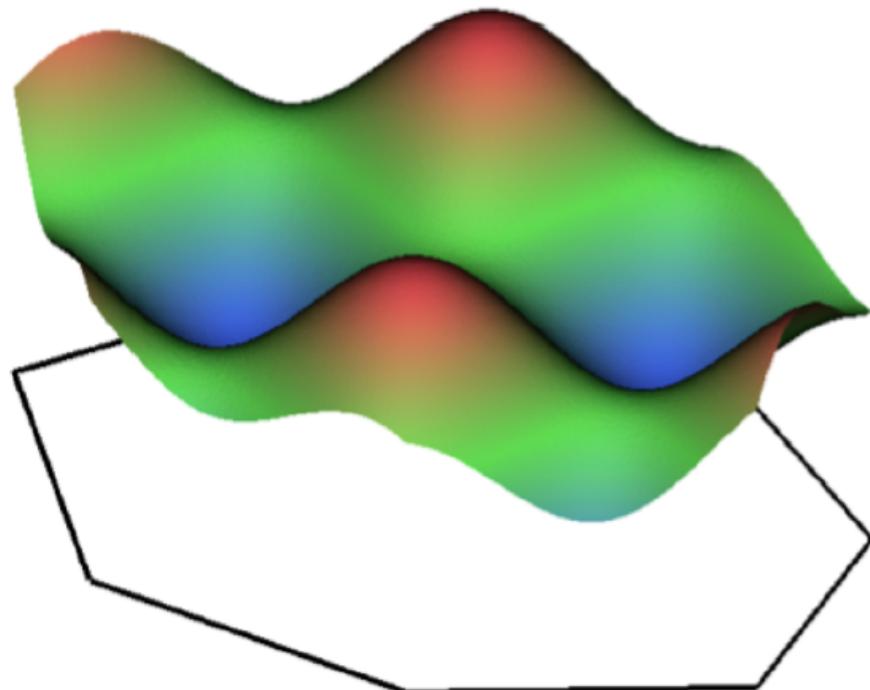


# Piecewise linear approximation

General result

1D

2D



# Piecewise linear approximation

- The precision matrix  $\mathbf{Q}$  for the Gaussian weight vector  $\tilde{\boldsymbol{\xi}} = \{\tilde{\xi}_1, \dots, \tilde{\xi}_G\}$  is given by

$$\mathbf{Q} = \tau^2 \left( \kappa^4 \mathbf{C} + 2\kappa^2 \mathbf{G} + \mathbf{G}\mathbf{C}^{-1}\mathbf{G} \right)$$

where

- the generic element of the diagonal matrix  $\mathbf{C}$  is  $C_{ii} = \int \varphi_i(\mathbf{s}) d\mathbf{s}$ ,
- the generic element of the sparse matrix  $\mathbf{G}$  is  $G_{ij} = \int \nabla \varphi_i(\mathbf{s}) \nabla \varphi_j(\mathbf{s}) d\mathbf{s}$  (where  $\nabla$  denotes the gradient),

The precision matrix  $\mathbf{Q}$ , whose elements depend on  $\tau$  and  $\kappa$ , is sparse and consequently  $\tilde{\boldsymbol{\xi}}$  is a GMRF distributed as  $\text{Normal}(\mathbf{0}, \mathbf{Q}^{-1})$ : it represents the approximated solution to the SPDE.

## Take home message

- Do the modelling using GF and the computations using the GMRF representation (computational advantages thanks to algorithms for sparse matrices):

$$\boldsymbol{\xi} \sim \text{Normal}(\mathbf{0}, \boldsymbol{\Sigma}) \implies \tilde{\boldsymbol{\xi}} \sim \text{Normal}(\mathbf{0}, \boldsymbol{Q}^{-1})$$

# The SPDE approach with R-INLA

# SPDE toy example

Data Plot

We use the SPDEtoy dataset , consisting in 200 simulated values for the variable y which refer to as many randomly sampled locations in the unit square area delimited by the points (0, 0) and (1, 1) and with coordinates given by s1 and s2.

```
> library(INLA)
> data(SPDEtoy)
> dim(SPDEtoy)
```

```
[1] 200    3
```

```
> head(SPDEtoy, n=3)
```

	s1	s2	y
1	0.08265625	0.05640625	11.521206
2	0.61230625	0.91680625	5.277960
3	0.16200625	0.35700625	6.902959

```
> summary(SPDEtoy$y)
```

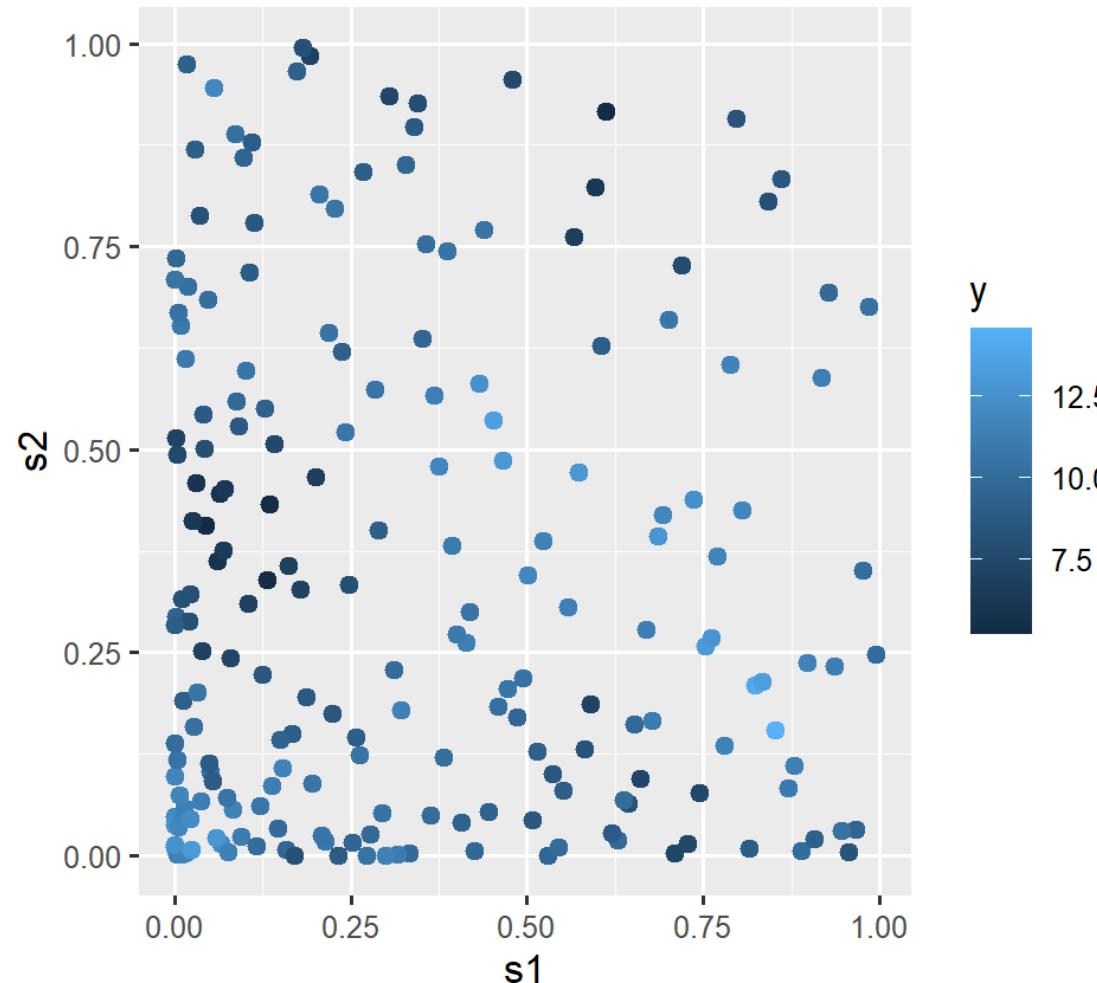
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
5.210	8.688	10.023	9.858	11.307	14.600

# SPDE toy example

Data

Plot

```
> library(tidyverse)
> library(INLA)
>
> SPDEtoy %>%
+   ggplot() +
+   geom_point(aes(s1,s2, col=y),size=2)
```



## SPDE toy example: model for simulation

- The model used for simulating the SPDEtoy data assumes that the distribution of the observation  $y_i$  is

$$y_i \mid \eta_i, \sigma_e^2 \sim \text{Normal}(\eta_i, \sigma_e^2) \quad i = 1, \dots, 200$$

where  $\sigma_e^2$  is the variance of the zero mean measurement error  $e_i$  which is supposed to be Gaussian iid.

- The response mean, which coincides with the **linear predictor**, is defined as

$$\eta_i = b_0 + \xi_i$$

and includes the intercept  $b_0$  and a random effect represented by  $\xi_i$ , which is the realization of the latent GF  $\xi(s) \sim \text{MVNormal}(\mathbf{0}, \Sigma)$ . The covariance matrix  $\Sigma$  is defined by the Matérn spatial covariance function.

- The parameter values chosen for simulating the data are:  $b_0 = 10$ ,  $\sigma_e^2 = 0.3$ ,  $\sigma^2 = 5$ ,  $\kappa = 7$ ,  $r = \frac{\sqrt{8}}{\kappa} = 0.404$ .

# The SPDE representation and the projector matrix

- Using the SPDE basis function representation, the linear predictor  $\eta_i$  can be rewritten as

$$\eta_i = b_0 + \sum_{g=1}^G \varphi_g(\mathbf{s}_i) \tilde{\xi}_g$$

where  $\varphi_g(\mathbf{s}_i)$  is the value of the  $g$ -th basis function evaluated in  $\mathbf{s}_i$ .

