Spatio-Temporal Hawkes

Francesco Serafini

28/07/2021

Target Model

Hawkes process model with conditional intensity at time t and location $\mathbf{s}$ given the history of the process at time t, $\mathcal{H}_t$, is given by:

$$\lambda(t, \mathbf{s}|\mathcal{H}_t) = \mu(\mathbf{s}) + K \sum_{i:t_i<t} g_m(m_i) g_t(t - t_i) g_s(\mathbf{s} - \mathbf{s}_i)$$

with expected number of point in a region $A \times [T_1, T_2]$

$$\Lambda(T_1, T_2, A) = \int_A \int_{T_1}^{T_2} \lambda(t, \mathbf{s}|\mathcal{H}_t) dt d\mathbf{s}$$

$$= (T_2 - T_1) \int_A \mu(\mathbf{s}) d\mathbf{s} + K \sum_i g_m(m_i) \int_A \int_{max(t_i, T_1)}^{T_2} g_t(t - t_i) g_s(|\mathbf{s} - \mathbf{s}_i|) dt d\mathbf{s}$$

$$= (T_2 - T_1) \int_A \mu(\mathbf{s}) d\mathbf{s} + K \sum_i g_m(m_i) I_t(t_i, T_1, T_2) I_s(\mathbf{s}_i, A)$$

where the summation is over $i : t_i \in H_{T_2}$ the history of the process up to time T_2 and

$$I_t(t_i, T_1, T_2) = \int_{max(t_i, T_1)}^{T_2} g_t(t - t_i) dt$$

$$I_s(s_i, A) = \int_A g_s(\mathbf{s} - \mathbf{s}_i) d\mathbf{s}$$

The log-likelihood is given by

$$\mathcal{L} = -\left((T_2 - T_1) \int_A \mu(\mathbf{s}) d\mathbf{s} + K \sum_i g_m(m_i) I_t(t_i, T_1, T_2) I_s(\mathbf{s}_i, A)\right) + \sum_i \log\left(\mu(\mathbf{s}_i) + K \sum_{j:t_j<t_i} g_m(m_j) g_t(t_i - t_j) g_s(|\mathbf{s}_i - \mathbf{s}_j|)\right)$$

In our case

$$\mu(\mathbf{s}) = \exp(\theta_1) f(\mathbf{s})$$

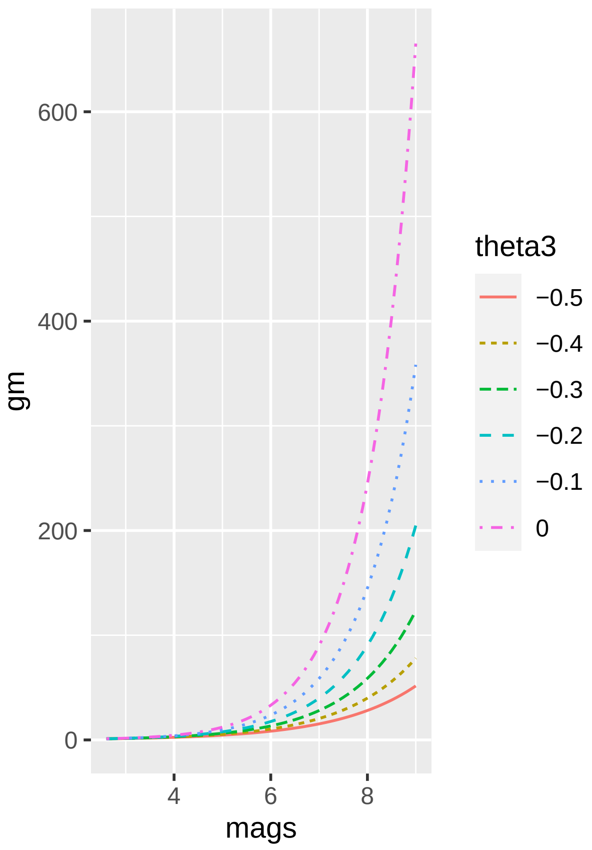
We assume $f(\mathbf{s})$ is given or estimated independently from others parameters.

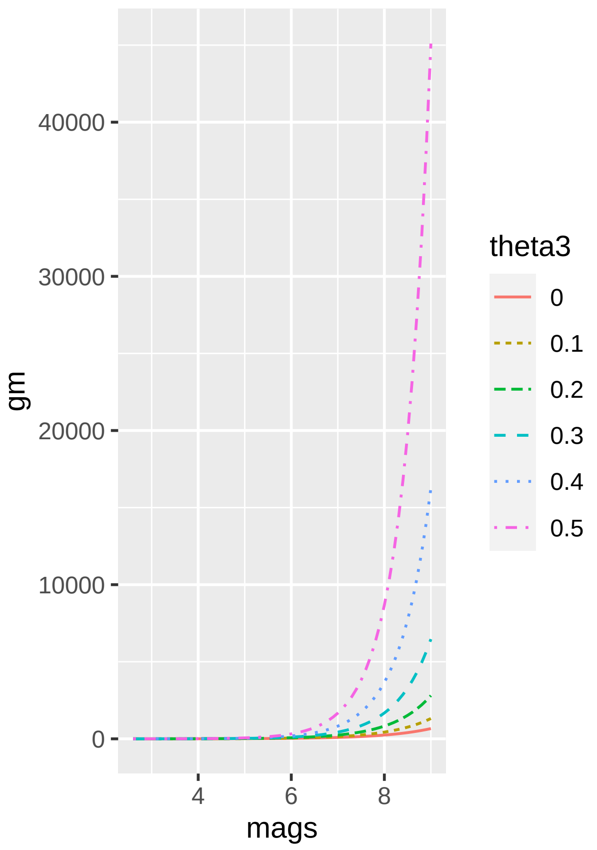
$$K = \exp(\theta_2)$$

Considering a magnitude of completeness M_0, we consider a specific form for each triggering function. The first, $g_m(\cdot)$ represents the influence of the magnitude of past events on the present. We suppose that events with higher magnitude have a stronger effect and, therefore, we consider $\exp\theta_3$ which is a non-negative quantity.

