An introduction to MetricGraph

Statistical modeling

David Bolin and Alexandre Simas

Glasgow 2025

Whittle-Matern fields on metric graphs

We define the Whittle–Matérn fields through the fractional-order equation

$$(\kappa^2 - \Delta_\Gamma)^{\alpha/2}(\tau u) = \mathcal{W} \quad \text{on } \Gamma,$$

- $\mathcal{W}$ is Gaussian white noise,
- $\kappa > 0, \tau > 0, \alpha = \nu + 1/2 > 1/2$,
- Δ_Γ is the Kirchhoff Laplacian, which acts as the second derivative on the edges and the following vertex conditions:

$$\{u \text{ is continuous on } \Gamma \text{ and } \forall v \in \mathcal{V} : \textstyle\sum_{e \in \mathcal{E}_v} \partial_e u(v) = 0\}.$$

Properties

- α controls the sample path regularity, and $\alpha > 3/2$ results in differentiable sample paths.
- These fields are in general not isotropic.
- The fields are invariant to addition or removal of vertices of degree 2.
- If $\alpha \in \mathbb{N}$, these are Markov random fields and we can do exact and computationally efficient inference without FEM.

Meshes on graphs

To illustrate these fields, consider the following graph

```
edge1 <- rbind(c(0,0),c(1,0))
edge2 <- rbind(c(0,0),c(0,1))
edge3 <- rbind(c(0,1),c(-1,1))
theta <- seq(from=pi,to=3*pi/2,length.out = 20)
edge4 <- cbind(sin(theta),1+ cos(theta))
edges = list(edge1, edge2, edge3, edge4)
graph <- metric_graph$new(edges = edges)
```

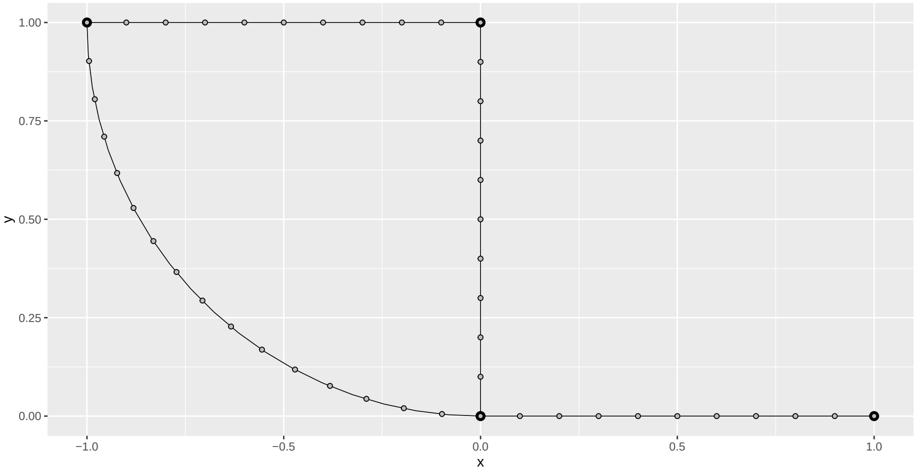
`metric_graph` objects has built-in functionality for generating meshes.

The primary usage of meshes is to plot functions on metric graphs.

We can either specify the mesh width `h` (the maximal distance between nodes) or the number of mesh nodes per edge.

Meshes on graphs

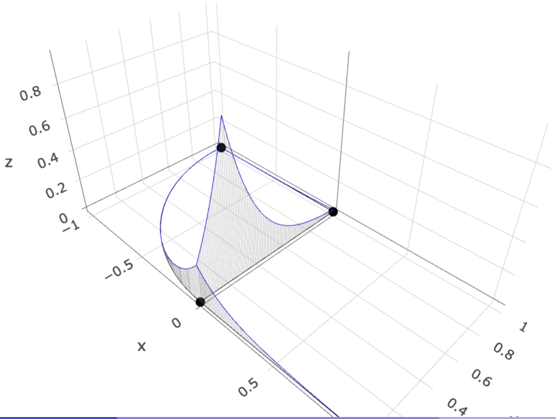
```r
graph$build_mesh(h = 0.1)
graph$plot(mesh=TRUE)
```



