Chapter 5
Space-time examples

The R source for this file is available at http://www.math.ntnu.no/inla/r-_inla.org/tutorials/spde/R/spde-_tutorial-_spacetime.R

In this chapter we show an example on fitting a space-time model. This model is a separable one described on [Cameletti et al., 2012]. Basically the model is defined as a SPDE model for the spatial domain and an AR(1) model for the time dimention. The space-time separable model is defined by the kronecker product between the precision of these two models.

We provide two examples, one for discrete time domain and another when the time is discretized over a set of knots. Basically the difference appears only in the simulation process, wich is not that important. The main difference in the fitting process is that in the continuous time case we have to select time knots and build the projector matrix considering it. However, both cases allows to have different locations at different times.

5.1 Discrete time domain

In this section we show how to fit a space-time separable model, as in [Cameletti et al., 2012]. Additionally, we show the use of a categorical covariate.

5.1.1 Data simulation

We use the Paraná state border, available on INLA package, as the domain.

```
data(PRborder)
```

-

We start by defining the spatial model. Because we need that the example run faster, we use the low resolution mesh for Paraná state border created in Section 1.3.

There is two options to simulate from Cameletti's model. One is based on the marginal distribution of the latent field and another is on the conditional distribution at each time. This last option is easy as we can simulate one realization of a spatial random field for each time.

First we set $k = 12$, the time dimention

```
k <- 12
```

-

and consider the location points from the `PRprec` data in a random order

```
data(PRprec)
coords <- as.matrix(PRprec[sample(1:nrow(PRprec)), 1:2])
```

-

In the following simulation step we will use the `rspde()` function available in the file at http://www.math.ntnu.no/inla/r-_inla.org/tutorials/spde/R/spde-_tutorial-_functions.R.

The k independent realizations can be done by

```
params <- c(variance=1, kappa=1)
set.seed(1)
x.k <- rspde(coords, kappa=params[2], variance=params[1], n=k,
             mesh=prmesh1, return.attributes=TRUE)
dim(x.k)
```

```
## [1] 616  12
```

-

Now, we define the autoregressive parameter ρ

```
rho <- 0.7
```

-

and get the correlated sample over time using

```
x <- x.k
for (j in 2:k)
x[,j] <- rho*x[,j-1] + sqrt(1-rho^2)*x.k[,j]
```

-

where the $\sqrt{(1 - \rho^2)}$ term is added as we would like to consider that the innovation noise follows the stationary distribution, see [Rue and Held, 2005] and [Cameletti et al., 2012].

We can visualize the realization at the figure 5.1.1 with commands bellow

```
c100 <- rainbow(101)
par(mfrow=c(4,3), mar=c(0,0,0,0))
for (j in 1:k)
plot(coords, col=c100[round(100*(x[,j]-min(x[,j]))/diff(range(x[,j])))],
axes=FALSE, asp=1, pch=19, cex=0.5)
```

-

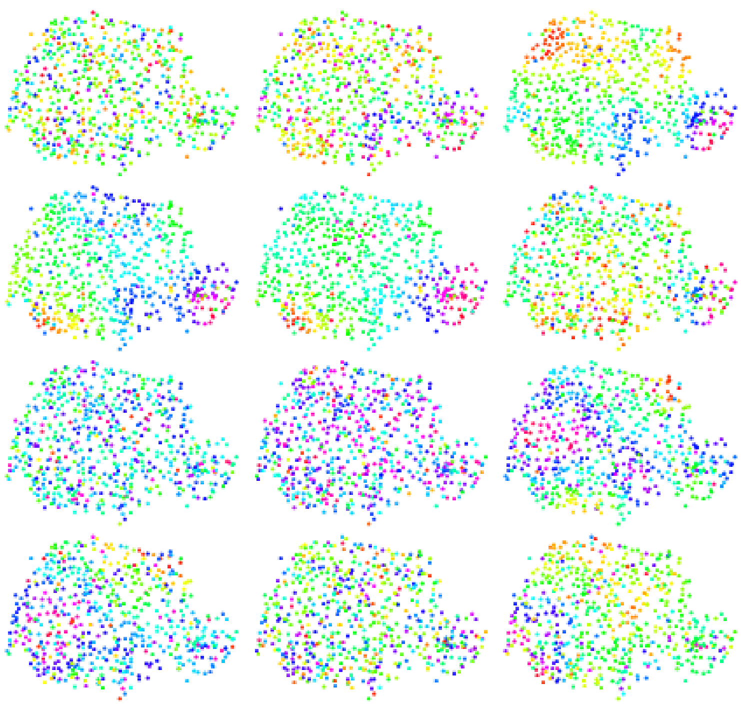


Figure 5.1: Realization of the space-time random field.

In this example we need to show the use of a categorical covariate. First we do the simulation of the covariate as

```
n <- nrow(coords)
set.seed(2)
table(ccov <- factor(sample(LETTERS[1:3], n*k, replace=TRUE)) )
```

```
##
##    A    B    C
## 2458 2438 2496
```

-

and the regression parameters as

```
beta <- -1:1
```

-

The response is

```
sd.y <- 0.1
y <- beta[unclass(ccov)] + x + rnorm(n*k, 0, sd.y)
tapply(y, ccov, mean)
```

```
##           A           B           C 
## -0.88285886  0.08870521  1.11408627
```

-

To show that is allowed to have different locations on different times, we drop some of the observations. We do it by just selecting a half of the simulated data. We do it by creating a index for the selected observations

```
isel <- sample(1:(n*k), n*k/2)
```

-

and we organize the data on a `data.frame`

```
dat <- data.frame(y=as.vector(y), w=ccov,
                  time=rep(1:k, each=n),
                  xcoo=rep(coords[,1], k),
                  ycoo=rep(coords[,2], k))[isel, ]
```

-

In real applications some times we have completely missaligned locations between different times. The code provided here to fit the model still work on this situation.

5.1.2 Data stack preparation

Defining the SPDE model considering the PC-prior derived in [Fuglstad et al., 2017] for the model parameters as the practical range, $\sqrt{8\nu}/\kappa$, and the marginal standard deviation.

```
spde <- inla.spde2.pcmatern(
  mesh=prmesh1, alpha=2, ### mesh and smoothness parameter
  prior.range=c(0.5, 0.01), ### P(practic.range<0.05)=0.01
  prior.sigma=c(1, 0.01)) ### P(sigma>1)=0.01
```

-

Now, we need the data preparation to build the space-time model. The index set is made taking into account the number of weights on the SPDE model and the number of groups

```
iset <- inla.spde.make.index('i', n.spde=spde$n.spde, n.group=k)
```

-

Notice that the index set for the latent field is not depending on the data set locations. It only depends on the SPDE model size and on the time dimention.

The projector matrix must be defined considering the coordinates of the observed data. We have to inform the time index for the group to build the projector matrix. This also must be defined on the `inla.spde.make.A()` function

```
A <- inla.spde.make.A(mesh=prmesh1,
                      loc=cbind(dat$xcoo, dat$ycoo),
```

```
group=dat$time)
```

-

The effects on the stack are a list with two elements, one is the index set and another the categorical covariate. The stack data is defined as

```
sdat <- inla.stack(tag='stdata', data=list(y=dat$y),
                   A=list(A,1),  effects=list(iset, w=dat$w))
```

-

5.1.3 Fitting the model and some results

We set the PC-prior for the temporal autoregressive parameter with $P(cor > 0) = 0.9$

```
h.spec <- list(theta=list(prior='pccor1', param=c(0, 0.9)))
```

-

The likelihood hyperparameter is fixed on a hight precision, just because we haven’t noise. To deal with the categorical covariate we need to set `expand.factor.strategy='inla'` on the `control.fixed` argument list.

```
formulae <- y ~ 0 + w +
  f(i, model=spde, group=i.group,
    control.group=list(model='ar1', hyper=h.spec))
prec.prior <- list(prior='pc.prec', param=c(1, 0.01))
res <- inla(formulae,  data=inla.stack.data(sdat),
            control.predictor=list(compute=TRUE, A=inla.stack.A(sdat)),
            control.family=list(hyper=list(theta=prec.prior)),
            control.fixed=list(expand.factor.strategy='inla'))
```