- More generally it is possible to express the linear predictor as

$$\eta_i = b_0 + \sum_{g=1}^G A_{ig} \tilde{\xi}_g$$

with  $A_{ig} = \varphi_g(\mathbf{s}_i)$  being the generic element of the sparse matrix  $\mathbf{A}$  (known as **projector matrix**) which maps the GMRF  $\tilde{\xi}$  from the  $G$  triangulation vertices to the  $n$  observation locations. This allows the SPDE model to be treated as standard indexed random effects.

# Mesh

- The SPDE approach is based on a **triangulation** of the spatial domain given by the **mesh**.
- The definition of the mesh is a **trade-off** between the accuracy of the GMRF representation and computational costs, both depending on the number of vertices used in the triangulation: the bigger the number of mesh triangles, the finer the GF approximation but the higher the computational costs.
- To create the mesh in R-INLA we use the helper function `inla.mesh.2d`. The arguments for a two-dimensional mesh construction can be checked using `args(inla.mesh.2d)`.
- Here we are going to start with the following options:
  - `loc` or `loc.domain`: specify information about the spatial domain
  - `max.edge`: specify the largest allowed triangle edge length. If a vector of two values is provided, the spatial domain is divided into an inner and an outer area whose triangle resolution is specified by `max.edge` (the higher the value for `max.edge` the lower the resolution and the accuracy).

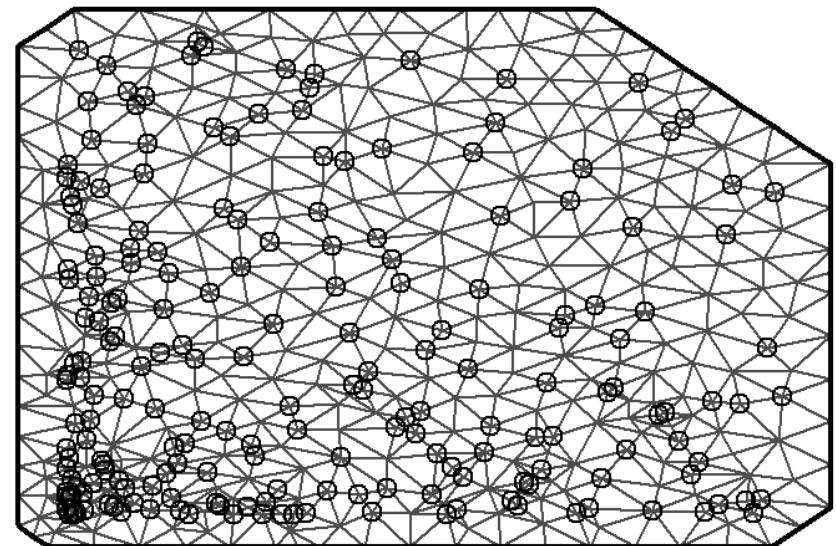
# Mesh: changing max.edge

mesh0    mesh0 inlabru    mesh1    mesh2

```
> coords = as.matrix(SPDEtoy[, 1:2])
> mesh0 = inla.mesh.2d(loc = coords,
+                         max.edge = 0.1)
```

```
> plot(mesh0)
> points(coords)
```

## Constrained refined Delaunay triangulation



# Mesh: changing max.edge

mesh0

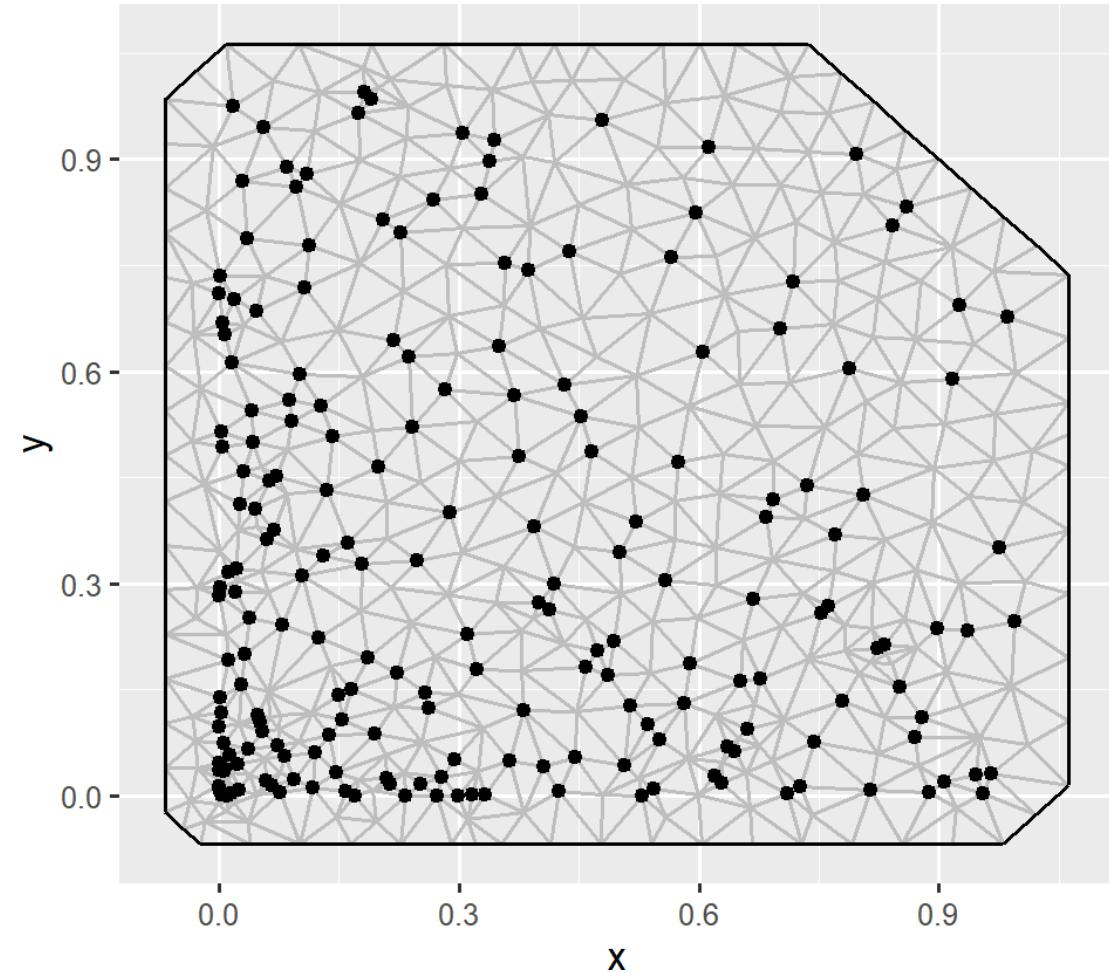
mesh0 inlabru

mesh1

mesh2

```
> coords = as.matrix(SPDEtoy[, 1:2])
> mesh0 = inla.mesh.2d(loc = coords,
+                         max.edge = 0.1)
```

```
> library(tidyverse)
> library(inlabru)
> ggplot() +
+   gg(mesh0) +
+   geom_point(data = data.frame(coords),
+              aes(s1, s2))
```

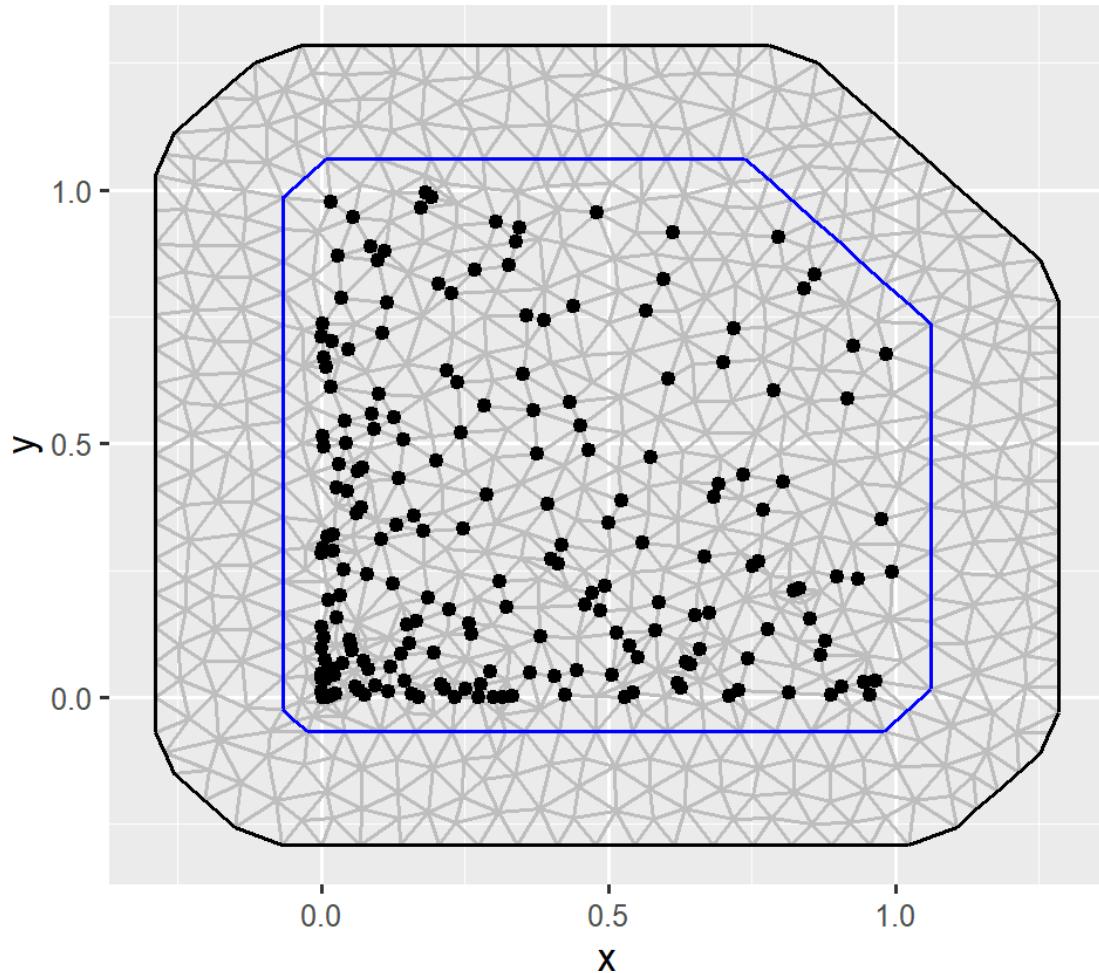


## Mesh: changing max.edge

mesh0    mesh0.inlabru    **mesh1**    mesh2

```
> mesh1 = inla.mesh.2d(loc = coords,  
+                         max.edge = c(0.1, 0.1))
```

```
> ggplot() +  
+   gg(mesh1) +  
+   geom_point(data = data.frame(coords),  
+              aes(s1, s2))
```