$$g_m(m) = \exp\{\exp(\theta_3)(m - M_0)\}$$

```r
toplot.gm <- function(theta3.seq, mm, M0){
  gm.list <- lapply(theta3.seq, function(x) data.frame(mags = mm,
                                                        gm = sapply(mm, function(m) g.m(m, x, M0)),
                                                        theta3 = x))
  gm.df <- bind_rows(gm.list)

  ggplot(gm.df, aes(x = mags, y = gm, color = as.factor(theta3),
                    linetype = as.factor(theta3))) +
           geom_line() +
    labs(color = "theta3", linetype = 'theta3')

}

M0 <- 2.5
mm <- seq(M0 + 0.1, 9, by = 0.1)

theta3.seq <- seq(-0.5, 0, by = 0.1)
pl.left <- toplot.gm(theta3.seq, mm, M0)

theta3.seq <- seq(0, 0.5, by = 0.1)
pl.right <- toplot.gm(theta3.seq, mm, M0)


multiplot(pl.left, pl.right, cols = 2)
```





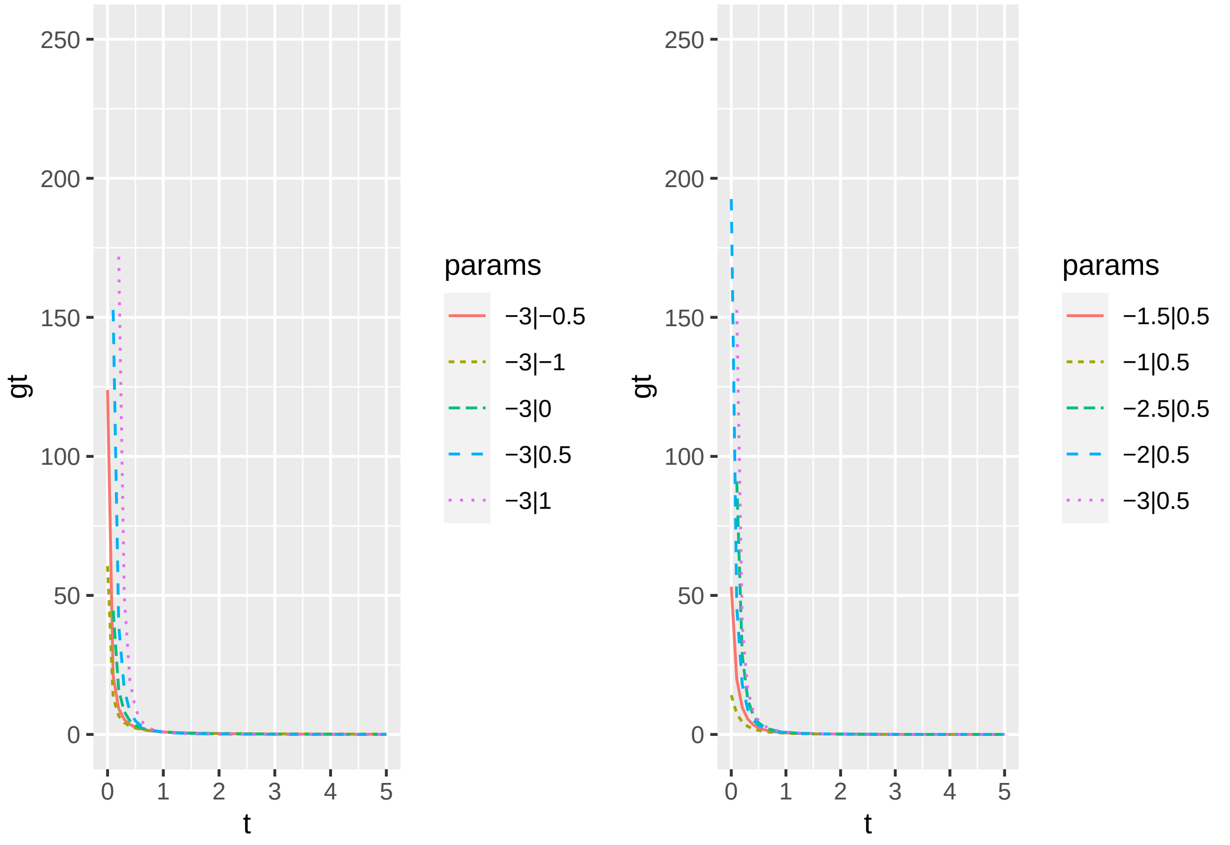
The second function, $g_t(\cdot)$ represents the temporal decay of the influence of a past event on the present. It is given by the Omori's law and has two parameters θ_4 and θ_5

$$g_t(t - t_i) = \frac{1}{(t - t_i + \exp\theta_4)^{1+\exp\theta_5}}$$

```r
source('ETAS_utils.R')
tt <- seq(1e-5, 5 + 1e-5, by = 0.1)
ht <- 0

th4.seq <- seq(-3, -1, by = 0.5)
th5.seq <- seq(-1, 1, by = 0.5)

th.gr <- expand.grid(th4.seq, th5.seq)
gt.list <- lapply(1:nrow(th.gr), function(i)
  data.frame(t = tt,
             gt = g.t(tt, th.gr[i, 1], th.gr[i, 2], ht),
             params = paste(th.gr[i,1], th.gr[i,2], sep = '|'),
             th4 = th.gr[i,1], th5 = th.gr[i,2]))

gt.df <- bind_rows(gt.list)

pl.left <-
  ggplot(gt.df[gt.df$th4 == -3,], aes(x = t, y = gt, color = params, linetype = params)) +
  geom_line() + ylim(0, 250)

pl.right <-
```

```
  ggplot(gt.df[gt.df$th5 == 0.5,], aes(x = t, y = gt, color = params, linetype = params)) +
  geom_line()+ ylim(0, 250)

multiplot(pl.left, pl.right, cols = 2)
```