-

Summary for the trhee intercepts (and the observed mean for each covariate level)

```
round(cbind(observed=tapply(dat$y, dat$w, mean), res$summary.fixed), 4)
```

```
##   observed    mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## A  -0.9242 -0.7795 0.3467    -1.4635  -0.7799    -0.0945 -0.7806   0
## B   0.0245  0.2182 0.3467    -0.4658   0.2178     0.9032  0.2170   0
## C   1.1124  1.2246 0.3467     0.5406   1.2242     1.9096  1.2235   0
```

-

Look a the posterior marginal distributions for the random field parameters and the marginal ditribution for the temporal correlation, on the Figure 5.1.3 with the commands bellow

```
par(mfrow=c(2,2), mar=c(3,3,1,0.1), mgp=2:0)
for (j in 1:4) {
  plot(res$marginals.hyper[[j]], type='l',
       xlab=names(res$marginals.hyper)[j], ylab='Density')
  abline(v=c(1/sd.y^2, sqrt(8)/params[1],
             params[2]^0.5, rho)[j], col=2)
}
```

-

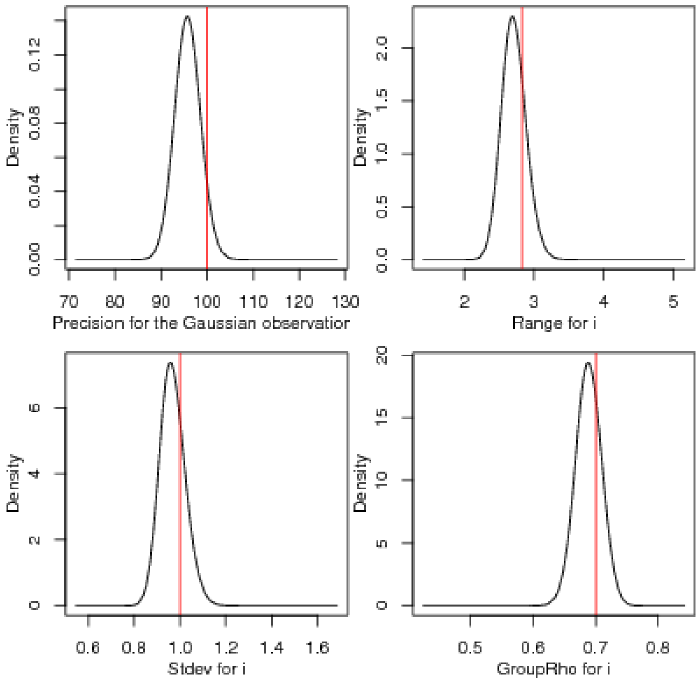


Figure 5.2: Marginal posterior distribution for the practical range (left), standard deviation of the field (mid) and the temporal corrlation (right). The red vertical lines are placed at true value.

5.1.4 A look at the posterior random field

The first look at the random field posterior distribution is to compare the realized random field with the posterior mean, median or/and mode and any quantile.

First, we found the index for the random field at data locations

```
str(idat <- inla.stack.index(sdat, 'stdata')$data)
```

```
##  int [1:3696] 1 2 3 4 5 6 7 8 9 10 ...
```

-

The correlation between the simulated data response and the posterior mean of the predicted values (there is no error term in the model):

```
cor(dat$y, res$summary.linear.predictor$mean[idat])
```

```
## [1] 0.9982051
```

-

We also can do prediction for each time and visualize it. First, we define the projection grid in the same way that in the rainfall example in Section 2.1.

```
stepsize <- 4*1/111
nxy <- round(c(diff(range(coords[,1])), diff(range(coords[,2])))/stepsize)
projgrid <- inla.mesh.projector(prmesh1, xlim=range(coords[,1]),
                                ylim=range(coords[,2]), dims=nxy)
```

-

The prediction for each time can be done by

```
xmean <- list()
for (j in 1:k)
```

```
xmean[[j]] <- inla.mesh.project(
  projgrid, res$summary.random$i$mean[iset$i.group==j])
```

-

We found what points of the grid are inside the Paraná state border.

```
library(splancs)
xy.in <- inout(projgrid$lattice$loc, cbind(PRborder[,1], PRborder[,2]))
```

-

To plot, we set NA to the points of the grid out of the Paraná border.

```
for (j in 1:k)   xmean[[j]][!xy.in] <- NA
```

-

The visualization at Figure 5.1.4 can be made by the comands bellow

```
library(gridExtra)
do.call(function(...) grid.arrange(..., nrow=4),
        lapply(xmean, levelplot,  xlab='', ylab='',
               col.regions=topo.colors(16), scale=list(draw=FALSE)))
```

-

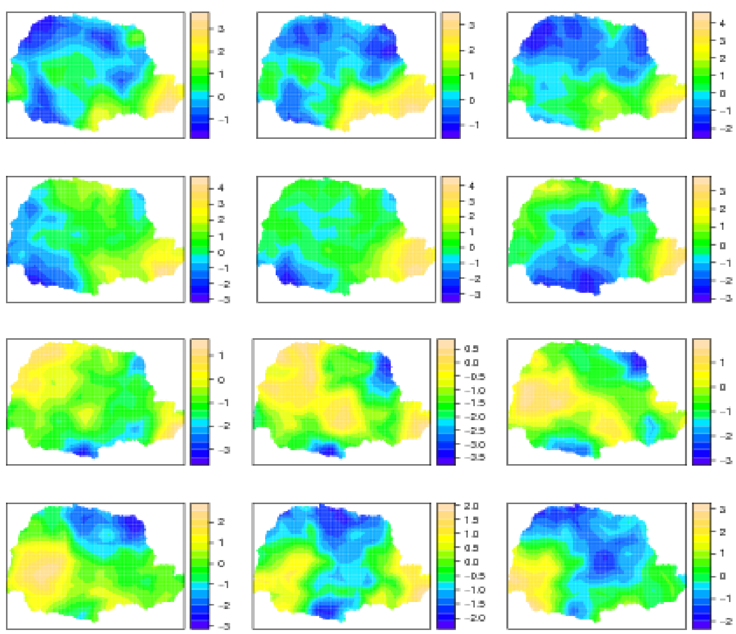


Figure 5.3: Visualization of the posterior mean of the space-time random field.

5.1.5 Validation

The results on previous section are done using part of the simulated data. This part of the simulated data is now used as a validation data. So, we prepare another data stack to compute posterior distributions to this part of the data:

```
vdat <- data.frame(r=as.vector(y), w=ccov, t=rep(1:k, each=n),
                   x=rep(coords[,1], k), y=rep(coords[,2], k))[-isel, ]
Aval <- inla.spde.make.A(prmesh1, loc=cbind(vdat$x, vdat$y), group=vdat$t)
stval <- inla.stack(tag='stval', data=list(y=NA), ### set NA in order to predict
                    A=list(Aval,1),  effects=list(iset, w=vdat$w))
```

-

Now, we just use a full data stack to fit the model and consider the hyperparameters values fitted before

```
stfull <- inla.stack(sdat, stval)
vres <- inla(formulae,  data=inla.stack.data(stfull),
  control.predictor=list(compute=TRUE, A=inla.stack.A(stfull)),
      control.family=list(hyper=list(theta=prec.prior)),
      control.fixed=list(expand.factor.strategy='inla'),
    control.mode=list(theta=res$mode$theta, restart=FALSE))
```

-

We can plot the predicted versus observed values to look at goodness of fit. First, we found the index for this data from full stack data.

```
ival <- inla.stack.index(stfull, 'stval')$data
```

-

We plot it with following commands and visualize at Figure 5.1.5.

```
par(mfrow=c(1,1), mar=c(3,3,0.5,0.5), mgp=c(1.75,0.5,0))
  plot(vres$summary.fitted.values$mean[ival], vdat$r,
      asp=1, xlab='Posterior mean', ylab='Observed')
            abline(0:1, col=gray(.7))
```

-

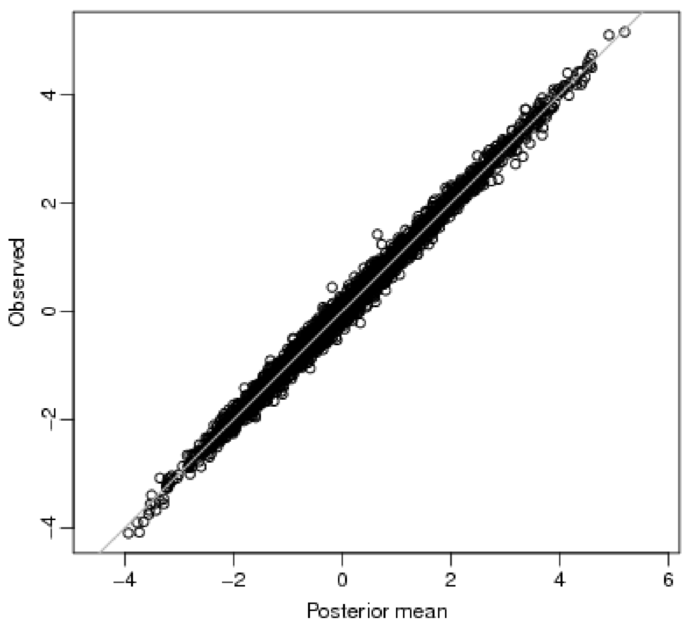


Figure 5.4: Validation: Observed versus posterior mean.