## Mesh: changing max.edge

mesh0

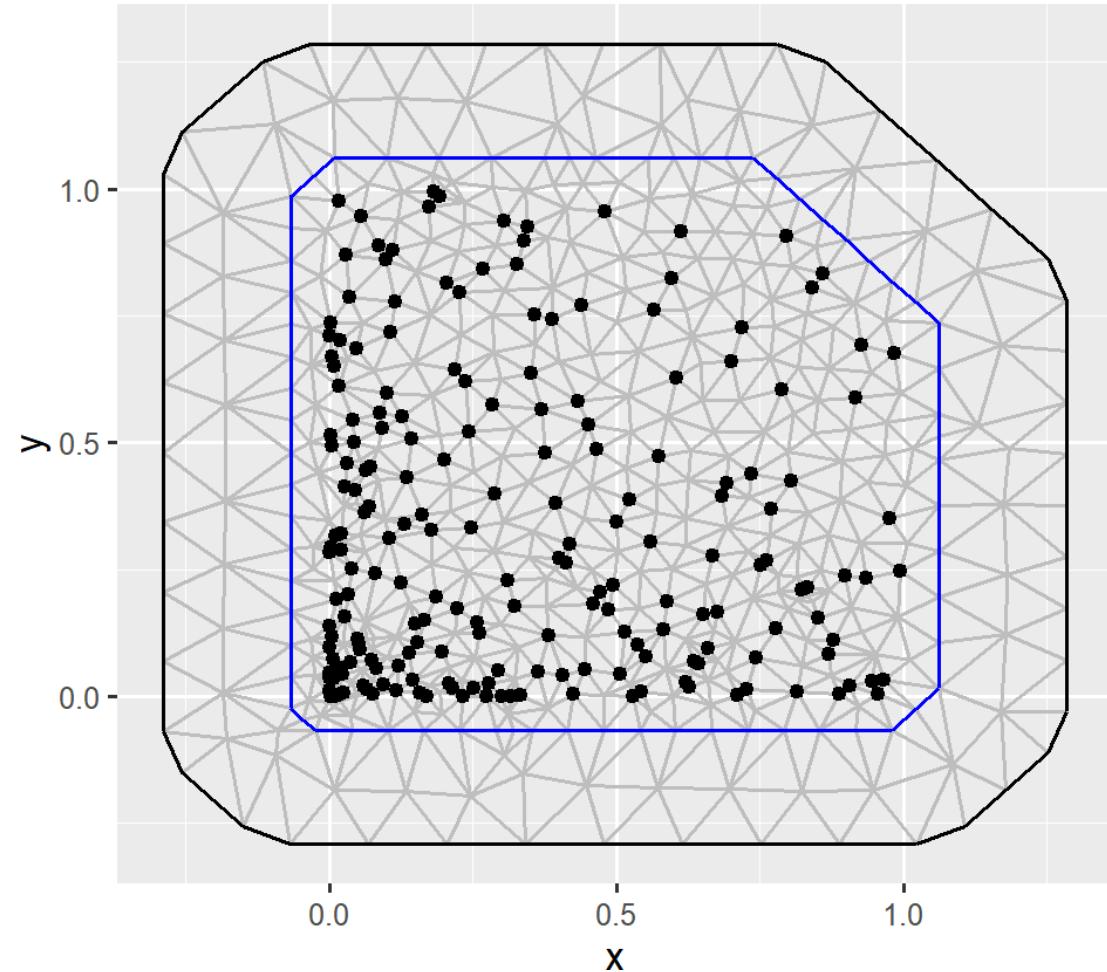
mesh0 inlabru

mesh1

mesh2

```
> mesh2 = inla.mesh.2d(loc = coords,  
+                         max.edge = c(0.1, 0.2))
```

```
> ggplot() +  
+   gg(mesh2) +  
+   geom_point(data = data.frame(coords),  
+               aes(s1, s2))
```



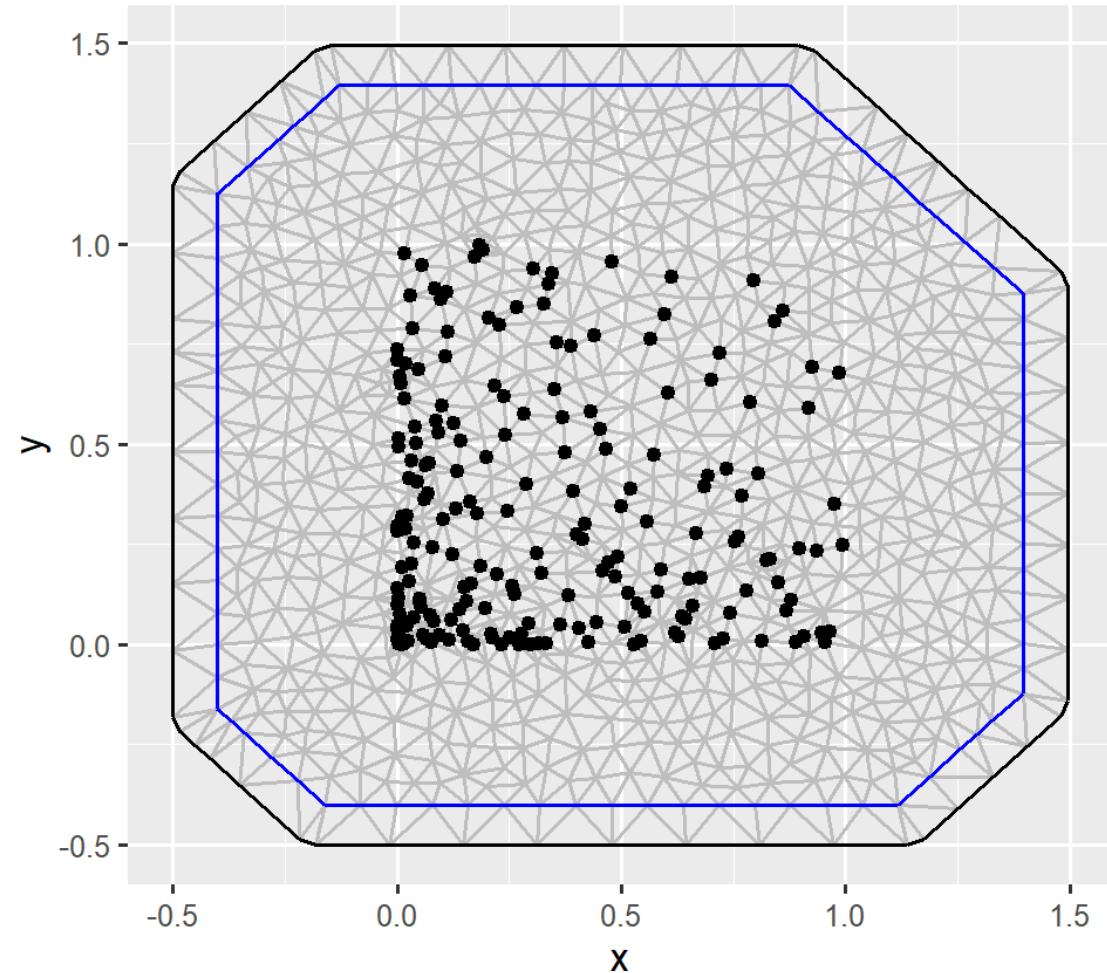
# Mesh optional arguments: offset

mesh3    mesh4

The option `offset` of the `inla.mesh.2d` function can be used to define how much the domain should be extended in the inner and outer part. The default values are `offset = c(-0.05, -0.15)`.

```
> mesh3 = inla.mesh.2d(loc = coords,  
+                         max.edge = c(0.1, 0.2),  
+                         offset = c(0.4, 0.1))
```

```
> ggplot() +  
+   gg(mesh3)+  
+   geom_point(data = data.frame(coords),  
+               aes(s1, s2))
```