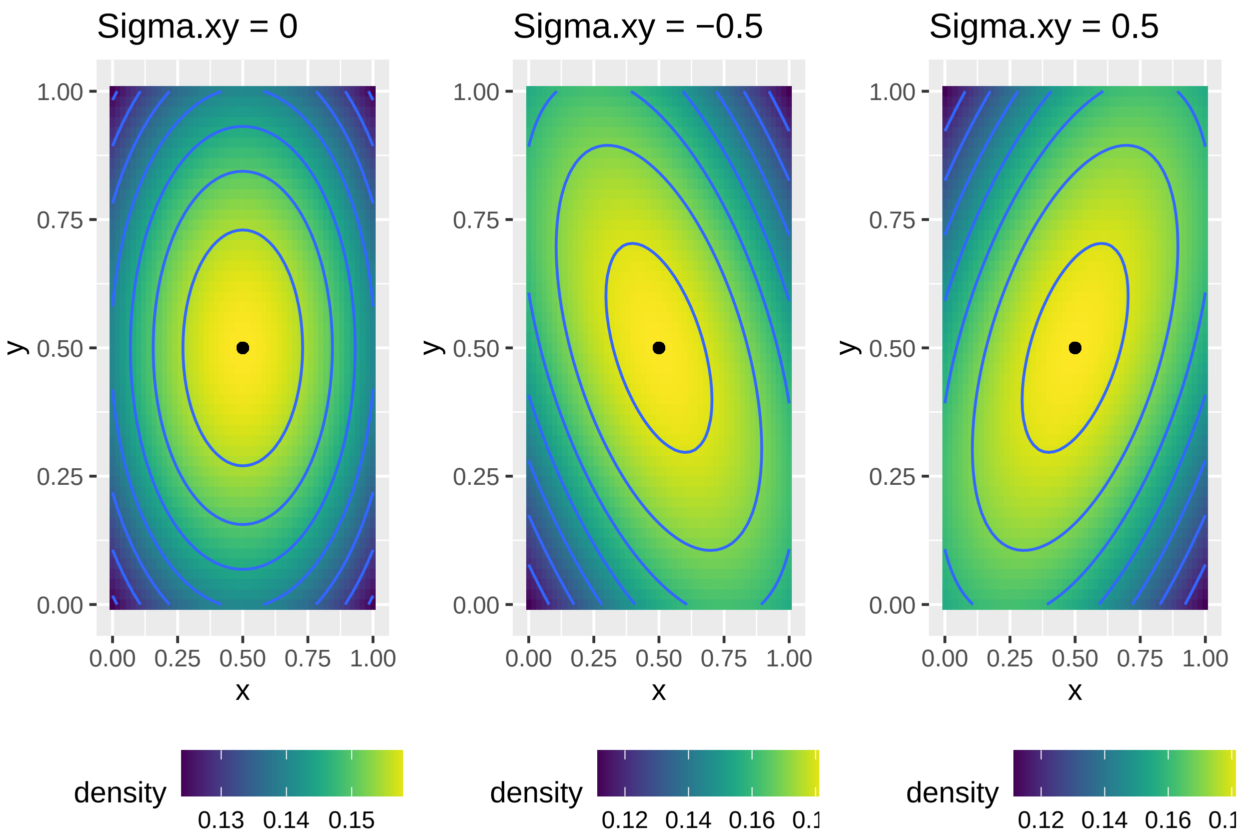
```
## Warning: Removed 4 row(s) containing missing values (geom_path).

## Warning: Removed 2 row(s) containing missing values (geom_path).
```



$$g_s(\mathbf{s}-\mathbf{s}_i) = \frac{1}{2\pi}(\det\Sigma)^{-1/2}\exp\{-(\mathbf{s}-\mathbf{s}_i)^T\Sigma^{-1}(\mathbf{s}-\mathbf{s}_i)\}$$

Where Σ is a 2×2 covariance matrix. It has to be symmetric so, in this case, we need to specify three elements which will determine the shape of the triggering function. The location is governed by the mean which, in this case, is considered to be the observed point $\mathbf{s}_i$.

```
library(mvtnorm)

toplot.gs <- function(Mu, Sigma, x.seq, y.seq){

  df <- expand.grid(x.seq, y.seq)
  colnames(df) <- c('x', 'y')
  df$density <- sapply(1:nrow(df), function(x) dmvnorm(df[x,1:2], mean = Mu,
                                                       sigma = Sigma))

  ggplot(df, aes(x = x, y = y, fill = density, z = density)) +
    geom_tile() + geom_contour() +
    scale_fill_viridis() +
    geom_point(aes(x = Mu[1], y = Mu[2])) +
    labs(title = paste0('Sigma.xy = ', Sigma[2,1])) +
    theme(legend.position = 'bottom')
}

Mu = rep(0.5, 2)
x.seq <- seq(0,1,by = 0.02)
y.seq <- seq(0,1,by = 0.02)

# indipendence
Sigma = matrix(c(1,0,0, 1), byrow = T, ncol = 2)
plotgs.1 <- toplot.gs(Mu, Sigma, x.seq, y.seq)

# negative corr
Sigma = matrix(c(1,-0.5,-0.5, 1), byrow = T, ncol = 2)
plotgs.2 <- toplot.gs(Mu, Sigma, x.seq, y.seq)

# positive corr
Sigma = matrix(c(1, 0.5, 0.5, 1), byrow = T, ncol = 2)
plotgs.3 <- toplot.gs(Mu, Sigma, x.seq, y.seq)

multiplot(plotgs.1, plotgs.2, plotgs.3, cols = 3)
```



Given the above specification of the triggering functions, we need to integrate them to retrieve the expected number of points in a given region of the domain.

For the time triggering function:

$$I_t(t_i, T_1, T_2) = \int_{max(t_i, T_1)}^{T_2} g_t(t - t_i) dt$$

$$= \int_{max(t_i, T_1)}^{T_2} (t - t_i + \exp\theta_4)^{-1-\exp\theta_5} dt$$

$$= \frac{(\max(0, T_1 - t_i) + \exp\theta_4)^{-\exp\theta_5} - (T_2 - t_i + \exp\theta_4)^{-\exp\theta_5}}{\exp\theta_5}$$