The `plot_function()` method

Let us plot a covariance of a Whittle–Mat'ern field on a finer mesh:

```
graph$build_mesh(h = 0.01)
C <- spde_covariance(c(2, 0.2), range = 0.4, sigma = 1,
                     alpha = 1, graph = graph)
graph$plot_function(X = C, type = "plotly")
```

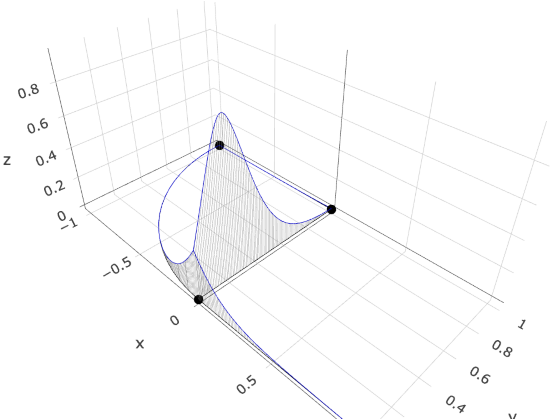


Comments

- By default `spde_covariance` computes the covariance function $r(s_0, s)$ for a specified location s_0 and all location on the mesh.
- The mesh is only used for plotting, there is no FEM approximation
- `range` is the practical correlation range, $\sqrt{8\alpha - 4}/\kappa$
- `sigma` is $\sqrt{\Gamma(\alpha - 0.5)/(\Gamma(\alpha)\sqrt{4\pi}\kappa^{2\alpha-1})}$
- If `X` has the same size as the mesh, `plot_function()` assumes that `X` is the function we want to plot evaluated at the mesh.

Example with $\alpha = 2$

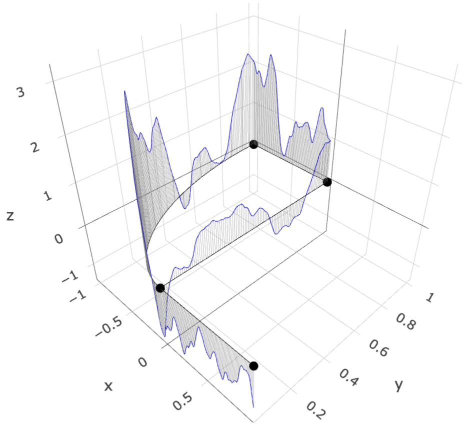
```
C <- spde_covariance(c(2, 0.2), range = 0.4, sigma = 1,
                     alpha = 2, graph = graph)
graph$plot_function(X = C, type = "plotly")
```



Simulation

`sample_spde` can be used to simulate the Whittle–Matérn fields.

```
u <- sample_spde(range = 0.4, sigma = 1, alpha = 2,
                 graph = graph, type = "mesh")
graph$plot_function(X = u, type = "plotly")
```



General smoothness and FEM

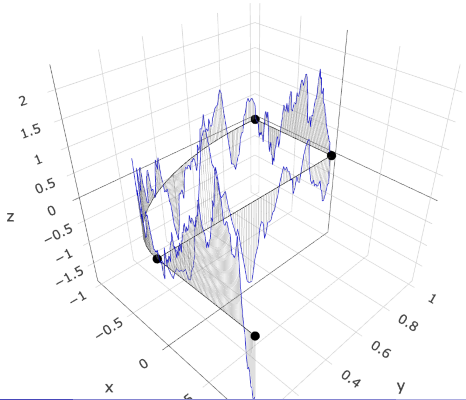
- For models with general α, we need to use FEM and rational approximations.
- `rSPDE` has a complete replacement for `INLA` SPDE models, allowing for general α and estimation of α:

$$\texttt{inla.spde2.matern} \longrightarrow \texttt{rspde.matern}$$

- Available on CRAN. Homepage: https://davidbolin.github.io/rSPDE/
- `rSPDE` also has full support for metric graph models.

simulation with rSPDE

```r
library(rSPDE)
op <- matern.operators(nu = 0.8, range = 0.4, sigma = 1,
                       parameterization = "matern",
                       m = 2, graph = graph)
u <- simulate(op)
graph$plot_function(X = u, type = "plotly")
```



Inference with MetricGraph

Three ways of fitting statistical models on metric graphs:

1. Likelihood-based inference through `graph_lme()`.
2. Bayesian inference through the `INLA` interface.
3. Bayesian inference through the `inlabru` interface.