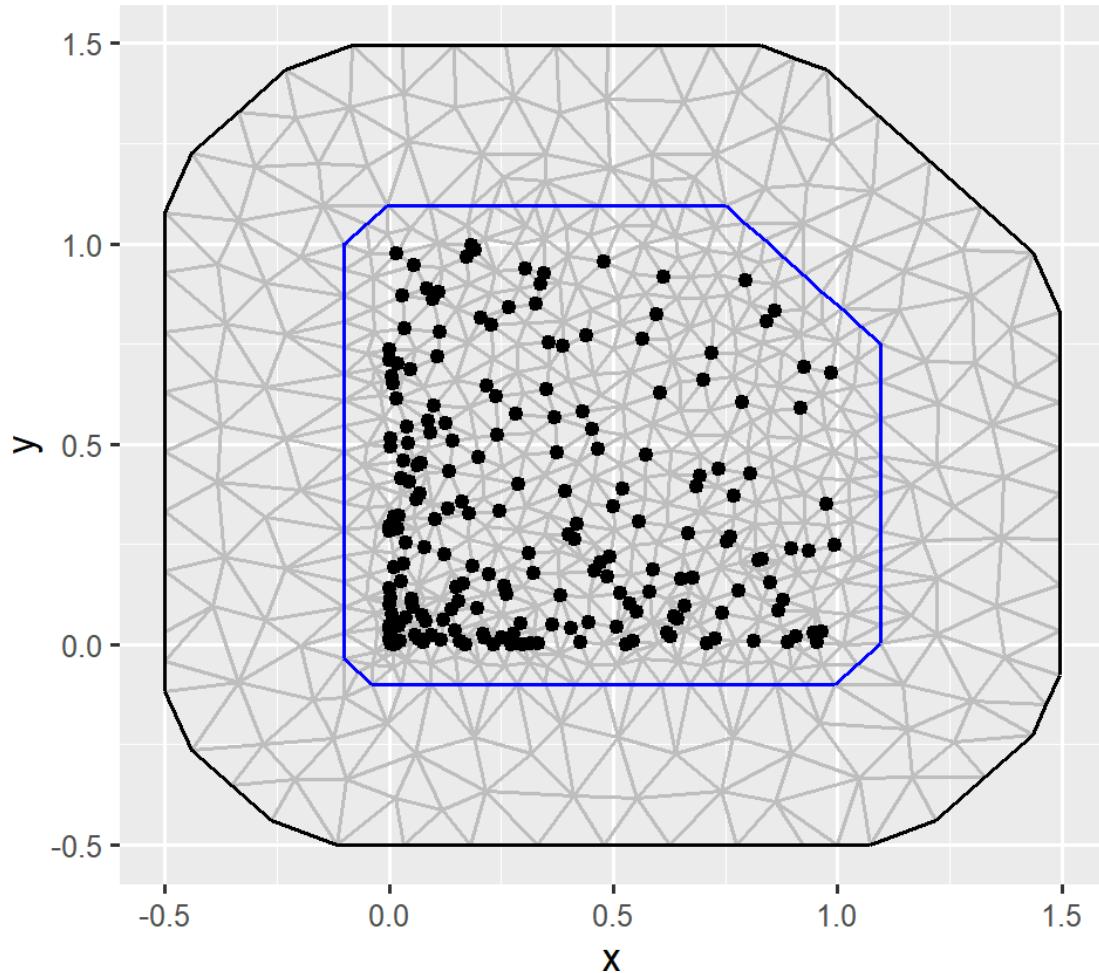


# Mesh optional arguments: offset

mesh3    mesh4

```
> mesh4 = inla.mesh.2d(loc = coords,  
+                         max.edge = c(0.1, 0.2),  
+                         offset = c(0.1, 0.4))
```

```
> ggplot() +  
+   gg(mesh4) +  
+   geom_point(data = data.frame(coords),  
+               aes(s1, s2))
```



# Mesh optional arguments cutoff

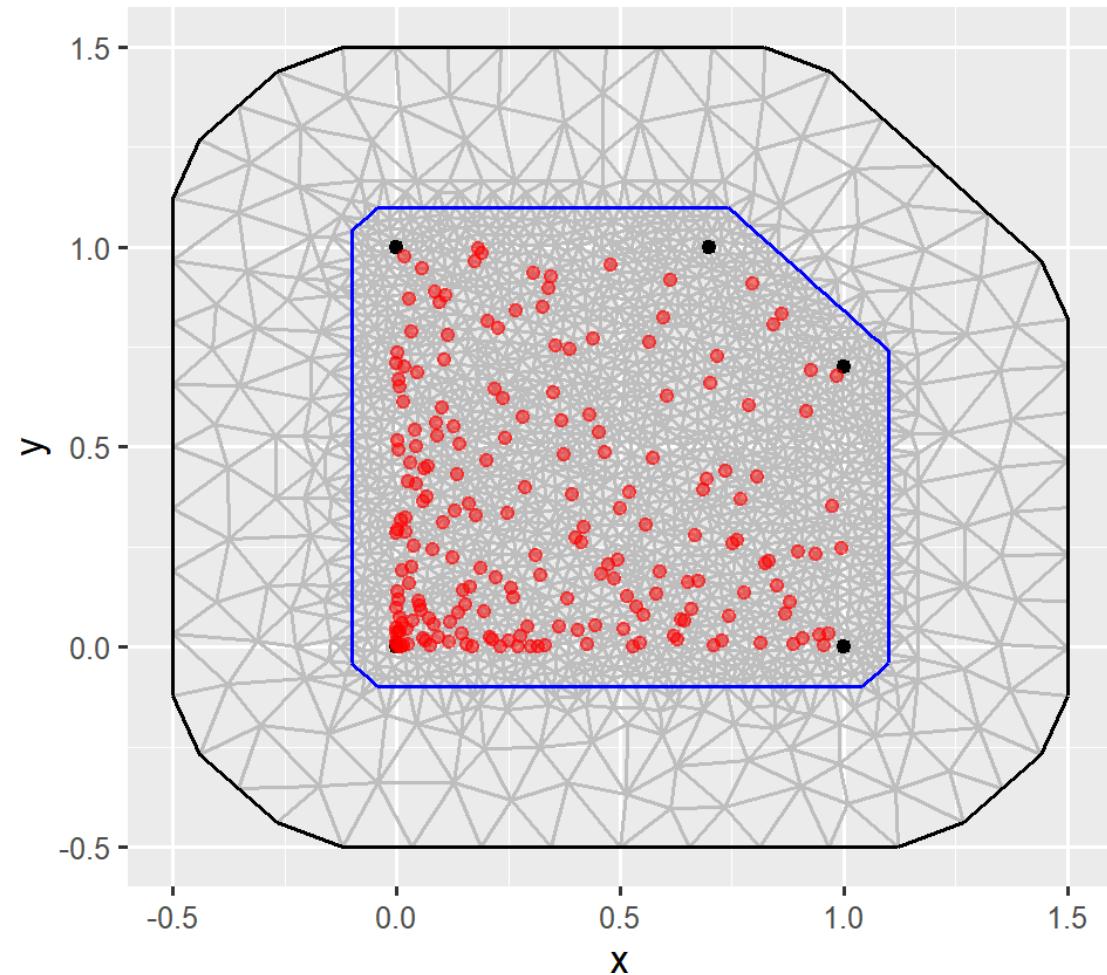
domain    mesh5    mesh6

Instead of the point coordinates, it is possible to use other point locations used to determine the domain extent.

```
> domain=matrix(cbind(c(0,1,1,0.7,0),
+                      c(0,0,0.7,1,1)), ncol=2)
> mesh5domain=inla.mesh.2d(loc.domain = domain,
+                           max.edge = c(0.04, 0.2),
+                           offset = c(0.1, 0.4))
```



```
> ggplot() +
+   gg(mesh5domain) +
+   geom_point(data = data.frame(domain),
+              aes(X1, X2)) +
+   geom_point(data = data.frame(coords),
+              aes(s1, s2), col = "red",
+              alpha = 0.5)
```