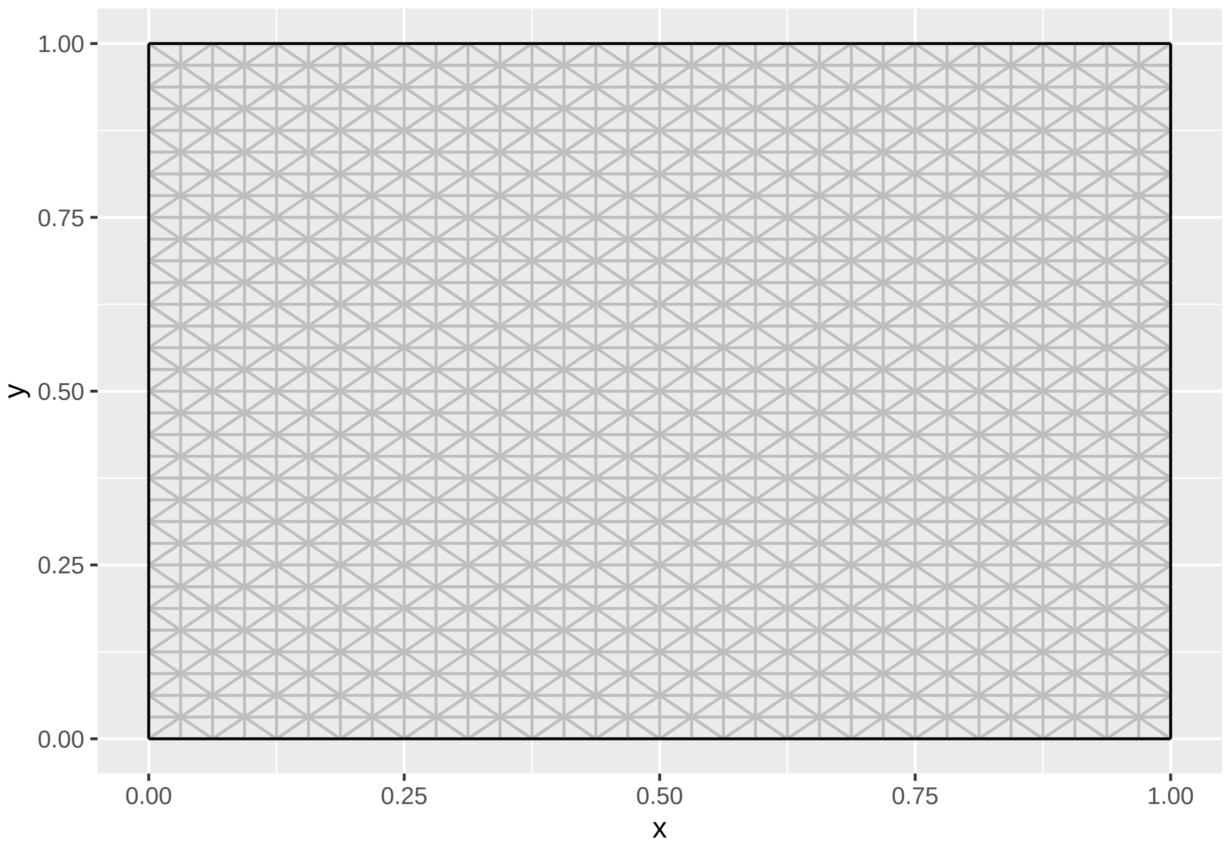
For the spatial triggering function, being a multivariate Gaussian density function:

$$I_g(A) = \int_A g_s(\mathbf{s} - \mathbf{s}_i) d\mathbf{s}$$

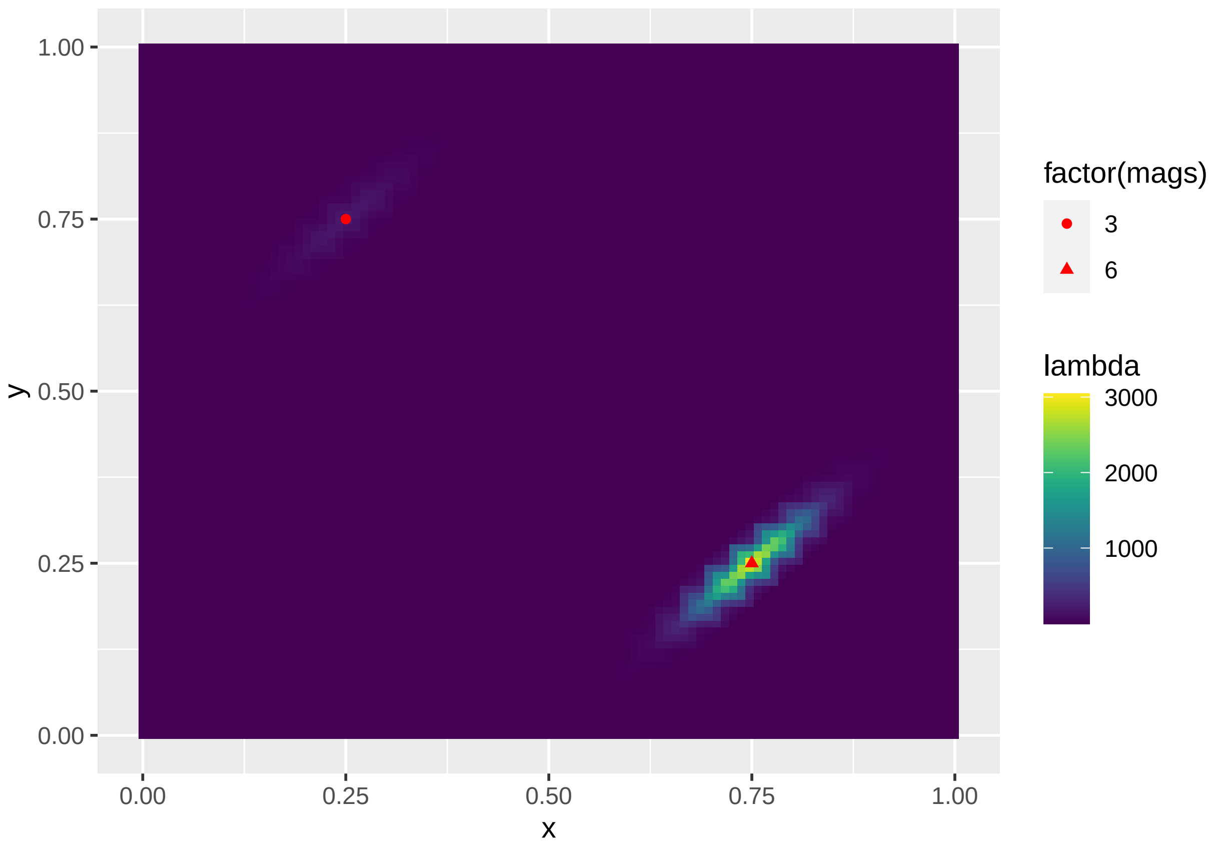
$$= \Pr(\mathbf{Z} \in A)$$

Where $\mathbf{Z} \sim N(\mathbf{s}_i, \Sigma)$ it's a multivariate Gaussian distribution with mean given by the observed location $\mathbf{s}_i$ and covariance matrix Σ.