Overview of `graph_lme()`

`graph_lme()` allows fitting mixed effect models

$$y_{ij} = \sum_{k=1}^{K} x_{kij}\beta_k + u_j(s_{ij}) + \varepsilon_{ij}$$

where $\mathbf{x}_k$ are fixed effects, $u_j(s)$ are independent copies of a Gaussian process and ε_{ij} Gaussian measurement noise.

```
fit <- graph_lme(y ~ x1 + x2, graph = graph, model = "WM")
```

- Data and replicates are extracted from the graph, where `group` is used for replicates.
- Confidence intervals are computed based on numerical approximation of the Hessian.
- Once the model has been fitted, prediction at new locations can be obtained through `predict()` or `augment()`.

Model implemented in `graph_lme()`

- 'lm' for linear models without random effects.
- 'WM1' and 'WM2' for exact Whittle-Matérn fields with $\alpha = 1, 2$.
- 'WM' Whittle–Matérn fields with general smoothness (α estimated).
- 'isoExp' for a model with isotropic exponential covariance.
- 'GL1' and 'GL2' for a SPDE model based on the graph Laplacian with $\alpha = 1, 2$.
- 'WMD1' is the directional Whittle-Matérn field with $\alpha = 1$.
- `list("WhittleMatern", fem = TRUE, B.kappa, B.tau)` for generalized Whittle–Matérn fields with log-linear models for κ and τ and general smoothness which is estimated.
- `list("isoCov", cov_function())` models with a general isotropic covariance function.

Models fo `INLA` and `inlabru`

Exact Whittle–Matérn models with integer α:

```
spde <- graph_spde(graph)
```

Models fo `INLA` and `inlabru`

Exact Whittle–Matérn models with integer α:

```
spde <- graph_spde(graph)
```

By `directional = TRUE` we obtain a directional model

$$(\kappa + \nabla)^{\alpha}(\tau u) = \mathcal{W}.$$

Models fo `INLA` and `inlabru`

Exact Whittle–Matérn models with integer α:

```
spde <- graph_spde(graph)
```

By `directional = TRUE` we obtain a directional model

$$(\kappa + \nabla)^{\alpha}(\tau u) = \mathcal{W}.$$

Generalized Whittle-Matérn fields:

```
rspde <- rspde.metric_graph(graph)
```

Models fo `INLA` and `inlabru`

Exact Whittle–Matérn models with integer α:

```r
spde <- graph_spde(graph)
```

By `directional = TRUE` we obtain a directional model

$$(\kappa + \nabla)^{\alpha}(\tau u) = \mathcal{W}.$$

Generalized Whittle-Matérn fields:

```r
rspde <- rspde.metric_graph(graph)
```

Spatio-temporal models

$$du + \gamma(\kappa^2 + \rho\nabla - \Delta)^{\alpha} u = dW_Q, \quad \text{on } T \times \Gamma$$

```r
st_model <- rspde.spacetime(mesh_space = graph,
                            mesh_time = mesh_time,
                            alpha = 1, beta = 1)
```

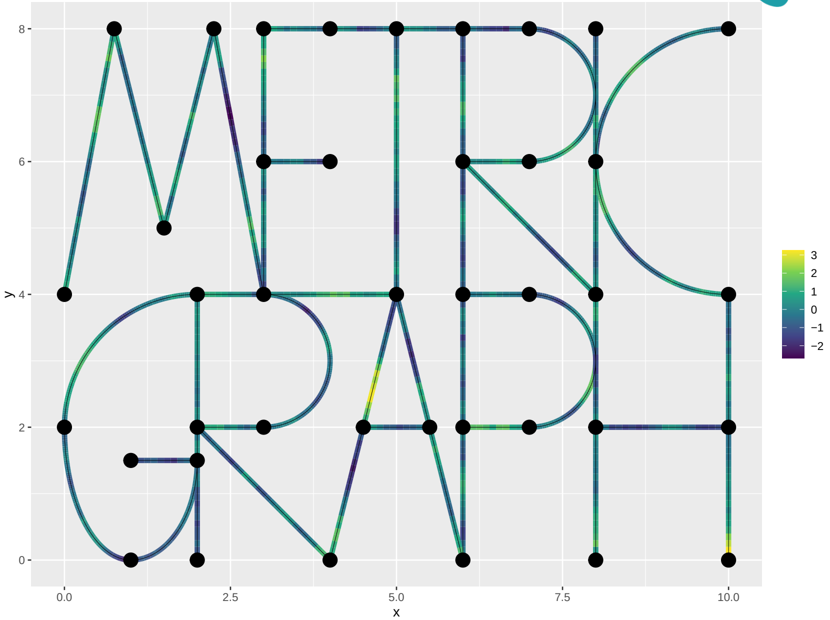
All models can be used as any other random effects in `INLA` and `inlabru`.