# Mesh optional arguments cutoff

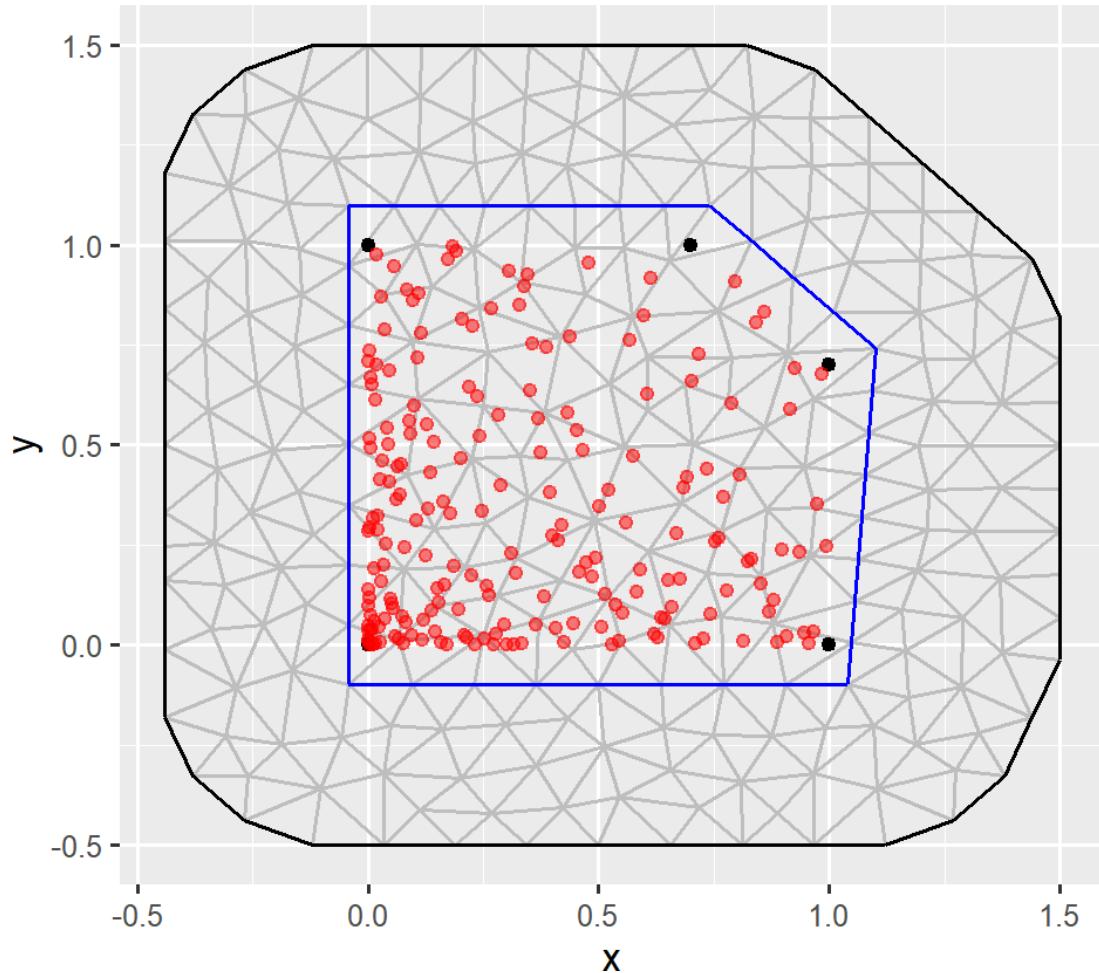
domain    **mesh5**    mesh6

The option `cutoff` can be used to avoid building too many small triangles around clustered data locations (the default value is equal to 0).

```
> mesh5 = inla.mesh.2d(loc.domain = domain,  
+                         max.edge = c(0.04, 0.2),  
+                         cutoff = 0.5,  
+                         offset = c(0.1, 0.4))
```



```
> ggplot() +  
+   gg(mesh5) +  
+   geom_point(data = data.frame(domain),  
+               aes(X1, X2)) +  
+   geom_point(data = data.frame(coords),  
+               aes(s1, s2), col = "red",  
+               alpha = 0.5)
```



# Mesh optional arguments cutoff

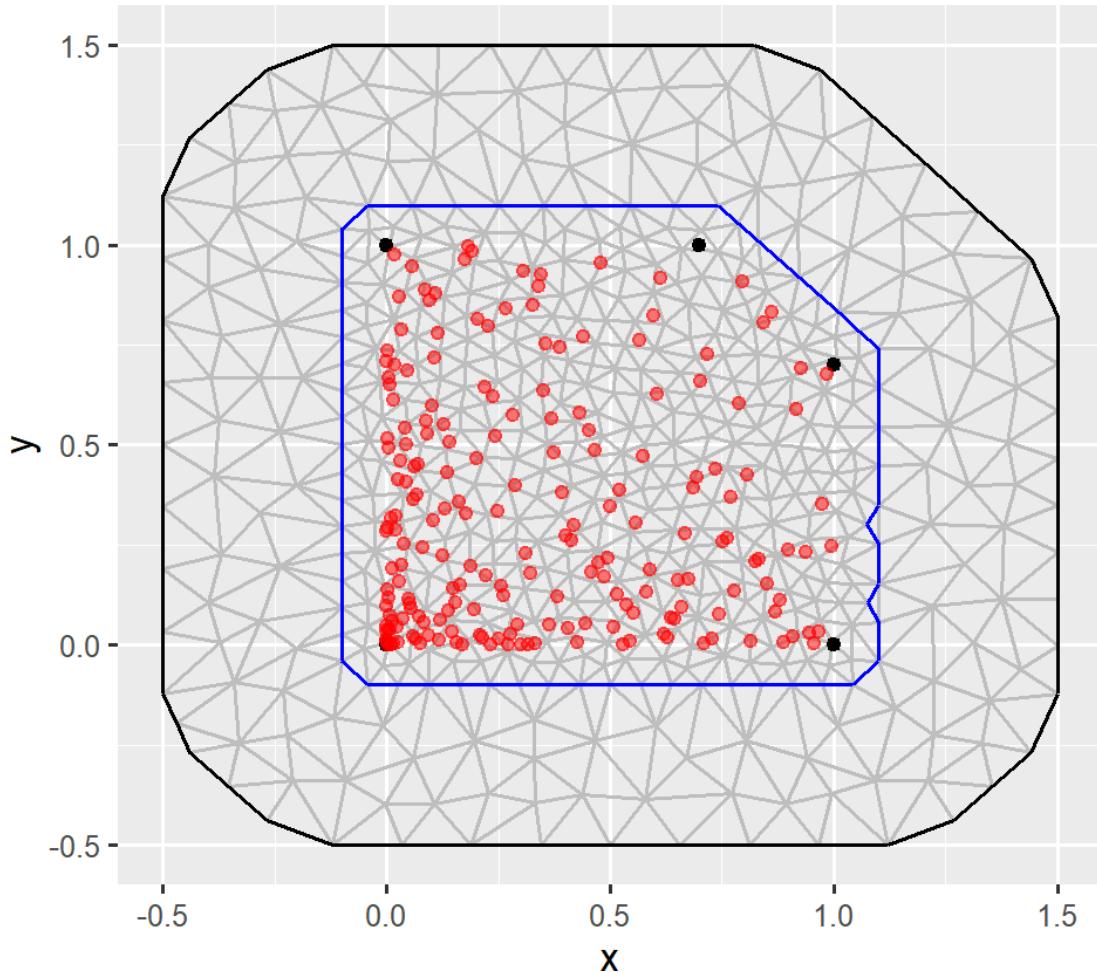
domain

mesh5

**mesh6**

```
> mesh6 = inla.mesh.2d(loc.domain = domain,
+                         max.edge = c(0.04, 0.2),
+                         cutoff = 0.05,
+                         offset = c(0.1, 0.4))
```

```
> ggplot() +
+   gg(mesh6) +
+   geom_point(data = data.frame(domain),
+              aes(X1, X2)) +
+   geom_point(data = data.frame(coords),
+              aes(s1, s2), col = "red",
+              alpha = 0.5)
```

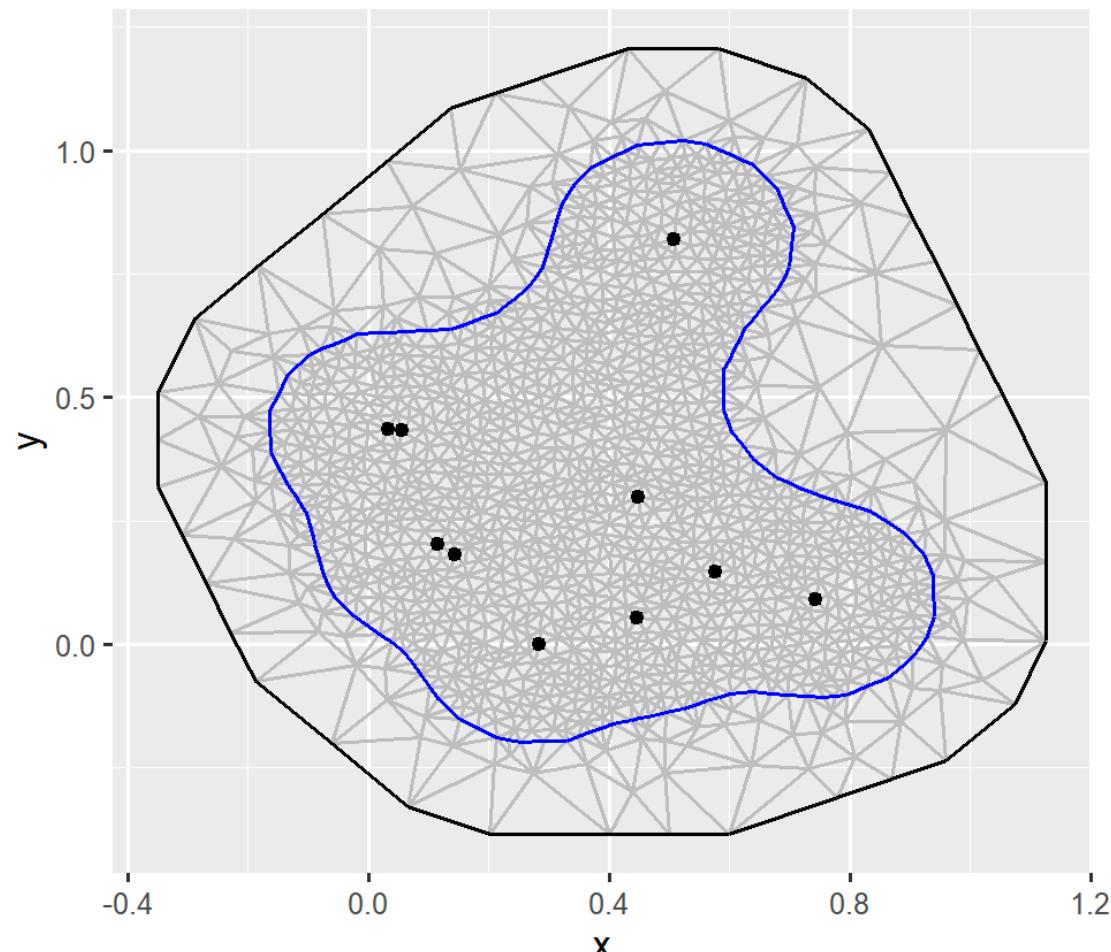


# Mesh: non convex hull

A feature in R-INLA named `inla.nonconvex.hull` makes it possible to compute a non convex hull to be included as boundary in the mesh construction. This can be particularly useful when the shape of the domain is of some importance.

```
> set.seed(44)
> loc = matrix(runif(20), 10, 2)
>
> boundary = inla.nonconvex.hull(loc,
+                                 convex=0.2)
> meshNC = inla.mesh.2d(loc = loc,
+                        boundary = boundary,
+                        max.edge = c(0.04, 0.2))
```

```
> ggplot() +
+   gg(meshNC) +
+   geom_point(data = data.frame(loc),
+              aes(X1, X2))
```