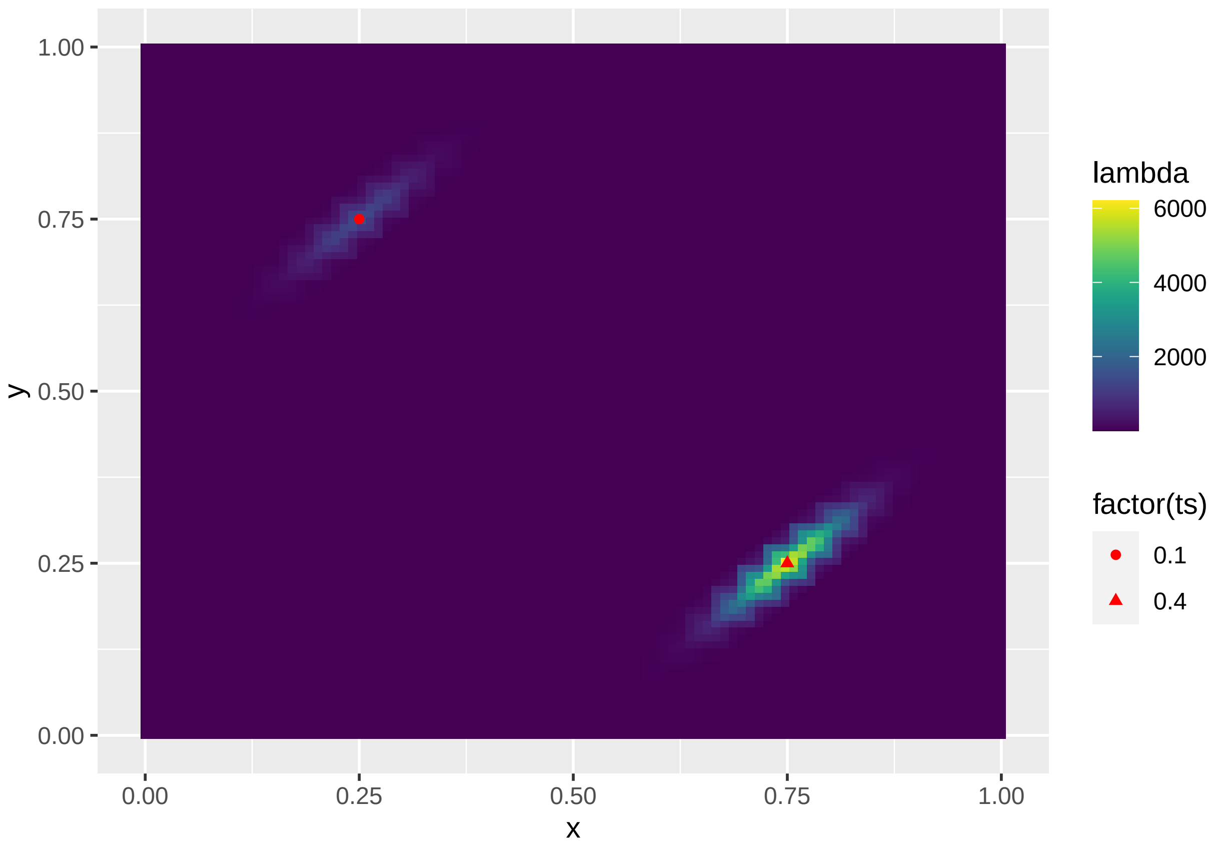
```r
x_coords <- c(0,1,1,0,0)
y_coords <- c(0,0,1,1,0)

poly1 <- sp::Polygon(cbind(x_coords,y_coords))
bdy <- sp::Polygons(list(poly1), ID = "A")
bdy <- sp::SpatialPolygons(list(bdy))
crswgs <- CRS("+proj=longlat +datum=WGS84")
proj4string(bdy) <- crswgs
bdy <- spTransform(bdy, crswgs)

mesh.n <- inla.mesh.2d(boundary = bdy, max.edge = 0.05)
mesh.n$crs <- crswgs

ggplot() + gg(mesh.n)
```



```r
# effect of magnitude 3 vs 6

Ht <- data.frame(long = c(0.25, 0.75),
                 lat = c(0.75, 0.25),
                 ts = 0.3,
                 mags = c(3,6))

theta.v <- log(c(1e-5, 1e-1, 0.9, 1e-2, 0.1, 0.002, 0.002, 0.0019))
current <- data.frame(ts = 0.5,
                      long = mesh.n$loc[,1],
                      lat = mesh.n$loc[,2])

loglambda <- sapply(1:nrow(current), function(x)
  logLambda(current[x,], theta.v, Ht, M0))

mm <- inla.mesh.projector(mesh.n, dims = c(100, 100))
mm2 <- inla.mesh.project(mm, exp(loglambda))
mm3 <- pixels(mesh.n, nx = 100, ny = 100)
mm3$lambda <- as.vector(mm2)

ggplot() + gg(mm3) +
  scale_fill_viridis() +
  geom_point(data = Ht,
             mapping = aes(x = long, y = lat, shape = factor(mags)),
             color = 'red')
```