5.2 Continuous time domain

We now suppose that we have that the observations are not collected over discrete time points. This is the case for fishing data and space-time point process in general. Similar to the Finite Method approach for the space, we can use piecewise linear basis function at a set of time knots, as we have in some other spacetime examples.

5.2.1 Data simulation

We now sample some locations over space and time points as well.

```
n <- nrow(loc <- unique(as.matrix(PRprec[,1:2])))
         time <- sort(runif(n, 0, 1))
```

-

To sample from the model, we define a space-time separable covariance function, which is Matérn in space and Exponential over time:

```
stcov <- function(coords, time, kappa.s, kappa.t, variance=1, nu=1) {
  s <- as.matrix(dist(coords))
  t <- as.matrix(dist(time))
  scorr <- exp((1-nu)*log(2) + nu*log(s*kappa.s) - lgamma(nu)) *
    besselK(s*kappa.s, nu)
  diag(scorr) <- 1
  return(variance * scorr * exp(-t*kappa.t))
}
```

-

and use it to sample from the model

```
kappa.s <- 1; kappa.t <- 5; s2 <- 1/2
xx <- crossprod(chol(stcov(loc, time, kappa.s, kappa.t, s2)),
                rnorm(n))
beta0 <- -3; tau.error <- 3
y <- beta0 + xx + rnorm(n, 0, sqrt(1/tau.error))
```

-

5.2.2 Data stack preparation

To fit the space-time continuous model we need to determine the time knots and the temporal mesh

```
k <- 10
(mesh.t <- inla.mesh.1d(seq(0+0.5/k, 1-0.5/k, length=k)))$loc
```

```
##  [1] 0.05 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95
```

-

Consider the the low resolution mesh for Paraná state border created in Section 1.3, used in the previous example and the SPDE model also defined in the previous example.

Building the index set

```
iset <- inla.spde.make.index('i', n.spde=spde$n.spde, n.group=k)
```

-

The projector matrix consider the spatial and time projection. So, it needs the spatial mesh and the spatial locations, the time points and the temporal mesh

```
A <- inla.spde.make.A(mesh=prmesh1, loc=loc,
                      group=time, group.mesh=mesh.t)
```

-

The effects on the stack are a list with two elements, one is the index set and another the categorical covariate. The stack data is defined as

```
sdat <- inla.stack(tag='stdata', data=list(y=y),
       A=list(A,1),  effects=list(iset, list(b0=rep(1,n))))
```

-

5.2.3 Fitting the model and some results

We used an Exponential correlation function for time with parameter κ as the inverse range parameter. It gives a correlation between time knots equals to

```
exp(-kappa.t*diff(mesh.t$loc[1:2]))
```

```
## [1] 0.6065307
```

-

Fitting the model considering a AR1 temporal correlation over the time knots

```
formulae <- y ~ 0 + b0 +
f(i, model=spde, group=i.group,
control.group=list(model='ar1', hyper=h.spec))
res <- inla(formulae, data=inla.stack.data(sdat),
control.family=list(hyper=list(theta=prec.prior)),
control.predictor=list(compute=TRUE, A=inla.stack.A(sdat)))
```

-

Look at the summary of the posterior marginal distributions for the likelihood precision and the random field parameters:

```
round(res$summary.hyper, 4)
```

```
##                                          mean     sd 0.025quant 0.5quant
## Precision for the Gaussian observations 2.8535 0.1903     2.4918   2.8496
## Range for i                             2.4527 0.4363     1.7246   2.4070
## Stdev for i                             0.6753 0.0725     0.5423   0.6721
## GroupRho for i                          0.5124 0.1464     0.1879   0.5270
##                                         0.975quant   mode
## Precision for the Gaussian observations     3.2401 2.8443
## Range for i                                 3.4361 2.3131
## Stdev for i                                 0.8274 0.6664
## GroupRho for i                              0.7559 0.5584
```

-

These distributions are showed in Figure 5.2.3, as well also the marginal ditribution for the intercept, error precision, spatial range, standard deviation and temporal correlation in the spacetime field with the commands bellow

```
par(mfrow=c(2,3), mar=c(3,3,1,0.1), mgp=2:0)
plot(res$marginals.fixed[[1]], type='l',
xlab=expression(beta[0]), ylab='Density')
abline(v=beta0, col=2)
for (j in 1:4) {
plot(res$marginals.hyper[[j]], type='l',
xlab=names(res$marginals.hyper)[j], ylab='Density')
abline(v=c(tau.error, sqrt(8)/kappa.s, sqrt(s2),
exp(-kappa.t*diff(mesh.t$loc[1:2])))[j], col=2)
}
```

-

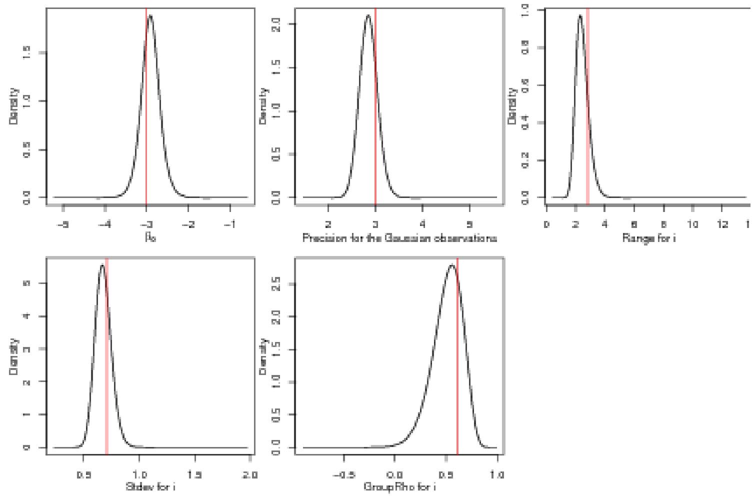


Figure 5.5: Marginal posterior distribution for the intercept, likelihood precision and the parameters in the space-time process.

5.3 Lowering resolution of a spatio-temporal model

The R source for this file is available at http://www.math.ntnu.no/inla/r-_inla.org/tutorials/spde/R/spde-_tutorial-_lower-_spatio-_temporal.R

It can be challenging when dealing with large data sets. In this chapter we want to show how to fit a model using some dimention reduction.

Before starting, the spatial mesh and the SPDE model is built with the following code.

```
data(PRprec)
bound <- inla.nonconvex.hull(as.matrix(PRprec[,1:2]), .2, .2, resol=50)
mesh.s <- inla.mesh.2d(bound=bound, max.edge=c(1,2),
                       offset=c(1e-5,0.7), cutoff=0.5)
spde.s <- inla.spde2.matern(mesh.s)
```

-

5.3.1 Data temporal aggregation

The data we are going to analyse is the daily rainfall in Paraná. We have rainfall at 616 location points observed over 365 days.

```
dim(PRprec)
```

```
## [1] 616 368
```

```
PRprec[1:2, 1:7]
```

```
##   Longitude Latitude Altitude d0101 d0102 d0103 d0104
## 1  -50.8744 -22.8511      365     0     0     0     0
## 3  -50.7711 -22.9597      344     0     1     0     0
```

-

To this example we are going to analyse the probability of rain. So we only consider if the value where bigger than 0.1 or not.

To reduce the time dimension of the data, we aggregate it summing every five days. At end we have two data matrix, one with the number of days without NA in each station and another with the number of raining days on such stations.

```
table(table(id5 <- 0:364%/%5 + 1))
```

```
##
##  5
## 73
```

```
n5 <- t(apply(!is.na(PRprec[,3+1:365]), 1, tapply, id5, sum))
y5 <- t(apply(PRprec[,3+1:365]>0.1, 1, tapply, id5, sum, na.rm=TRUE))
k <- ncol(n5);       table(as.vector(n5))
```

```
##
##     0     1     2     3     4     5
##  3563    77    72    95   172 40989
```

-

From now, our data has 73 time points.