# Projector matrix for mesh6

- Now, we create a **projection matrix**, which maps the GMRF from the mesh nodes to the  $n$  observation locations
- The projector matrix is build using the `inla.spde.make.A()` function

```
> A.est6 = inla.spde.make.A(mesh = mesh6,  
+                           loc = coords)  
> dim(A.est6)
```

[1] 200 549

The matrix has the number of rows equal to the number of observations and the number of columns equal to the number of vertices of the triangulation. The points are not necessarily on the vertices and thus each point will be associated with up to three non-zero weights (that collectively add up to 1)

```
> #No more than 3 elements in each line are non-  
> table(rowSums(A.est6>0))
```



The sum of each row is one:

```
> table(rowSums(A.est6))
```

1  
200

There are some columns whose sum is zero corresponding to triangles with no point location inside:

```
> table(colSums(A.est6) > 0)
```

FALSE TRUE  
313 236

# Creation of the Matérn SPDE model for mesh6

- Now, we create the Matérn **SPDE model object** with the function `inla.spde2.matern`. By default the smoothness term is 2
- Here we create the SPDE object using the `mesh6` triangulation:

```
> spde = inla.spde2.matern(mesh = mesh6)
> spde$n.spde # number of vertices in the mesh
```

[1] 549

# PC prior for the Matérn model

- Instead of `inla.spde2.matern` it is possible to use `inla.spde2.pcmatern` for creating an `inla.spde2` model object using a PC prior for the range  $r$  and the marginal standard deviation  $\sigma$  (Simpson, Rue, Riebler, Martins, and Sorbye, 2017).
- The prior for  $\sigma$  is such that

$$Pr(\sigma > \sigma_0) = p$$

and this requires to specify  $\sigma_0$  and  $p$  with the option `prior.sigma = c(sigma0, p)`.

- The prior for  $r$  is such that

$$Pr(r < r_0) = p$$

and this requires to specify  $r_0$  and  $p$  with the option `prior.range = c(r0, p)`.

For example:

```
> spde = inla.spde2.pcmatern(mesh,
+                               prior.range = c(0.01, 0.1),
+                               prior.sigma = c(100, 0.1))
```

# Creation of the indexes for the SPDE model

- We also create a list of named index vectors for the SPDE model with the function `inla.spde.make.index`. It will be useful in the stack preparation (see next slides)

```
> s.index = inla.spde.make.index(name = "spatial.field", n.spde = spde$n.spde)
```

Here the name represents the name of the effect which will be used in the formula

- The index set for the latent field does not depend on the data set locations. It only depends on the SPDE model size

# The `inla.stack()` function

- A function named `inla.stack()` has been introduced in R-INLA for an optimal and easy management of the SPDE objects (data, covariates, indices and projector matrices) and for the construction of the linear predictor (Lindgren and Rue, 2015).
- In the `SPDEtoy` example the **linear predictor** is given by

$$\eta_i = b_0 + \xi_i = b_0 + \sum_{g=1}^G A_{ig} \tilde{\xi}_g$$

and can be written as

$$\boldsymbol{\eta} = \mathbf{1}b_0 + \mathbf{A}\tilde{\boldsymbol{\xi}}$$

where the first term refers to the intercept and the second to the spatial effect. Note that each term in the linear predictor is represented as the product of a projector matrix and an effect.

- The main arguments of the `inla.stack()` function are:
  - `data`: a vector list with the data
  - `A`: a list of projector matrices
  - `effects`: the list of effects
  - `tag (optional)`: a label for the data stack

# The `inla.stack()` function for model fitting with the SPDEtoy data

We define the `inla.stack` object for model fitting as follows:

```
> stack.est = inla.stack(  
+   data = list(y = SPDEtoy$y),  
+   A = list(1, A.est6),  
+   effects = list(intercept = rep(1,nrow(SPDEtoy)),  
+                 s.index),  
+   tag="est")
```

Note that the function `inla.stack()` will take care of eliminating any column in the projector matrix which is full of zeros.

# Model fitting

- The vector of parameters is defined as  $\theta = \{\tilde{\xi}, b_0\}$  with hyper-parameter vector  $\psi = (\sigma_e^2, r, \tau)$ , where  $\tau = 1/\sigma^2$  and  $r$  (depending on  $\kappa$ ) are the Matérn covariance function parameters.
- In R-INLA the default **internal representation** for the SPDE parameters is  $\log(\tau) = \theta_1$  and  $\log(\kappa) = \theta_2$ , with  $\theta_1$  and  $\theta_2$  being given two independent Normal prior distributions.
- Steps: (1) Define the formula, then (2) Run INLA

1. Define the **linear predictor** through the formula. In R-INLA the Matérn GF is part of the linear predictor and is specified in the formula environment using a proper specification for `f()`.

```
> formula = y ~ -1 + intercept + f(spatial.field, model = spde)
```

where `spatial.field` is a proper index variable (from `inla.spde.make.index` function), and `spde` is the model created previously with `inla.spde2.matern`. Note that the intercept is removed and is added manually in the linear predictor.

2. Run `inla!`! The function `inla.stack.data()` and `inla.stack.A()` are used for extracting the data and the projector matrix from the `stack.est` object:

```
> output6 = inla(formula,
+                 data = inla.stack.data(stack.est),
+                 control.predictor = list(A = inla.stack.A(stack.est), compute = TRUE))
```

Note that the projector matrix is passed to `inla` through `control.predictor`. Moreover, with the option `compute = TRUE` we ask for the computation of the marginals of the linear predictor.

# Exploring the output: fixed effects and hyperparameters

```
> output6$summary.fixed[,c("mean", "0.025quant", "0.975quant")]
```

	mean	0.025quant	0.975quant
intercept	9.504753	8.036181	10.86548

```
> output6$summary.hyperpar[,c("mean", "0.025quant", "0.975quant")]
```

	mean	0.025quant	0.975quant
Precision for the Gaussian observations	2.865273	2.038041	3.880469
Theta1 for spatial.field	-4.029797	-4.324775	-3.716293
Theta2 for spatial.field	2.085501	1.600862	2.534648

# Exploring the output: spatial parameters

If we are interested in the posterior summaries of the spatial parameters on the scale of the variance  $\sigma^2 = 1/\tau$  and range  $r$  (instead of the internal scale regarding  $\theta_1 = \log(\tau)$  and  $\theta_2 = \log(\kappa)$ ) we use

```
> output6.field = inla.spde2.result(inla = output6,
+                                     name = "spatial.field",
+                                     spde = spde)
```

The resulting list contains the following elements:

```
> names(output6.field)
```

```
[1] "summary.values"                  "marginals.values"
[3] "summary.hyperpar"                "summary.theta"
[5] "summary.log.tau"                 "summary.log.kappa"
[7] "summary.log.variance.nominal"   "summary.log.range.nominal"
[9] "marginals.theta"                 "marginals.log.tau"
[11] "marginals.log.kappa"            "marginals.log.variance.nominal"
[13] "marginals.log.range.nominal"   "marginals.tau"
[15] "marginals.kappa"                 "marginals.variance.nominal"
[17] "marginals.range.nominal"
```

# Exploring the output

The posterior mean of  $\sigma^2$  and the range  $r$  can be obtained by typing

```
> inla.emarginal(function(x) x, output6.field$marginals.variance.nominal[[1]])
```

```
[1] 4.032827
```

```
> inla.emarginal(function(x) x, output6.field$marginals.range.nominal[[1]])
```

```
[1] 0.3613597
```

Also the other standard INLA functions can be applied to the marginal posteriors:

```
> inla.zmarginal(output6.field$marginals.range.nominal[[1]])
```

Mean	0.36136
Stdev	0.0877111
Quantile	0.025 0.225099
Quantile	0.25 0.298391
Quantile	0.5 0.348569
Quantile	0.75 0.410873
Quantile	0.975 0.567593

# Spatial prediction using SPDE approach

# Spatial prediction with the `inla.stack()` approach

- In geostatistics we are interested in predicting the (latent) spatial field (i.e. the linear predictor) at new spatial locations where we do not have data.
- It is possible to perform the spatial prediction jointly with the estimation by using the `inla.stack` approach.
- Consider the response variable distribution

$$\boldsymbol{y} \sim \text{Normal}(\boldsymbol{\eta} = \mathbf{1}\boldsymbol{b}_0 + \mathbf{A}\tilde{\boldsymbol{\xi}}, \sigma_e^2 \mathbf{I})$$

we are interested in the posterior distribution of the linear predictor  $\boldsymbol{\eta}$  everywhere in space, especially where we don't have observed data and  $y=\text{NA}$ .

- With regard to spatial prediction, it is worth noting that the INLA-SPDE algorithm provides the posterior conditional distribution of  $\boldsymbol{\eta}$  for all the triangulation vertices. By using the SPDE approximation, it is then immediate to get a prediction for  $\boldsymbol{\eta}$  for any location in the triangulated domain (i.e. the posterior predictive distribution).