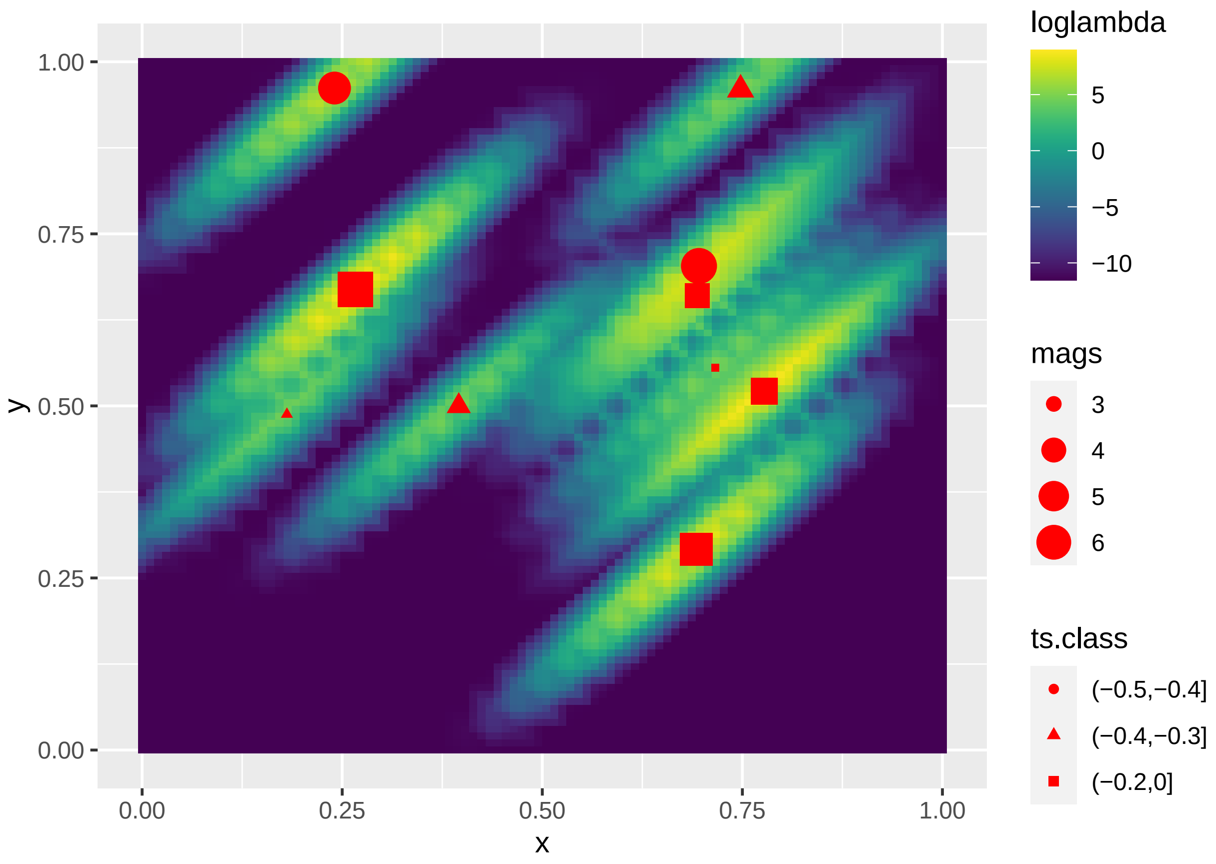


```
# effect of time
Ht <- data.frame(long = c(0.25, 0.75),
                 lat = c(0.75, 0.25),
                 ts = c(0.1, 0.4),
                 mags = c(6,6))

current <- data.frame(ts = 0.5,
                      long = mesh.n$loc[,1],
                      lat = mesh.n$loc[,2])

loglambda <- sapply(1:nrow(current), function(x)
  logLambda(current[x,], theta.v, Ht, M0))

mm <- inla.mesh.projector(mesh.n, dims = c(100, 100))
mm2 <- inla.mesh.project(mm, exp(loglambda))
mm3 <- pixels(mesh.n, nx = 100, ny = 100)
mm3$lambda <- as.vector(mm2)

ggplot() + gg(mm3) +
  scale_fill_viridis() +
  geom_point(data = Ht,
             mapping = aes(x = long, y = lat, shape = factor(ts)),
             color = 'red')
```



```r
# a <- spsample(bdy, 10, 'random')
# Ht <- data.frame(long = a@coords[,1],
#                  lat = a@coords[,2],
#                  ts = runif(10, 0, 0.5 - 1e-4),
#                  mags = runif(10, 2.5, 7))
#
# Ht$ts.class <- cut(Ht$ts - 0.5, breaks = c(-0.5, -0.4, -0.3, -0.2, 0))
#

# save(Ht, file = 'Ht_example.RData')

load('Ht_example.RData')

theta.v <- log(c(1e-5, 1e-1, 0.9, 1e-2, 0.1, 0.002, 0.002, 0.0019))
current <- data.frame(ts = 0.5,
                      long = mesh.n$loc[,1],
                      lat = mesh.n$loc[,2])

loglambda <- sapply(1:nrow(current), function(x)
  logLambda(current[x,], theta.v, Ht, M0))

mm <- inla.mesh.projector(mesh.n, dims = c(100, 100))
mm2 <- inla.mesh.project(mm, loglambda)
mm3 <- pixels(mesh.n, nx = 100, ny = 100)
mm3$loglambda <- as.vector(mm2)
```

```
ggplot() + gg(mm3) +
  geom_point(data = Ht,
             mapping = aes(x = long, y = lat, size = mags, shape = ts.class),
             color = 'red') +
  scale_fill_viridis() +
  scale_color_viridis(option = 'magma')
```



```
# estimated number of points
ips <- ipoints(mesh.n)
lambda.app.int <- sum(ips$weight*exp(loglambda))
lambda.app.int
```

```
## [1] 108.099
```

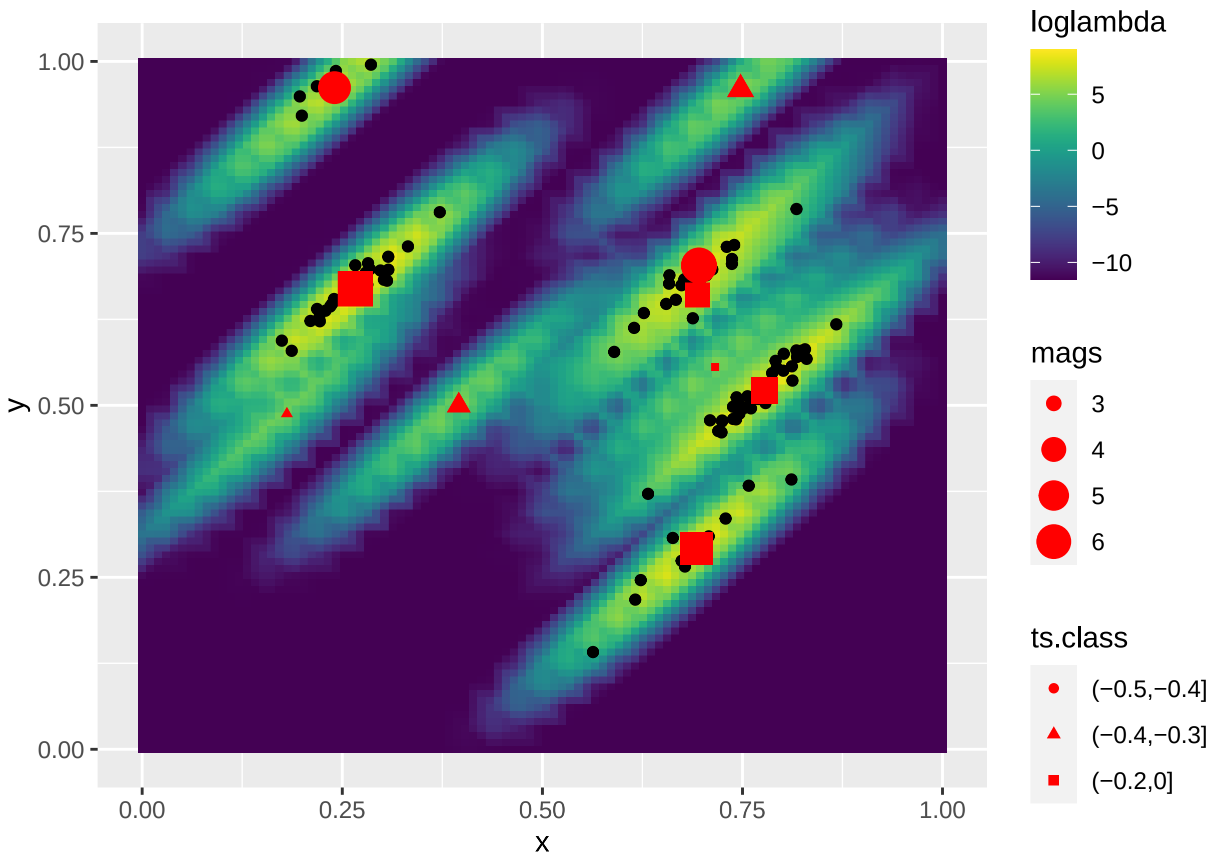
```
# sample points
n.points <- rpois(1, sum(ips$weight*exp(loglambda)))

s <- Sys.time()
ss <- point_sampler(loglambda, bdy, mesh.n, n.points, crswgs)
```

```
## [1] 633
```

```
Sys.time() - s
```

```
## Time difference of 0.6031661 secs
```

```
ggplot() + gg(mm3) +
  gg(ss) +
  geom_point(data = Ht,
             mapping = aes(x = long, y = lat, size = mags, shape = ts.class),
             color = 'red') +
  scale_fill_viridis()
```