Example data

Let us simulate some data on a graph.

```
graph <- metric_graph$new()

graph$build_mesh(h = 0.1)

op <- matern.operators(nu = 0.8,
                       range = 1,
                       sigma = 1,
                       parameterization = "matern",
                       graph = graph)

rep <- 20
u <- simulate(op, nsim = rep)

graph$plot_function(X = u[,10], edge_width = 2)
```

Example data



Example data

We now generate 10 random observation locations per edge and construct the corresponding observation matrix.

```
loc <- NULL
for(i in 1:graph$nE) {
  loc <- rbind(loc, cbind(rep(i,10), runif(10)))
}
A <- graph$fem_basis(loc)
```

Then we simulate data

$$Y_{ij} = u_i(s_j) + \varepsilon_{ij},$$

where ε_{ij} are iid $N(0, 0.1^2)$.

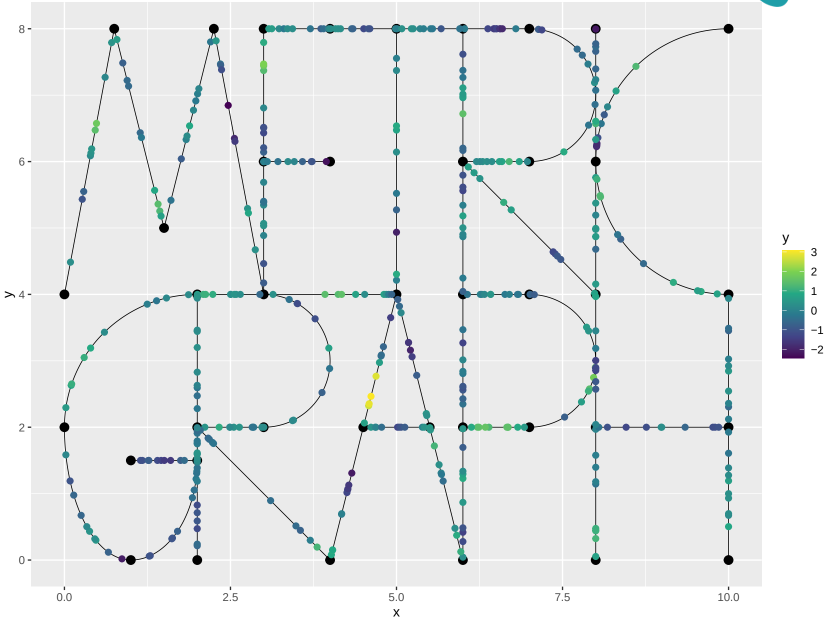
```
Y <- A%*%u + 0.1*matrix(rnorm(10 * graph$nE * rep),
                        ncol = rep)
```

Adding the data to the graph

Then create a dataframe and add the data to the graph.

```
df <- data.frame(y=as.vector(Y),
                 edge_number = rep(loc[,1], rep),
                 distance_on_edge = rep(loc[,2], rep),
                 repl = rep(1:rep, each = 10 * graph$nE))

graph$add_observations(data = df, group = "repl",
                       normalized = TRUE)

graph$plot(data = "y", group = 10, data_size = 2)
```

Adding the data to the graph



Fitting a model with INLA

Step 1: Create the model

```
rspde <- rspde.metric_graph(graph)
```

Fitting a model with INLA

Step 1: Create the model

```
rspde <- rspde.metric_graph(graph)
```

Step 2: Extract the required data and give the model a name, using `graph_data_spde` for exact models, or `graph_data_rspde` for `rSPDE` models:

```
data_spde <- graph_data_rspde(rspde, name = "field",
                              repl = ".all",
                              repl_col = "repl")
```