# Grid for spatial prediction

Consider the following regular grid of points:

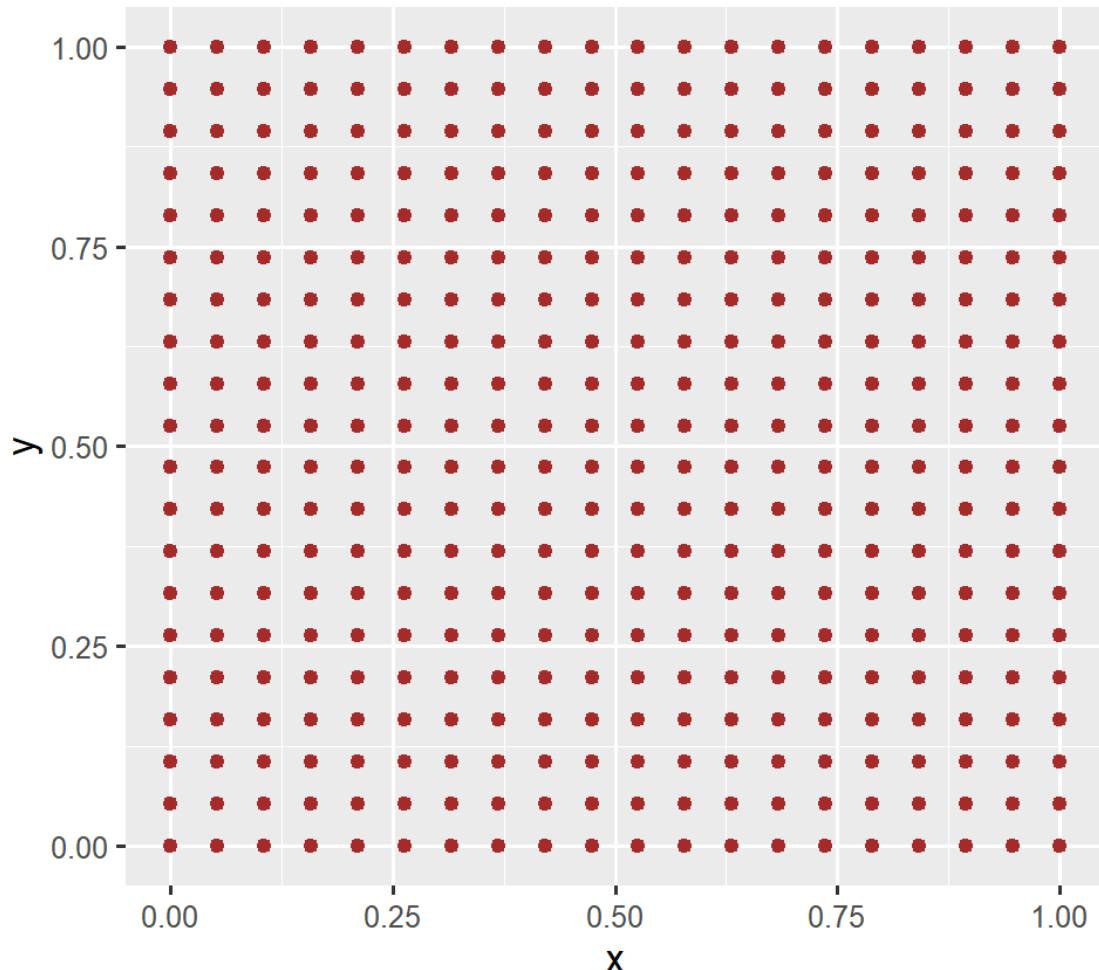
```
> grid.x = 20  
> grid.y = 20  
> pred.grid = expand.grid(  
+     x = seq(0, 1, length.out = grid.x),  
+     y = seq(0, 1, length.out = grid.y))
```



It is necessary to define a new projector matrix:

```
> A.pred6 = inla.spde.make.A(mesh = mesh6,  
+     loc = as.matrix(pred.grid))  
> dim(A.pred6)
```

```
[1] 400 549
```



# Model fitting jointly with spatial prediction

For performing **jointly** the estimation and the prediction, we create a new `inla.stack` object:

```
> stack.pred = inla.stack(data = list(y = NA),
+                           A = list(1, A.pred6),
+                           effects = list(intercept = rep(1, nrow(pred.grid)),
+                                         spatial.field = 1:spde$n.spde),
+                           tag = "pred")
```

and then we join it to the `inla.stack` object created previously for the estimation (`stack.est`):

```
> join.stack = inla.stack(stack.est, stack.pred) #full stack object
```

And finally, we run INLA again:

```
> output6pred = inla(formula,
+                      data = inla.stack.data(join.stack),
+                      control.compute = list(return.marginals=TRUE),
+                      control.predictor = list(A = inla.stack.A(join.stack), compute = TRUE)
+ )
```

The option `return.marginals.predictor=TRUE` is necessary to obtain the marginals for the linear predictor.

# Retrieve the predictions

To access the predictions (posterior summary stats or marginal distribution) at the target grid locations, we extract with the `inla.stack.index()` function the corresponding indexes from the full stack object using the corresponding tag set before (`pred`):

```
> index.pred = inla.stack.index(join.stack, tag = "pred")$data  
> length(index.pred)
```

```
[1] 400
```

We then extract the prediction posterior mean and sd at the first 3 grid points:

```
> output6pred$summary.linear.predictor[index.pred[1:3],c("mean", "sd")]
```

	mean	sd
APredictor.201	12.24728	0.2609542
APredictor.202	12.06214	0.4240041
APredictor.203	10.58517	0.4330569

In this case (identity link) `output6pred$summary.fitted.values` would return the same output.

# Manipulate the posterior predictive distribution

As described in the previous weeks, it is possible to manipulate marginal distributions.

Consider for example the first grid point and its posterior predictive distribution:

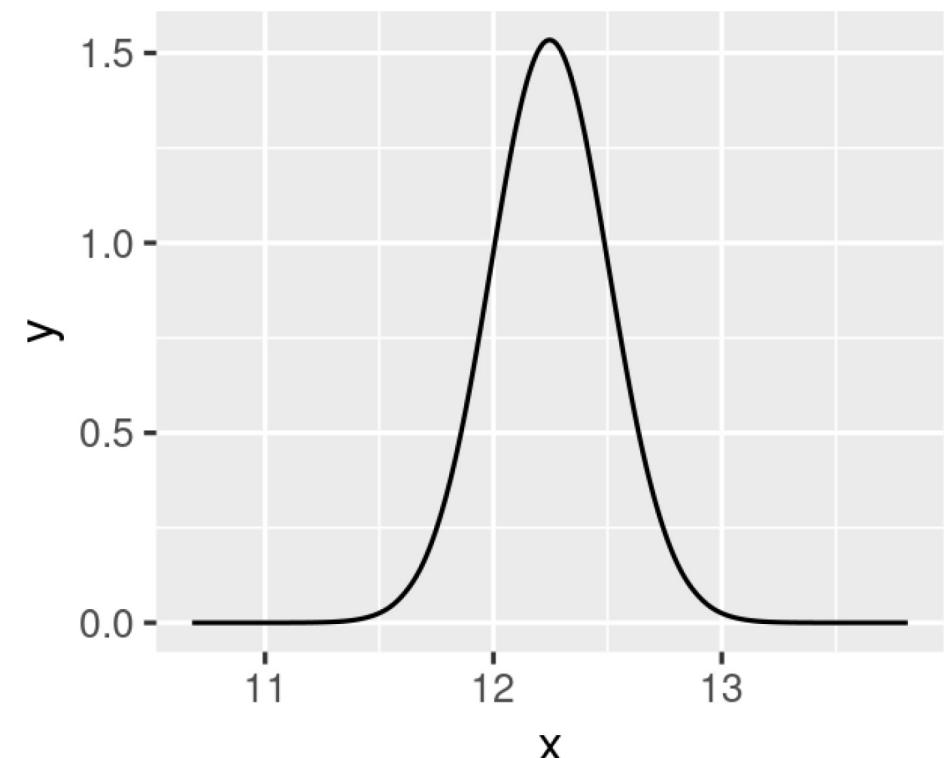
```
> distr.point1 = output6pred$ marginals.linear.pre  
> distr.point1.smooth = inla.smarginal(distr.poir
```



```
> ggplot(data.frame(distr.point1.smooth)) +  
+   geom_line(aes(x,y))
```

We can be also interested in computing the posterior probability of getting a value bigger than 13:

```
> 1 - inla.pmarginal(13, distr.point1)
```



# Mapping the linear predictor: posterior mean [1]

- We plot now the posterior mean of the linear predictor at the grid level.
- We can take advantage of `inlabru`, a R package for Bayesian spatial modelling, originally developed for ecological applications (Bachl, Lindgren, Borchers, and Illian, 2019).
- `inlabru` is a wrapper around INLA tailored towards spatial data. It makes fitting spatial models with INLA easier as there is no more need to deal with projector matrices and stack objects.
- The `inlabru` package works with spatial objects from the `sp` package (e.g. `SpatialPointsDataFrame`, `SpatialGridDataFrame`, `SpatialPixelsDataFrame`): <https://cran.r-project.org/web/packages/sp/index.html>
- The function `gg()` is an extension of the `ggplot()` function for generating geometries from spatial fitted object.