From the above table, we can see that there were 3563 periods of five days with no data recorded. The first approach can be removing such pairs data, both y and n. If we do not remove it, we have to assign NA to y when $n = 0$. However, we have to assign a positive value, five for example, for such n and it will be treated as a prediction scenario.

```
y5[n5==0] <- NA;     n5[n5==0] <- 5
```

-

5.3.2 Lowering temporal model resolution

This approach is better expained in[Blangiardo and Cameletti, 2015]. The main idea is to place some knots over the time window and define a model on such knots. Them project the model into the data time points as we do using the Finite Element Method in the SPDE approach.

We choose to place knots at each 6 time points of the temporally aggregated data, which has 73 time points. So, we en up with only 12 knots over time.

```
bt <- 6;   gtime <- seq(1+bt, k, length=round(k/bt))-bt/2
mesh.t <- inla.mesh.1d(gtime, degree=1)
table(igr <- apply(abs(outer(mesh.t$loc, 1:k, '-')), 2, which.min))
```

```
##
##  1  2  3  4  5  6  7  8  9 10 11 12
##  7  6  6  6  6  6  6  6  6  6  6  6
```

-

The fist knot is closer to 7 time blocks and the others to 6.

The model dimention is then

```
spde.s$n.spde*mesh.t$n
```

```
## [1] 1152
```

-

To built the spatial projector matrix, we need to replicate the spatial coordinates as

```
n <- nrow(PRprec)
st.sloc <- cbind(rep(PRprec[,1], k), rep(PRprec[,2], k))
```

-

and then to consider the temporal mesh considering the group index in the scale of the data to be analised.

```
Ast <- inla.spde.make.A(mesh=mesh.s, loc=st.sloc,
                        group.mesh=mesh.t, group=rep(1:k, each=n))
```

-

The index set and the stack is built as usual

```
idx.st <- inla.spde.make.index('i', n.spde=spde.s$n.spde,
                               n.group=mesh.t$n)
dat <- inla.stack(data=list(yy=as.vector(y5), nn=as.vector(n5)),
                  A=list(Ast, 1),
                  effects=list(idx.st,
                    data.frame(mu0=1,
                  altitude=rep(PRprec$Alt/1e3, k))))
```

-

The formula is also as the usual for the separable spatio temporal model

```
form <- yy ~ 0 + mu0 + altitude +
  f(i, model=spde.s, group=i.group,
    control.group=list(model='ar1'))
```

-

To "fit" the model as fast as possible, we use the 'gaussian' approximation and the Empirical Bayes ('eb') integration strategy over the hyperparameters. We also fixed the mode at the values we have find in previous analisys.

```
result <- inla(form, 'binomial', data=inla.stack.data(dat),
               Ntrials=inla.stack.data(dat)$nn,
               control.predictor=list(A=inla.stack.A(dat)),
     control.mode=list(theta=c(-0.48, -0.9, 2.52)), ###restart=TRUE),
        control.inla=list(strategy='gaussian', int.strategy='eb'))
```

-

We can plot the fitted spatial effect for each temporal knot and overlay the proportion raining days considering the data closest to the time knots.

Defining a grid to project

```
data(PRborder)
r0 <- diff(range(PRborder[,1]))/diff(range(PRborder[,2]))
prj <- inla.mesh.projector(mesh.s, xlim=range(PRborder[,1]),
              ylim=range(PRborder[,2]), dims=c(100*r0, 100))
in.pr <- inout(prj$lattice$loc, PRborder)
```

-

Project the posterior mean fitted at each time knot

```
mu.spat <- lapply(1:mesh.t$n, function(j) {
r <- inla.mesh.project(prj, field=result$summary.ran$i$mean[
              1:spde.s$n.spde + (j-1)*spde.s$n.spde])
   r[!in.pr] <- NA;   return(r)})
```

-

The images in Figure 5.3.2 were made using the following commands

```
par(mfrow=c(4,3), mar=c(0,0,0,0))
zlm <- range(unlist(mu.spat), na.rm=TRUE)
for (j in 1:mesh.t$n) {
image.plot(x=prj$x, y=prj$y, z=mu.spat[[j]], asp=1, axes=FALSE, zlim=zlm)
lines(PRborder)
points(PRprec[, 1:2],
cex=rowSums(y5[, j==igr], na.rm=TRUE)/rowSums(n5[,j==igr]))
}
```

-

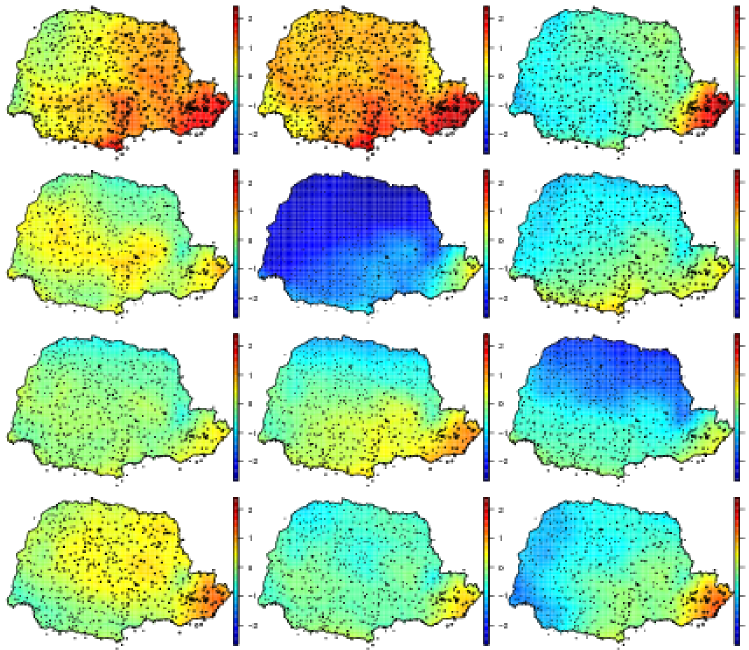


Figure 5.6: Spatial effect at each time knots.

5.4 Space-time coregionalization model

The R source for this file is available at http://www.math.ntnu.no/inla/r-_inla.org/tutorials/spde/R/spde-_tutorial-_stcoregionalization.R

```
## Loading required package: sp

## Loading required package: Matrix

## This is INLA 0.0-1480483130, dated 2016-11-30 (23:34:36+0100).
## See www.r-inla.org/contact-us for how to get help.
```

-

In this Chapter we present a way to fit a spacetime version of the Bayesian spatial coregionalization model proposed by [Schimdt and Gelfand, 2003]. Because we do the modeling with SPDE that consider the model on a mesh and it can be considered projections for other points in the spacetime domain. This is an important point as we can have the outcomes measured at different points in space and time. The only need is to have the data in the same spacetime domain.

WARNING: a crude mesh and empirical Bayes is used in order to run this example in a short time.

5.4.1 The model and parametrization

The case of three outcomes is defined considering the following equations

$$y_1(s,t) = \alpha_1 + z_1(s,t) + e_1(s,t)$$

$$y_2(s,t) = \alpha_2 + \lambda_1 y_1(s,t) + z_2(s,t) + e_2(s,t)$$

$$y_3(s,t) = \alpha_3 + \lambda_2 y_1(s,t) + \lambda_3 y_2(s,t) + z_3(s,t) + e_3(s,t)$$

where the $z_k(s,t)$ are spacetime correlated processes and $e_k(s,t)$ are uncorrelated error terms, $k = 1,2,3$.