ITALY

```
italy.collection <- read.delim('ITALY_BDY/italy.collection.nodes.dat', sep = '', header = FALSE)
colnames(italy.collection) <- c('long', 'lat')
coordinates(italy.collection) <- ~ long + lat
proj4string(italy.collection) <- crswgs
italy.collection <- spTransform(italy.collection, crswgs)

italy.grid <- SpatialPixelsDataFrame(italy.collection, proj4string = crswgs,
                            data = data.frame(id = 1:11207))
italy.contour <- adehabitatMA::getcontour(italy.grid)
```

```
## Registered S3 methods overwritten by 'adehabitatMA':
##   method                       from
##   print.SpatialPixelsDataFrame sp
##   print.SpatialPixels          sp
```

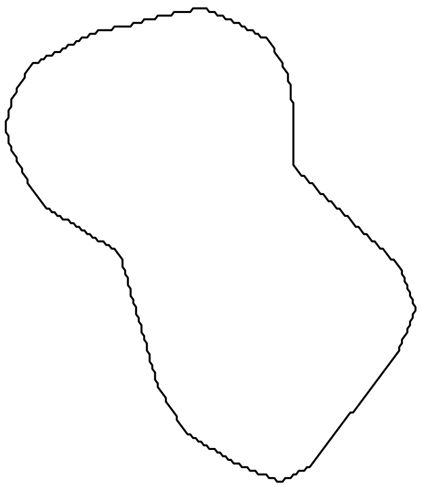
```
class(italy.contour)
```

```
## [1] "SpatialPolygons"
## attr(,"package")
## [1] "sp"
```

```
italy.contour@proj4string
```

```
## CRS arguments:
##  +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
```

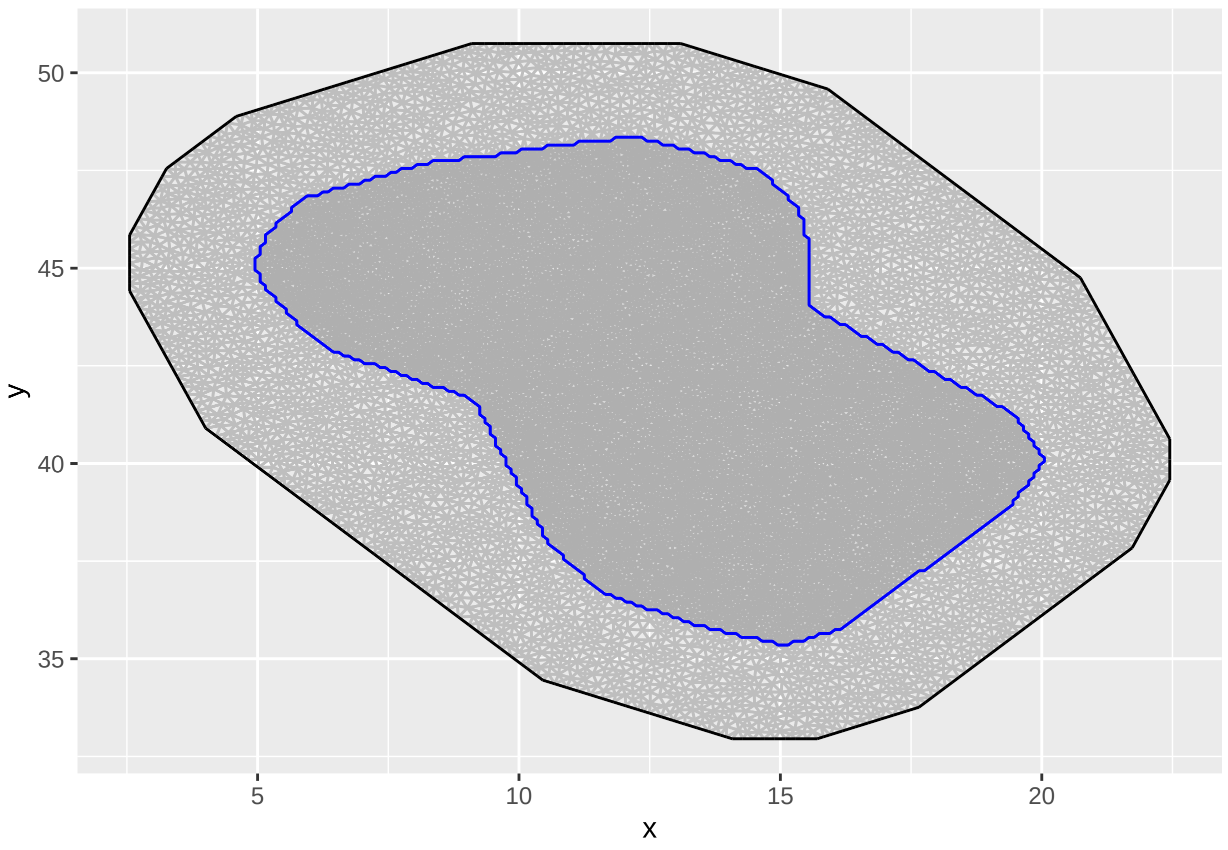
```
plot(italy.contour)
```



```
italy.mesh <- inla.mesh.2d(boundary = italy.contour, max.edge = c(0.2, 0.3))
italy.mesh$crs <- crswgs
italy.mesh$n
```

```
## [1] 10814
```

```
ggplot() + gg(italy.mesh) + gg(italy.contour, color = 'blue')
```