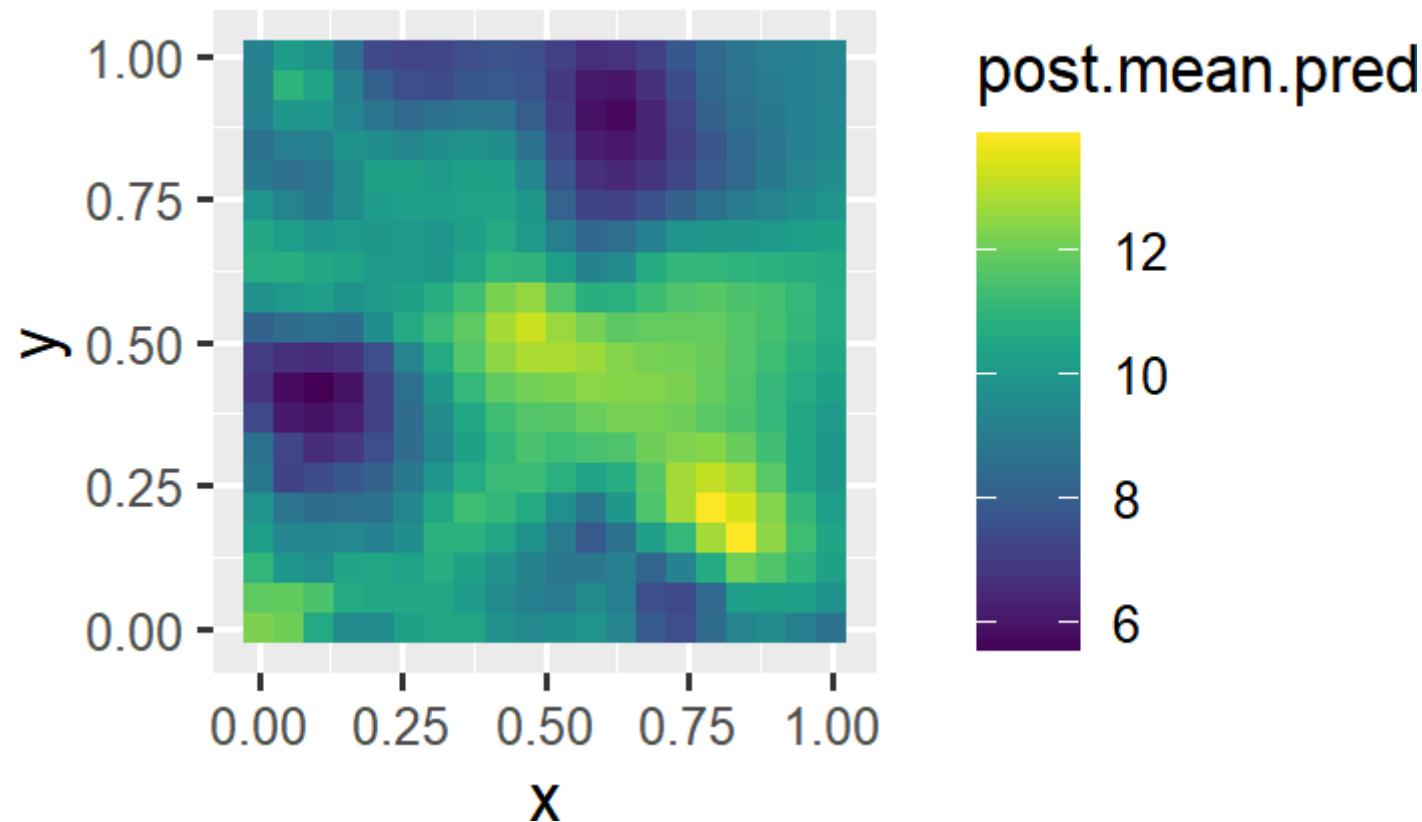
See:

- Website: <https://sites.google.com/inlabru.org/inlabru>
- Github: <https://github.com/inlabru-org/inlabru>

## Mapping the linear predictor: posterior mean [2]

```
> library(inlabru)
> post.mean.pred = output6pred$summary.linear.predictor[index.pred, "mean"]
> post.mean.df = SpatialPixelsDataFrame(SpatialPoints(pred.grid),
+                                         data = data.frame(post.mean.pred))
>
> library(viridis)
> ggplot() +
+   gg(post.mean.df, aes(x, y, post.mean.pred)) +
+   gtitle("Posterior mean") +
+   coord_fixed() + #x and y axis with the same length in the plot
+   scale_fill_viridis()
```

# Posterior mean

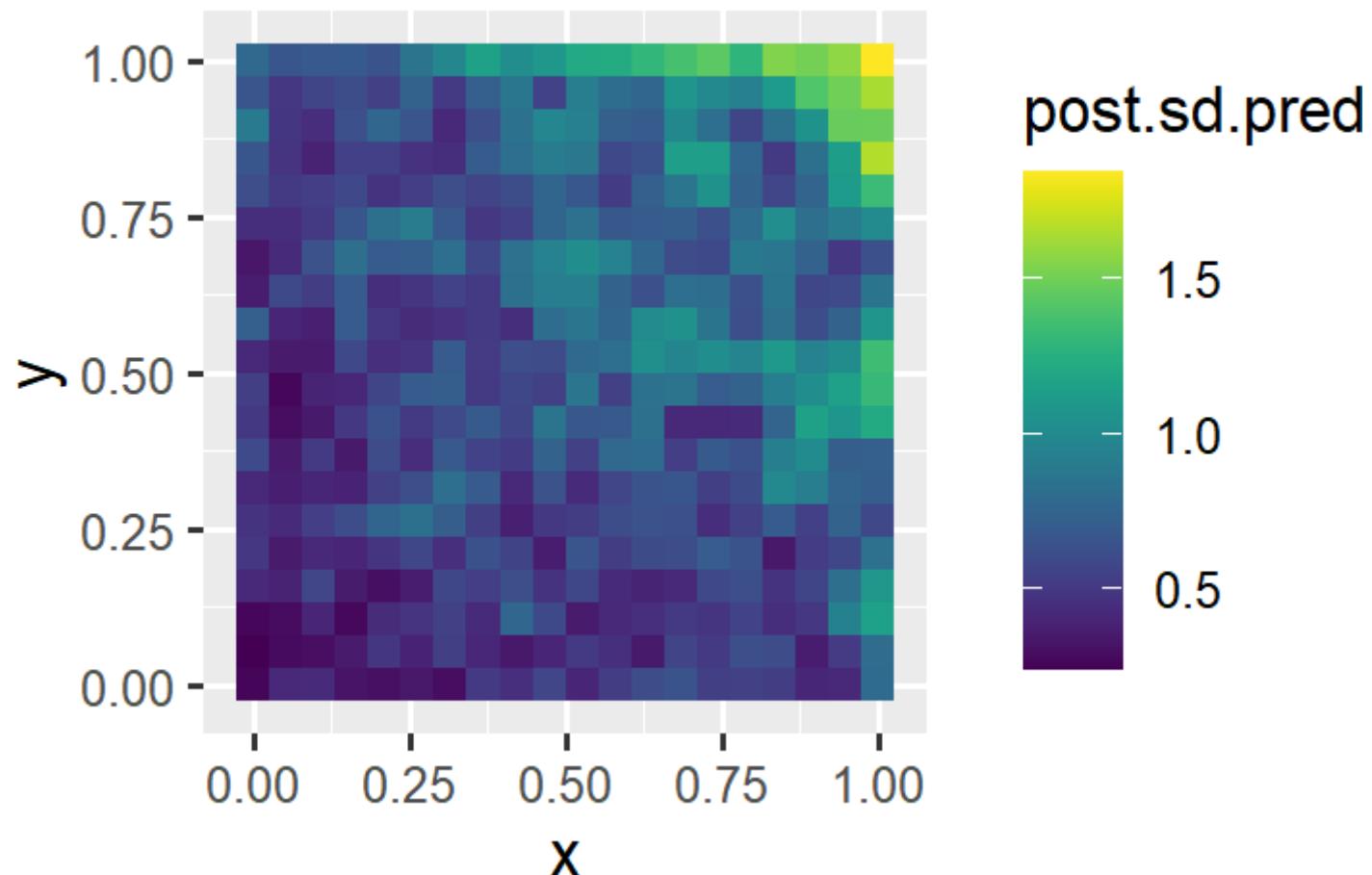


# Mapping the linear predictor: posterior standard deviation

We plot now the posterior standard deviation of the linear predictor at the grid level

```
> post.sd.pred = output6pred$summary.linear.predictor[index.pred, "sd"]
>
> post.sd.df = SpatialPixelsDataFrame(SpatialPoints(pred.grid),
+                                         data = data.frame(post.sd.pred))
> ggplot() +
+   gg(post.sd.df, aes(x, y, post.sd.pred)) +
+   ggtitle("Posterior standard deviation") +
+   coord_fixed() +
+   scale_fill_viridis()
```

# Posterior standard deviation



# Wrap-up: fitting a geostatistical model using SPDE approach

- Construct the mesh to obtain a triangulation of the domain using the function `inla.mesh.2d`
- Construct the SPDE model using the function `inla.spde2.matern` or `inla.spde2.pcmatern`
- Construct the index set using the function `inla.spde.make.index`
- Construct the projection matrix to link the observation locations to the mesh using the function `inla.spde.make.A`
- Put everything together in a stack object using the function `inla.stack`
- Run the model

## References

- Bachl, F. E., F. Lindgren, D. L. Borchers, et al. (2019). "inlabru: an R package for Bayesian spatial modelling from ecological survey data". In: *Methods in Ecology and Evolution* 10.6, pp. 760-766.
- Banerjee, S., B. P. Carlin, and A. E. Gelfand (2014). *Hierarchical modeling and analysis for spatial data*. Chapman and Hall/CRC.
- Lindgren, F. and H. Rue (2015). "Bayesian Spatial Modelling with R-INLA". In: *Journal of Statistical Software* 63.19, pp. 1-25.
- Lindgren, F., H. Rue, and J. Lindström (2011). "An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach (with discussion)". In: *J. R. Statist. Soc. B* 73.4, pp. 423-498.
- Simpson, D., H. Rue, A. Riebler, et al. (2017). "Penalising Model Component Complexity: A Principled, Practical Approach to Constructing Priors". In: *Statistical Science* 32.1, pp. 1-28.