In order to fit this model in `R-INLA` we consider a reparametrization. This reparametrization is to change the second equation as follows

$$\begin{aligned} y_2(s,t) &= \alpha_2 + \lambda_1[\alpha_1 + z_1(s,t) + e_1(z,t)] + z_2(s,t) + e_2(s,t) \\ &= (\alpha_2 + \lambda_1\alpha_1) + \lambda_1[z_1(s,t) + e_1(s,t)] + z_2(s,t) + e_2(s,t) \end{aligned}$$

and the third equation as follows

$$
\begin{aligned}
y_3(s,t) &= \alpha_3 + \lambda_2(\alpha_2 + \lambda_1\alpha_1) + \lambda_2\lambda_1[z_1(s,t) + e_1(s,t)] + \lambda_3\{\alpha_2 + \lambda_1\alpha_1 + \lambda_1[z_1(s,t) + e_1(s,t)] + z_2(s,t) + e_2(s,t)\} + z_3(s,t) \\
&= [\alpha_3 + \lambda_2\alpha_1 + \lambda_3(\alpha_2 + \lambda_1\alpha_1)] + \qquad (5.1) \\
&\quad (\lambda_2 + \lambda_3\lambda_1)[z_1(s,t) + e_1(s,t)] + \lambda_3[z_2(s,t) + e_2(s,t)] + z_3(s,t) + e_3(s,t)
\end{aligned}
$$

We have then two new intercepts $\alpha_2^* = \alpha_2 + \lambda_1\alpha_1$ and $\alpha_3^* = \alpha_3 + \lambda_2(\alpha_2 + \lambda_1\lambda_1) + \lambda_3(\alpha_2 + \lambda_1\alpha_1)$. We also have one new regression coeffiecient $\lambda_2^* = \lambda_2 + \lambda_3\lambda_1$.

This model can be fitted in `R-INLA` using the copy feature. In the parametrization above it is needed to copy the linear predictor in the first equation to the second and the linear predictor in the second equation to the third.

We will use the copy feature to fit $\lambda_1 = \beta_1$. In the second equation and $\lambda_2 + \lambda_3\lambda_1 = \beta_2$ will be the first copy parameter in the third equation. A second copy will be used in the third equation to fit $\lambda_3 = \beta_3$.

5.4.2 Data simulation

Parameter setting

```
alpha <- c(-5, 3, 10) ### intercept on reparametrized model
m.var <- (3:5)/10 ### random field marginal variances
kappa <- c(12, 10, 7) ### GRF scales: inverse range parameters
beta <- c(.7, .5, -.5) ### copy par.: reparam. coregionalization par.
rho <- c(.7, .8, .9) ### temporal correlations
n <- 300;  k <- 9  ### number of spatial locations and time points
```

-

When working with SPDE models is not required for the spatial locations to be the same for each process to fit this model in `R-INLA`, as shown in the Chapter 8 of [Blangiardo and Cameletti, 2015] and in the measurement error example in Section 3.1. As we define the model over a set of time knots to fit a spacetime continuous random field, it is also not required for the spacetime coordinates from each outcome to be the same. However, to simplify the code, we just use the same spatial locations and the same time points for all three processes.

```
loc <- cbind(runif(n), runif(n))
```

-

We can use the `rMatern()` function defined in the section 1.1.4 to simulate independent random field realizations for each time. This function is available in the file at http://www.math.ntnu.no/inla/r-_inla.org/tutorials/spde/R/spde-_tutorial-_functions.R

```
x1 <- rMatern(k, loc, kappa[1], m.var[1])
x2 <- rMatern(k, loc, kappa[2], m.var[2])
x3 <- rMatern(k, loc, kappa[3], m.var[3])
```

-

The time evolution will follows an autoregressive first order process as we used in Chapter 5.

```
z1 <- x1; z2 <- x2; z3 <- x3
for (j in 2:k) {
z1[, j] <- rho[1] * z1[,j-1] + sqrt(1-rho[1]^2) * x1[,j]
z2[, j] <- rho[2] * z2[,j-1] + sqrt(1-rho[2]^2) * x2[,j]
z3[, j] <- rho[3] * z3[,j-1] + sqrt(1-rho[3]^2) * x3[,j]
}
```

-

The term $\sqrt{(1-\rho^2)}$ is because we are sampling from the stationary distribution, and is in accord to the first order autoregressive process parametrization implemented in `R-INLA`.

Then we define the observation samples

```
e.sd <- c(0.3, 0.2, 0.15)
y1 <- alpha[1] + z1 + rnorm(n, 0, e.sd[1])
y2 <- alpha[2] + beta[1] * z1 + z2 + rnorm(n, 0, e.sd[2])
y3 <- alpha[3] + beta[2] * z1 + beta[3] * z2 + z3 +
rnorm(n, 0, e.sd[3])
```

-

5.4.3 Model fitting

Build the mesh to use in the fitting process (this is a crude mesh used here for short computational time pourpose)

```
mesh <- inla.mesh.2d(loc, max.edge=0.2,
                     offset=0.1, cutoff=0.1)
```

-

Defining the SPDE model considering the PC-prior derived in [Fuglstad et al., 2017] for the model parameters as the practical range, $\sqrt{8\nu}/\kappa$, and the marginal standard deviation.

```
spde <- inla.spde2.pcmatern(
mesh=mesh, alpha=2, ### mesh and smoothness parameter
prior.range=c(0.05, 0.01), ### P(practic.range<0.05)=0.01
prior.sigma=c(1, 0.01)) ### P(sigma>1)=0.01
```

-

Defining all the index set for the space-time fields and the for the copies. As we have the same mesh, they are the same.

```
s1 = s2 = s3 = s12 = s13 = s23 = rep(1:spde$n.spde, times=k)
g1 = g2 = g3 = g12 = g13 = g23 = rep(1:k, each=spde$n.spde)
```

-

Prior for ρ_j is chosen as the Penalized Complexity prior, [Simspon et al., 2017]

```
rho1p <- list(theta=list(prior='pccor1', param=c(0, 0.9)))
ctr.g <- list(model='ar1', hyper=rho1p)
```

-

Ther prior chosen above consider $P(\rho > 0) = 0.9$.

Priors for each of the the copy parameters $N(0, 10)$

```
hc3 <- hc2 <- hc1 <- list(theta=list(prior='normal', param=c(0,10)))
```

-

The priors for the fields are the default ones, described in [Lindgren and Rue, 2013].

Define the formula including all the terms in the model.

```
form <- y ~ 0 + intercept1 + intercept2 + intercept3 +
f(s1, model=spde, ngroup=k, group=g1, control.group=ctr.g) +
f(s2, model=spde, ngroup=k, group=g2, control.group=ctr.g) +
f(s3, model=spde, ngroup=k, group=g3, control.group=ctr.g) +
f(s12, copy="s1", group=g12, fixed=FALSE, hyper=hc1) +
f(s13, copy="s1", group=g13, fixed=FALSE, hyper=hc2) +
f(s23, copy="s2", group=g23, fixed=FALSE, hyper=hc3)
```

-

Define the projector matrix (all they are equal in this example, but it can be different)

```
stloc <- kronecker(matrix(1,k,1), loc) ### rep. coordinates each time
A <- inla.spde.make.A(mesh, stloc, n.group=k, group=rep(1:k, each=n))
```

-

Organize the data in three data stack and join it

```
stack1 <- inla.stack(
  data=list(y=cbind(as.vector(y1), NA, NA)), A=list(A),
  effects=list(list(intercept1=1, s1=s1, g1=g1)))
stack2 <- inla.stack(
  data=list(y=cbind(NA, as.vector(y2), NA)), A=list(A),
  effects=list(list(intercept2=1, s2=s2, g2=g2,
                    s12=s12, g12=g12)))
stack3 <- inla.stack(
  data=list(y=cbind(NA, NA, as.vector(y3))), A=list(A),
  effects=list(list(intercept3=1, s3=s3, g3=g3,
                    s13=s13, g13=g13, s23=s23, g23=g23)))
stack <- inla.stack(stack1, stack2, stack3)
```

-

We consider a penalized complexity prior for the errors precision, [Simspon et al., 2017],

```
eprec <- list(hyper=list(theta=list(prior='pc.prec',
                                    param=c(1, 0.01))))
```

-

We have 15 hyperparameters in the model. To make the optimization process fast, we use the parameter values used in the simulation as the initial values

```
theta.ini <- c(log(1/e.sd^2),
  c(log(sqrt(8)/kappa), log(sqrt(m.var)),
  qlogis(rho))[c(1,4,7, 2,5,8, 3,6,9)], beta)
```