Fitting a model with INLA

Step 1: Create the model

```
rspde <- rspde.metric_graph(graph)
```

Step 2: Extract the required data and give the model a name, using graph_data_spde for exact models, or graph_data_rspde for rSPDE models:

```
data_spde <- graph_data_rspde(rspde, name = "field",
                              repl = ".all",
                              repl_col = "repl")
```

Step 3: Create the stack

```
stk <- inla.stack(data = data_spde[["data"]]["y"],
                  A = data_spde[["basis"]],
                  effects = c(data_spde[["index"]],
                              list(Intercept = 1)))
```

Fitting a model with INLA

Step 4: Define the formula and fit the model

```
f.s <- y ~ -1 + Intercept + f(field, model = rspde,
                              replicate = field.repl)
fit <- inla(f.s, data = inla.stack.data(stk),
            control.predictor=list(A=inla.stack.A(stk)))
```

Fitting a model with INLA

Step 4: Define the formula and fit the model

```
f.s <- y ~ -1 + Intercept + f(field, model = rspde,
                              replicate = field.repl)
fit <- inla(f.s, data = inla.stack.data(stk),
            control.predictor=list(A=inla.stack.A(stk)))
```

Step 5: Extract results using `spde_metric_graph_result()` for exact models and `rspde.results()` for rSPDE models.

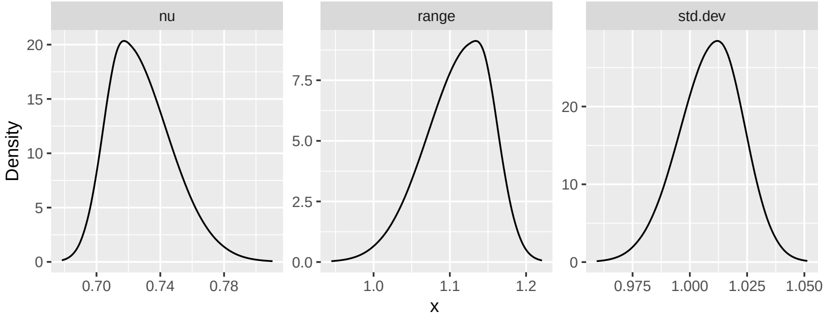
```
result <- rspde.result(fit, "field", rspde)

summary(result, digits = 3)
```

```
##          mean     sd 0.025quant 0.5quant 0.975quant  mode
## std.dev 1.010 0.0140      0.980    1.010      1.030 1.010
## range   1.110 0.0438      1.010    1.110      1.180 1.130
## nu      0.728 0.0204      0.695    0.726      0.774 0.717
```

Plotting posterior densities

We can use the gg_df function to extract a dataframe for plotting.

```
df_plot <- gg_df(result)
ggplot(df_plot) + geom_line(aes(x = x, y = y)) +
  facet_wrap(~parameter, scales = "free") +
  labs(y = "Density")
```



Overview of the inlabru interface

The inlabru interface is similar. We do not need to define the stack.

Let us fit an exact model without FEM to the data:

```
graph$clear_observations()
graph$add_observations(data = df,  group = "repl",
                       normalized = TRUE)
spde <- graph_spde(graph)

data_spde <- graph_data_spde(spde, repl = ".all",
                             repl_col = "repl",
                             loc_name = "loc")
repl <- data_spde[["repl"]]
f.s <- y ~ -1 + Intercept(1) + field(loc,
                                     model = spde,
                                     replicate = repl)
fit <- bru(f.s, data=data_spde[["data"]])
result <- spde_metric_graph_result(fit, "field", spde)
```