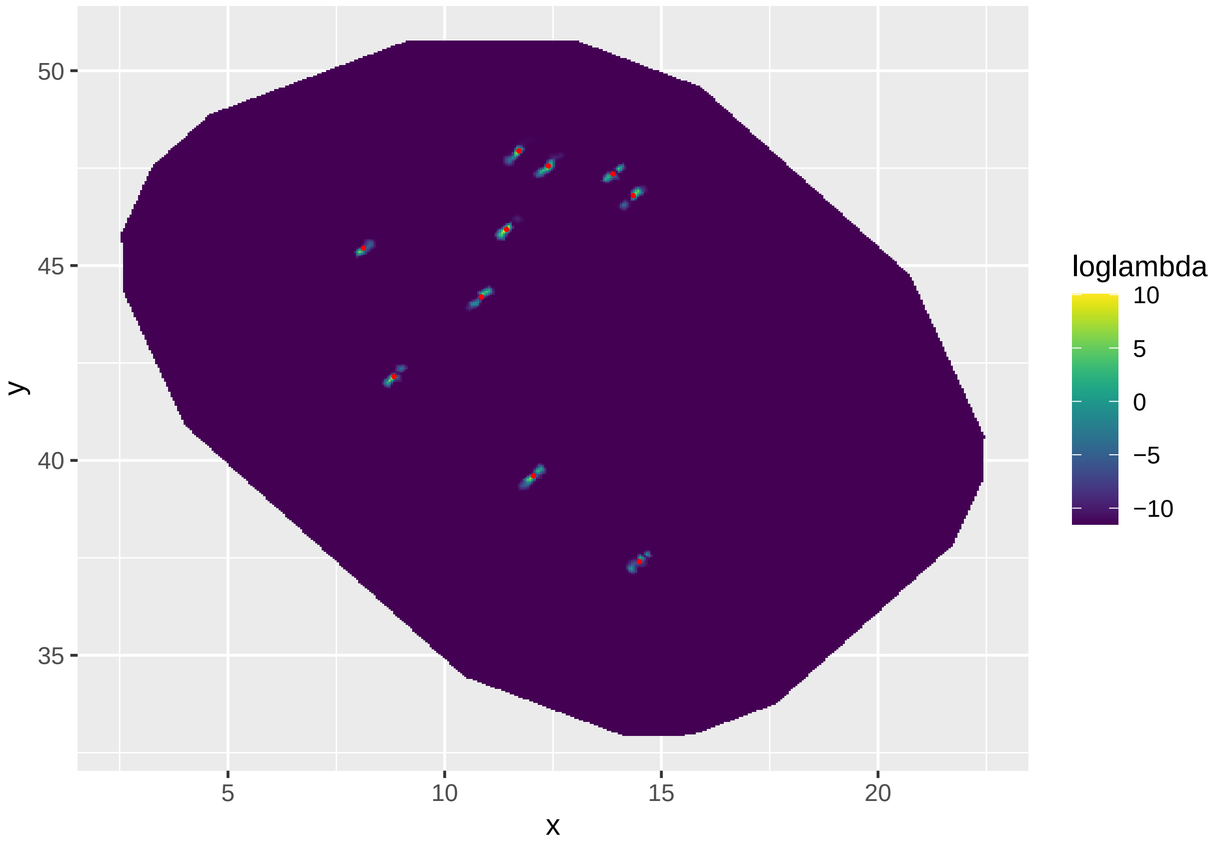
Problems with Thinning technique for sampling

In case we have a large low intensity area, the thinning technique may be really slow. This is because the majority of the homogeneously sampled points will be discarded and we need an high number of samples to reach a specified number of retained points.

Practical example: simultaneously simulate the aftershocks of a set of events. We take 100 points at random and we want to generate their aftershocks.

```
a <- spsample(italy.contour, 10, 'random')
Ht <- data.frame(long = a@coords[,1],
                 lat = a@coords[,2],
                 ts = runif(100, 0, 0.5 - 1e-4),
                 mags = runif(100, 2.5, 7))

Ht$ts.class <- cut(Ht$ts - 0.5, breaks = c(-0.5, -0.4, -0.3, -0.2, 0))


# save(Ht, file = 'Ht_example.RData')

#load('Ht_example.RData')

theta.v <- log(c(1e-5, 1e-1, 0.9, 1e-2, 0.1, 0.002, 0.002, 0.0019))
current <- data.frame(ts = 0.5,
                      long = italy.mesh$loc[,1],
                      lat = italy.mesh$loc[,2])
```

```
loglambda <- sapply(1:nrow(current), function(x)
  logLambda(current[x,], theta.v, Ht, M0))


italy.pixels <- pixels(italy.mesh, nx = 400, ny = 400)
mm <- inla.mesh.projector(italy.mesh, loc = italy.pixels@coords, crs = crswgs)
mm2 <- inla.mesh.project(mm, loglambda)
italy.pixels$loglambda <- as.vector(mm2)


ggplot() + gg(italy.pixels) +
   geom_point(data = Ht,
              mapping = aes(x = long, y = lat),
              color = 'red', size = 0.2) +
  scale_fill_viridis() +
  scale_color_viridis(option = 'magma')
```



```
ips <- ipoints(italy.mesh)
lambda.app.int <- sum(ips$weight*exp(loglambda))
lambda.app.int
```

```
## [1] 1816.988
```

```
n.points <- rpois(1, sum(ips$weight*exp(loglambda)))

s <- Sys.time()
ss <- point_sampler(loglambda, italy.contour, italy.mesh, n.points, crswgs)
```

```r
Sys.time() - s

ggplot() + gg(italy.pixels) +
  gg(ss) +
  geom_point(data = Ht,
             mapping = aes(x = long, y = lat),
             color = 'red') +
  scale_fill_viridis()
```

What I think is the right way

The sampling should work as follows:

1. Sample background points according to field. So, we already need two cases here: a) the field is normalized and the number of points to be simulated is given and we have just to place them. b) the field is not normalized and we have to approximate the number of points to be simualted and then place them.
2. For each point: .a) determine the number of aftershocks generated ($Kg_m(m_i) * I_t(t_i, T_{lim}) * I_s(W)$, need a function for that). .b) place them in time (need a function for that). .c) determine the area of interest (we need a function that builds a SpatialPolygon delimiting the aftershock region, we can make search the right polygon looking at the number of aftershock generated in the aftershock region A, that would be $Kg_m(m_i) * I_t(t_i, T_{lim}) * I_s(A)$ and we can use the same function as before, I think it may be even be optimized given that we know the shape parameters of the spatial triggering). .d) use thinning technique to place them in the region A, considering how A is determining the problem of discarding too many events should be alleviated.