-

With 15 hyperparameters in the model and the CCD strategy will use 287 integration points to compute

$$\pi(x_i|y) = \int \pi(y|x)\pi(x|\theta)\pi(\theta)d\theta$$

We avoid it using the Empirical Bayes, setting `int.strategy='eb'` and these marginals will consider only the modal configuration for θ.

```
(result <- inla(form, rep('gaussian', 3), data=inla.stack.data(stack),
                control.family=list(eprec, eprec, eprec),
                control.mode=list(theta=theta.ini, restart=TRUE),
                control.inla=list(int.strategy='eb'),
                control.predictor=list(A=inla.stack.A(stack))))$cpu
```

```
## Note: method with signature 'CsparseMatrix#Matrix#missing#replValue' chosen for function '[<-',
##  target signature 'dgCMatrix#ngCMatrix#missing#numeric'.
##  "Matrix#nsparseMatrix#missing#replValue" would also be valid

## Note: method with signature 'Matrix#numLike' chosen for function '%*%',
##  target signature 'dgTMatrix#numeric'.
##  "TsparseMatrix#ANY" would also be valid

## Note: method with signature 'sparseMatrix#matrix' chosen for function '%*%',
##  target signature 'dgTMatrix#matrix'.
##  "TsparseMatrix#ANY" would also be valid
```

-

```
##  Pre-processing    Running inla Post-processing           Total
##       1.2817590     275.5530427       0.7301118     277.5649135
```

-

```
result$logfile[grep('Number of function evaluations', result$logfile)]
```

```
## [1] "Number of function evaluations = 1063"
```

```
round(result$mode$theta, 2)
```

```
##    Log precision for the Gaussian observations
##                                           2.03
## Log precision for the Gaussian observations[2]
##                                           2.23
## Log precision for the Gaussian observations[3]
##                                           2.50
##                            log(Range) for s1
##                                          -1.33
##                            log(Stdev) for s1
##                                          -0.42
##                      Group rho_intern for s1
##                                           2.38
##                            log(Range) for s2
##                                          -1.06
##                            log(Stdev) for s2
##                                          -0.36
##                      Group rho_intern for s2
##                                           2.91
##                            log(Range) for s3
##                                          -0.81
##                            log(Stdev) for s3
##                                          -0.45
##                      Group rho_intern for s3
##                                           3.60
##                          Beta_intern for s12
##                                           0.78
##                          Beta_intern for s13
##                                           0.59
##                          Beta_intern for s23
##                                          -0.60
```

-

Summary of the posterior marginal density for the intercepts

```
round(cbind(true=alpha, result$summary.fix), 2)
```

```
##            true  mean   sd 0.025quant 0.5quant 0.975quant  mode kld
## intercept1   -5 -4.90 0.14      -5.18    -4.90      -4.62 -4.90   0
## intercept2    3  3.28 0.25       2.79     3.28       3.76  3.28   0
## intercept3   10  9.97 0.32       9.35     9.97      10.59  9.97   0
```

-

Posterior marginal for the errors precision

```
round(cbind(true=c(e=e.sd^-2), result$summary.hy[1:3, ]), 4)
```

```
##       true    mean     sd 0.025quant 0.5quant 0.975quant    mode
## e1 11.1111  7.6468 0.2234     7.2153   7.6441     8.0947  7.6395
## e2 25.0000  9.3143 0.2777     8.7766   9.3113     9.8702  9.3069
## e3 44.4444 12.2250 0.3627    11.5293  12.2190    12.9551 12.2067
```

-

Summary of the posterior marginal density for the temporal correlations:

```
round(cbind(true=rho, result$summary.hy[c(6,9,12),]), 4)
```

```
##                 true   mean     sd 0.025quant 0.5quant 0.975quant   mode
## GroupRho for s1  0.7 0.8316 0.0226     0.7841   0.8326     0.8731 0.8341
```

```
## GroupRho for s2  0.8 0.8942 0.0167     0.8576    0.8956     0.9230 0.8984
## GroupRho for s3  0.9 0.9480 0.0116     0.9230    0.9488     0.9681 0.9500
```

-

Summary of the posterior marginal density for the copy parameters:

```
round(cbind(true=beta, result$summary.hy[13:15,]), 4)
```

```
##                true    mean     sd 0.025quant 0.5quant 0.975quant    mode
## Beta for s12  0.7  0.7761 0.0499     0.6778   0.7762     0.8743  0.7764
## Beta for s13  0.5  0.5935 0.0447     0.5056   0.5935     0.6813  0.5935
## Beta for s23 -0.5 -0.5986 0.0506    -0.6981  -0.5986    -0.4990 -0.5986
```

-

Look for the random field parameters for each field. The practical range for each random field

```
round(cbind(true=sqrt(8)/kappa, result$summary.hy[c(4, 7, 10),]), 3)
```

```
##                 true  mean    sd 0.025quant 0.5quant 0.975quant  mode
## Range for s1 0.236 0.263 0.024      0.219    0.263      0.313 0.261
## Range for s2 0.283 0.350 0.034      0.290    0.348      0.421 0.345
## Range for s3 0.404 0.451 0.053      0.359    0.447      0.566 0.438
```

-

The standard deviation for each random field

```
round(cbind(true=m.var^0.5, result$summary.hy[c(5, 8, 11),]), 3)
```

```
##                 true  mean    sd 0.025quant 0.5quant 0.975quant  mode
## Stdev for s1 0.548 0.662 0.042      0.585    0.660      0.748 0.656
## Stdev for s2 0.632 0.700 0.051      0.604    0.699      0.806 0.696
## Stdev for s3 0.707 0.647 0.065      0.533    0.642      0.788 0.630
```

-

The posterior mean for each random field is projected to the observation locations and shown against the simulated correspondent fields in Figure 5.4.3 with the code bellow.

```
par(mfrow=c(1,3), mar=c(2,2,0.5,0.5), mgp=c(1.5,0.5,0))
plot(drop(A%*%result$summary.ran$s1$mean), as.vector(z1),
     xlab='', ylab='', asp=1); abline(0:1)
plot(drop(A%*%result$summary.ran$s2$mean), as.vector(z2),
     xlab='', ylab='', asp=1); abline(0:1)
plot(drop(A%*%result$summary.ran$s3$mean), as.vector(z3),
     xlab='', ylab='', asp=1); abline(0:1)
```

-

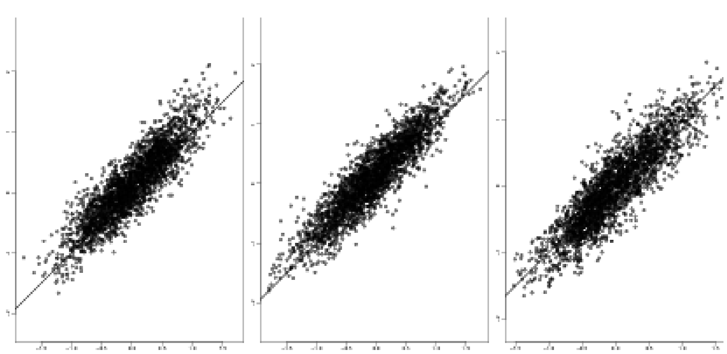


Figure 5.7: True and fitted random field values.

Remember that the crude approximation for the covariance and the simplifications on the inference procedure is not recommended to use in practice. It can be considered for having initial results. Even thou, it seems that the method was reazonable well having covered the parameter values used to simulate the data.

5.5 Dynamic regression example

The R source for this file is available at http://www.math.ntnu.no/inla/r-_inla.org/tutorials/spde/R/spde-_tutorial-_dynamic.R

There is a large literature about dynamic models with also some books about it, from [West and Harrison, 1997] to [Petris et al., 2009]. These models basically defines an hierarchical framework for a class of time series models. A particular case is the dynamic regression model, were the regression coefficients are modeled as time series. That is the case when the regression coefficients vary smoothly over time.

5.5.1 Dynamic space-time regression

The specific class of models for spatially structured time series was proposed by [Gelfand et al., 2003], where the regression coefficients varies smoothly over time and space. For the areal data case, the use of proper Gaussian Markov random fields (PGMRF) over space as proposed by [Vivar and Ferreira, 2009]. There exists a particular class of such models called “spatially varying coefficient models” were the regression coefficients veries over space, [Assunção et al., 1999], [Assunção et al., 2002], [Gamerman et al., 2003].

In [Gelfand et al., 2003] the Gibbs sampler were used for inference and it was claimed that better algorithms is needed due to strong autocorrelations. In [Vivar and Ferreira, 2009] the use of forward information filtering and backward sampling (FIFBS) recursions were proposed. Both MCMC algorithms are computationally expensive.

One can avoid the FFBS algorithm as a relation between the Kalman-filter and the Cholesky factorization is provided in [Knorr-Held and Rue, 2002]. The Cholesky fator is more general and has superior performance when using sparse matrix methods, [Rue and Held, 2005, p. 149]. Additionally, the restriction that the prior for the latent field has to be proper can be avoided.