Prediction

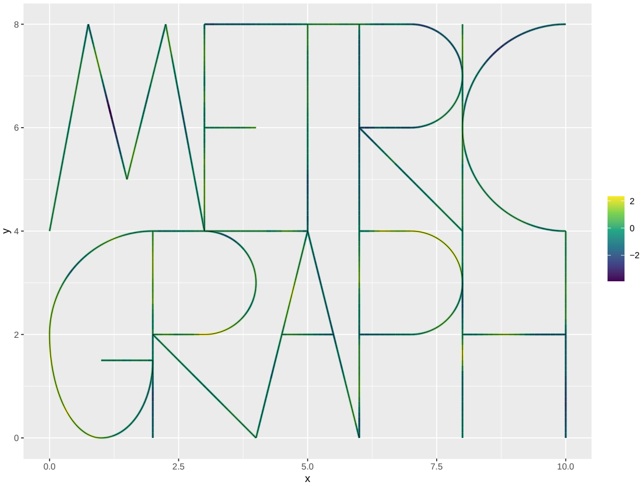
We can use predict to do prediction, note that for exact models, this is not the `inlabru` predict method.

```
data_list <- graph$get_mesh_locations(bru = TRUE,
                                      loc_name = "loc")
data_list$repl <- rep(1, dim(data_list$loc)[1])
y_pred <- predict(spde, f.s, fit,
                  newdata=data_list,
                  formula = ~ Intercept + field,
                  repl = 1, repl_col = "repl")
```

Plot of prediction

Let us process the predictions and plot them

```
plot(y_pred)
```



Log-Gaussian Cox Processes (LGCP)

We will consider LGCPs on metric graphs driven by a Gaussian Whittle–Matérn fields. Therefore, the LGCPs we will consider are point process models with intensity $\lambda = \exp(\beta + u)$

- β is an intercept or linear predictor
- u is a Gaussian Whittle–Matérn field: $(\kappa^2 - \Delta)^{\alpha/2}\tau u = \mathcal{W}$
- Used to model spatially correlated point patterns
- `MetricGraph` provides interfaces for:
  - Simulation via `graph_lgcp_sim()`
  - Parameter estimation through `INLA` via `lgcp_graph()`

LGCP: Simulating Data

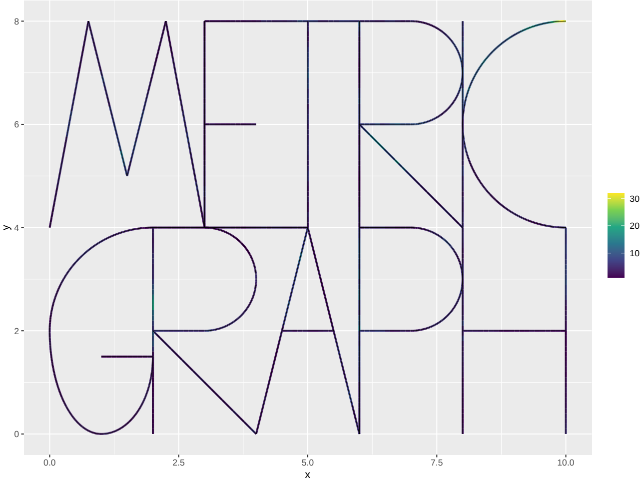
```r
graph <- metric_graph$new(remove_deg2 = TRUE)
graph$build_mesh(h = 0.1)
graph$compute_fem()
```

Let us simulate a LGCP with covariates.

```r
cov_lgcp <- graph$mesh$VtE[,1]/max(graph$mesh$VtE[,1])
lgcp_sample <- graph_lgcp_sim(intercept = 1 + cov_lgcp,
                              sigma = 0.5, range = 2,
                              alpha = 1,
                              graph = graph)
```

LGCP: Simulated Intensity

```
graph$plot_function(X = exp(lgcp_sample$u), vertex_size = 0)
```