When the likelihood is Gaussian, there is no approximation needed in the inference process since the distribution of the latent field given the data and the hyperparameters is Gaussian. So, the main task is to perform inference for the hyperparameters in the model. For this, the mode and curvature around can be found without any sampling method. For the class of models in [Vivar and Ferreira, 2009] it is natural to use INLA, as shown in [Ruiz-Cárdenas et al., 2012], and for the models in [Gelfand et al., 2003] we can use the SPDE approach when considering the Matérn covariance for the spatial part.

In this example we will show how to fit the space-time dynamic regression model as in [Gelfand et al., 2003], considering the Matérn spatial covariance and the AR(1) model for time which corresponds to the exponential correlation function. This particular covariance choise correspond to the model in [Cameletti et al., 2012], where only the intercept is dynamic. Here, we show the case when we have a dynamic intercept and a dynamic regression coefficient for an harmonic over time.

5.5.2 Simulation from the model

We can start on defining the spatial locations:

```
n <- 150;   set.seed(1);  coo <- matrix(runif(2*n), n)
```

-

To sample from a random field on a set of location, we can use the `rMatern()` function defined in the Section 1.1.4 to simulate independent random field realizations for each time. This function is available in the file at http://www.math.ntnu.no/inla/r-_inla.org/tutorials/spde/R/spde-_tutorial-_functions.R

We draw k (number of time points) samples from the random field. Then, we make it temporally correlated considering the time autoregression

```
kappa <- c(10, 12);    sigma2 <- c(1/2, 1/4)
k <- 15;  rho <- c(0.7, 0.5)
set.seed(2); beta0 <- rMatern(k, coo, kappa[1], sigma2[1])
set.seed(3); beta1 <- rMatern(k, coo, kappa[2], sigma2[2])
beta0[,1] <- beta0[,1] / (1-rho[1]^2)
beta1[,1] <- beta1[,1] / (1-rho[2]^2)
```

```
for (j in 2:k) {
beta0[, j] <- beta0[,j-1]*rho[1] + beta0[,j] * (1-rho[1]^2)
beta1[, j] <- beta1[,j-1]*rho[2] + beta1[,j] * (1-rho[2]^2)
}
```

-

where the $(1 - \rho_j^2)$ term is in accord to the parametrization of the AR(1) model in INLA.

To get the response, we define the harmonic as a function over time, compute the mean and add an error term

```
set.seed(4); hh <- runif(n*k) ### simlate the covariate values
mu.beta <- c(-5, 1);   taue <- 20
set.seed(5); error <- rnorm(n*k, 0, sqrt(1/taue)) ### error in the observation
length(y <- (mu.beta[1] + beta0) + (mu.beta[2]+beta1)*hh + ### dynamic regression part
error)
```

```
## [1] 2250
```

-

5.5.3 Fitting the model

We have two space-time terms on the model, each one with three hyperparameters: precision, spatial scale, temporal scale (or temporal correlation). So, considering the likelihood, 7 hyperparameters in total. To perform fast inference, we choose to have a crude mesh with with small number of vertices.

```
(mesh <- inla.mesh.2d(coo, max.edge=c(0.25), ### coarse mesh
offset=c(0.15), cutoff=0.05))$n
```

```
## [1] 142
```

-

Defining the SPDE model considering the PC-prior derived in [Fuglstad et al., 2017] for the model parameters as the practical range, $\sqrt{8\nu}/\kappa$, and the marginal standard deviation.

```
spde <- inla.spde2.pcmatern(
mesh=mesh, alpha=2, ### mesh and smoothness parameter
prior.range=c(0.05, 0.01), ### P(practic.range<0.05)=0.01
prior.sigma=c(1, 0.01)) ### P(sigma>1)=0.01
```

-

We do need one set of index for each call of the `f()` function, no matter if they are the same, so:

```
i0 <- inla.spde.make.index('i0', spde$n.spde, n.group=k)
i1 <- inla.spde.make.index('i1', spde$n.spde, n.group=k)
```

-

In the SPDE approach, the space-time model is defined in a set of mesh nodes. As we have considered continuous time, it is also defined on a set of time knots. So, we have to deal with the projection from the model domain (nodes, knots) to the space-time data locations. For the intercept it is the same way as in the other examples. For the regression coefficients, we need to account for the covariate value in the projection matrix. It can be seen as follows

$$\begin{aligned} \boldsymbol{\eta} &= \mu_{\beta_0} + \mu_{\beta_2}\boldsymbol{h} + \boldsymbol{A}\boldsymbol{\beta}_0 + (\boldsymbol{A}\boldsymbol{\beta}_1)\boldsymbol{h} \\ &= \mu_{\beta_0} + \mu_{\beta_1}\boldsymbol{h} + \boldsymbol{A}\boldsymbol{\beta}_0 + (\boldsymbol{A} \oplus (\boldsymbol{h}\mathbf{1}^{'}))\boldsymbol{\beta}_1 \end{aligned} \tag{5.2}$$

where $\boldsymbol{A} \oplus (\boldsymbol{h}\mathbf{1}^{'})$ is the row-wise Kronecker product between $\boldsymbol{A}$ and a the vector $\boldsymbol{h}$ (with length equal the number of rows in $\boldsymbol{A}$) expressed as the Kronecker sum of $\boldsymbol{A}$ and $\boldsymbol{h}\mathbf{1}^{1}$. This operation can be performed usind the `inla.row.kron()` function and is done internally in the function `inla.spde.make.A()` when supplying a vector in the `weights` argument.

The space-time projector matrix $\boldsymbol{A}$ is defined as follows:

```
A0 <- inla.spde.make.A(mesh, cbind(rep(coo[,1], k), rep(coo[,2], k)),
                       group=rep(1:k, each=n))
A1 <- inla.spde.make.A(mesh, cbind(rep(coo[,1], k), rep(coo[,2], k)),
                       group=rep(1:k, each=n), weights=hh)
```

-

The data stack is as follows

```
stk.y <- inla.stack(data=list(y=as.vector(y)), tag='y',
                    A=list(A0, A1, 1),
                    effects=list(i0, i1,
                    data.frame(mu1=1, h=hh)))
```

-

where `i0` is similar to `i1` and the elements `mu1` and `h` in the second element of the effects `data.frame` is for $\boldsymbol{\mu}_\xi$.

The formula take these things into account

```
form <- y ~ 0 + mu1 + h + ### to fit mu_beta
  f(i0, model=spde, group=i0.group, control.group=list(model='ar1')) +
  f(i1, model=spde, group=i1.group, control.group=list(model='ar1'))
```

-

As we have Gaussian likelihood there is no approximation in the fitting process. The first step of the INLA algorithm is the optimization to find the mode of the 7 hyperparameters in the model. By choosing good starting values it will be needed less iteractions in this optimization process. Below, we define starting values for the hyperparameters in the internal scale considering the values used to simute the data

```
(theta.ini <- c(log(taue), ## likelihood log precision
                log(sqrt(8)/kappa[1]), ## log range 1
                log(sqrt(sigma2[1])), ## log stdev 1
                log((1+rho[1])/(1-rho[1])), ## inv.logit rho 1
                log(sqrt(8)/kappa[2]), ## log range 1
                log(sqrt(sigma2[2])), ## log stdev 1
                log((1+rho[2])/(1-rho[2]))))## inv.logit rho 2
```

```
## [1]  2.9957323 -1.2628643 -0.3465736  1.7346011 -1.4451859 -0.6931472
## [7]  1.0986123
```

-

This step takes around few minutes to fit, and with bigger `tolerance` value in `inla.control`, it will makes fewer posterior evaluations.

The integration step when using the CCD strategy, will integrates over 79 hyperparameter configurations, as we have 7 hyperparameters. However, in the following `inla()` call we avoid it.