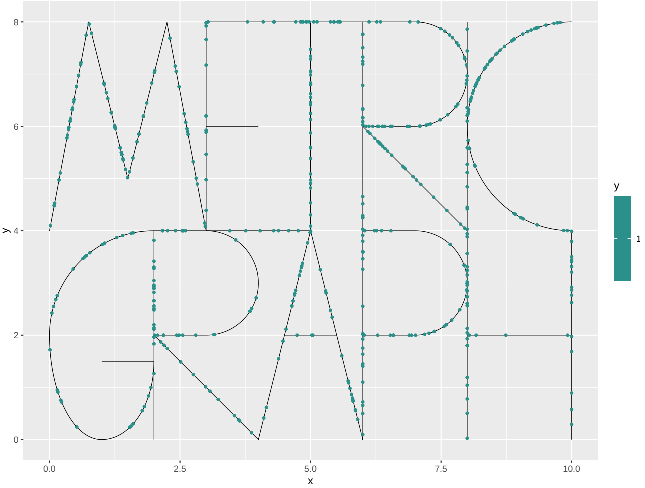


LGCP: Simulated Point Pattern

Let us add the point pattern to the graph.

```
graph$add_observations(
  data = data.frame(
    y = rep(1, length(lgcp_sample$edge_loc)),
    edge_number = lgcp_sample$edge_number,
    distance_on_edge = lgcp_sample$edge_loc,
    Intercept = 1,
    cov_lgcp =
    lgcp_sample$edge_number/max(lgcp_sample$edge_number)
  ),
  normalized = TRUE
)
```

LGCP: Simulated Point Pattern



LGCP: Estimation Interface

- Key components for LGCP inference:
  - Integration points (mesh vertices by default)
  - Covariates (interpolated from data or manually specified)
  - SPDE model (exact or FEM-based via `rSPDE`)

LGCP: Model Fitting with rSPDE

```
rspde_model <- rspde.metric_graph(graph, nu = 0.5)

inla_fit <- lgcp_graph(
  y ~ -1 + Intercept + cov_lgcp +
  f(field, model = rspde_model),
  graph = graph
)
```

Let us extract the results in original scale.

```
spde_result <- spde_metric_graph_result(inla_fit,
      "field", rspde_model)
```

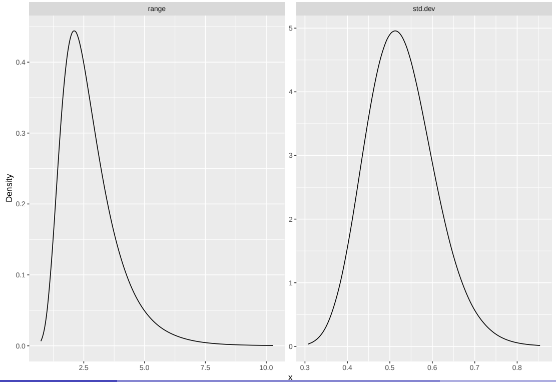
Let us compare the estimated parameters with the true values.

```
##   parameter true      mean
## 1   std.dev  0.5 0.5261923
## 2     range  2.0 2.7675832
```

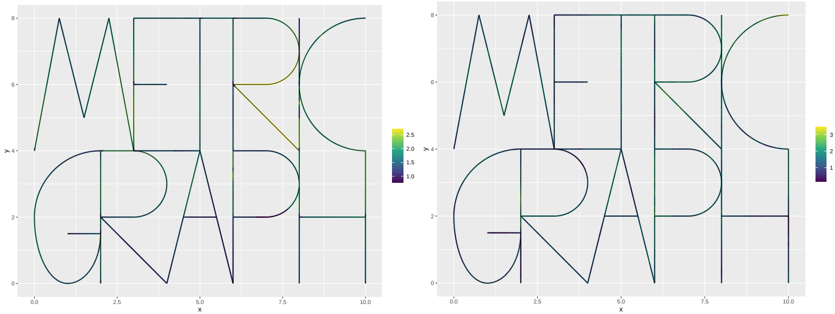
LGCP: Parameter Posteriors

We can also plot the posterior densities.

```
posterior_df_fit <- gg_df(spde_result)
ggplot(posterior_df_fit) +
  geom_line(aes(x = x, y = y)) +
  facet_wrap(~parameter, scales = "free") +
  labs(y = "Density")
```



LGCP: Estimated vs. True Field



Estimated field (left) vs. True field (right)

LGCP: Exact Model

We can also use exact model with `graph_spde()`:

```
spde_model <- graph_spde(graph, alpha = 1)
```

Let us fit the LGCP model with exact SPDE.

```
inla_fit_spde <- lgcp_graph(
  y ~ -1 + Intercept + cov_lgcp +
  f(field, model = spde_model),
  graph = graph, verbose = TRUE
)
```

LGCP: Exact Model (2)

Let us extract the results in original scale.

```
spde_result <- spde_metric_graph_result(inla_fit_spde,
      "field", spde_model)
```

```
##    parameter true      mean
## 1    std.dev  0.5 0.8727829
## 2      range  2.0 1.4611006
```