Fitting the model considering the initial values defined above

```
(res <- inla(form, family='gaussian', data=inla.stack.data(stk.y),
             control.predictor=list(A=inla.stack.A(stk.y)),
             control.inla=list(int.strategy='eb'), ### no integration wr theta
             control.mode=list(theta=theta.ini, ### initial theta value
                               restart=TRUE)))$cpu
```

```
##  Pre-processing    Running inla Post-processing           Total
##       0.9579186     116.2767818       0.4555109     117.6902113
```

-

Summary of the μ_β:

```
round(cbind(true=mu.beta, res$summary.fix), 4)
```

```
##      true    mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## mu1    -5 -4.7481 0.1818    -5.1050  -4.7481    -4.3915 -4.7481   0
## h       1  0.9428 0.0531     0.8386   0.9428     1.0469  0.9428   0
```

-

Summary for the likelihood precision

```
round(c(true=taue, unlist(res$summary.hy[1,])), 3)
```

```
##       true       mean         sd 0.025quant   0.5quant 0.975quant
##     20.000     11.485      0.536     10.458     11.475     12.570
##       mode
##     11.459
```

-

We can see the posterior marginal distributions for the range and standard deviation for each spatio-temporal process in Figure 5.5.3.

```
par(mfrow=c(2, 3), mar=c(2.5,2.5,0.3,0.3), mgp=c(1.5,0.5,0))
for (j in 2:7) {
  plot(res$marginals.hy[[j]], type='l',
  xlab=names(res$marginals.hyperpar)[j], ylab='Density')
  abline(v=c(sqrt(8)/kappa[1], sigma2[1]^0.5, rho[1],
         sqrt(8)/kappa[2], sigma2[2]^0.5, rho[2])[j-1])
}
```

-

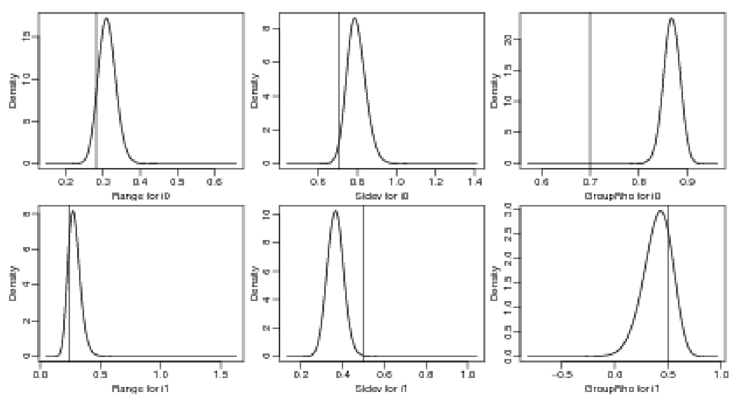


Figure 5.8: Posterior marginal distributions for the hyperparameters of the spacetime fields.

We can have a look over the posterior mean of the dynamic coefficients. We compute the correlation between the simulated and the posterior mean ones by

```
c(beta0=cor(as.vector(beta0), drop(A0%*%res$summary.ran$i0$mean)),
  beta1=cor(as.vector(beta1),
  drop(A0%*%res$summary.ran$i1$mean))) ## using A0 to account only for the coeff.
```

```
##     beta0     beta1
## 0.9453004 0.6163373
```

-

5.6 Space-time point process: Burkitt example

In this example we show hot to fit a space-time point process using the `burkitt` dataset from the splancs R package. The R source for this file is available at http://www.math.ntnu.no/inla/r-_inla.org/tutorials/spde/R/spde-_tutorial-_burkitt.R

We use the `burkitt` data set from the splancs package.

```
data('burkitt', package='splancs')
t(sapply(burkitt[, 1:3], summary))
```

```
##   Min. 1st Qu. Median   Mean 3rd Qu. Max.
## x  255   269.0  282.5  286.3   300.2  335
## y  247   326.8  344.5  338.8   362.0  399
## t  413  2412.0 3704.0 3530.0  4700.0 5775
```

-

The folowing commands shows the time when each event occurred, Figure 5.6.

```
n <- nrow(burkitt)
par(mfrow=c(1,1), mar=c(1.5,.1,.1,.1), mgp=c(2,0.7,0))
plot(burkitt$t, rep(1,n), type='h', ylim=0:1, axes=FALSE, xlab='', ylab='')
box(); axis(1)
```

-

We have to define a set of knots over time in order to fit SPDE spatio temporal model. It is then used to built a temporal mesh

```
k <- 6
tknots <- seq(min(burkitt$t), max(burkitt$t), length=k)
abline(v=tknots, lwd=4, col=4) ## add to plot
mesh.t <- inla.mesh.1d(tknots)
```

-

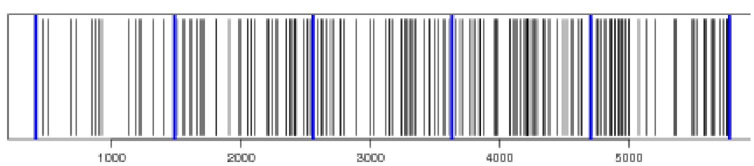


Figure 5.9: Time when each event occurred (black) and knots used for inference (blue).

The spatial mesh can be done using the polygon of the region as a boundary. We can convert the domain polygon into a `SpatialPolygons` class with

```
domainSP <- SpatialPolygons(list(Polygons(
         list(Polygon(burbdy)), '0')))
```

-

and the use it as a boundary

```
mesh.s <- inla.mesh.2d(burpts, boundary=inla.sp2segment(domainSP),
                   max.edge=c(10, 25), cutoff=3) ### just a crude mesh
```

-

Defining the SPDE model considering the PC-prior derived in [Fuglstad et al., 2017] for the model parameters as the practical range, $\sqrt{8\nu}/\kappa$, and the marginal standard deviation.

```
spde <- inla.spde2.pcmatern(
 mesh=mesh.s, alpha=2, ### mesh and smoothness parameter
 prior.range=c(0.05, 0.01), ### P(practic.range<0.05)=0.01
 prior.sigma=c(1, 0.01)) ### P(sigma>1)=0.01
m <- spde$n.spde
```

-

The spatio temporal projector matrix is made considering both spatial and temporal locations and both spatial and temporal meshes.

```
dim(Ast <- inla.spde.make.A(mesh=mesh.s, loc=burpts, n.group=length(mesh.t$n),
                            group=burkitt$t, group.mesh=mesh.t))
```

```
## [1]  188 3234
```

-

Internally `inla.spde.make.A` function makes a row Kronecker product (see `inla.row.kron`) between the spatial projector and the group (temporal in our case) projector. This matrix has number of columns equals to the number of nodes in the mesh times the number of groups.

The index set is made considering the group feature:

```
idx <- inla.spde.make.index('s', spde$n.spde, n.group=mesh.t$n)
```

-

The data stack can be made considering the ideas for the purerly spatial model. So, we do need to consider the expected number of cases at the 1) integration points and 2) data locations. For the integration points it is the spacetime volume computed for each mesh node and time knot, considering the spatial area of the dual mesh polygons, as in Chapter [4](), times the the length of the time window at each time point. For the data locations it is zero as for a point the expectation is zero, in accord to the likelihood approximation proposed by [[Simpson et al., 2016]()].

The dual mesh is extracted considering the function `inla.mesh.dual()`, available in http://www.math.ntnu.no/inla/r-_inla.org/tutorials/spde/R/spde-_tutorial-_functions.R

```
source('R/spde-tutorial-functions.R')
dmesh <- inla.mesh.dual(mesh.s)
```

-

Them, we compute the intersection with each polygon from the mesh dual using the functions `gIntersection()` from the **rgeos** package (show the sum of the intersection polygons areas):

```
library(rgeos)
sum(w <- sapply(1:length(dmesh), function(i) {
  if (gIntersects(dmesh[i,], domainSP))
    return(gArea(gIntersection(dmesh[i,], domainSP)))

  else return(0)
}))
```

```
## [1] 11035.01
```

-

We can see that it sum up the same as the domain area:

```
gArea(domainSP)
```

```
## [1] 11035.01
```

-

The spatio temporal volume is the product of these values and the time window length of each time knot.

```
st.vol <- rep(w, k) * rep(diag(inla.mesh.fem(mesh.t)$c0), m)
```

-

The data stack is built using

```
y <- rep(0:1, c(k * m, n))
expected <- c(st.vol, rep(0, n))
stk <- inla.stack(data=list(y=y, expect=expected),
  A=list(rBind(Diagonal(n=k*m), Ast), 1),
  effects=list(idx, list(a0=rep(1, k*m + n))))
```

-

Model fitting (using the cruder approximation: 'gaussian')

```
pcrho <- list(prior='pccor1', param=c(0.7, 0.7))
form <- y ~ 0 + a0 +
f(s, model=spde, group=s.group,
control.group=list(model='ar1', hyper=list(theta=pcrho)))
burk.res <- inla(form, family='poisson',
data=inla.stack.data(stk), E=expect,
control.predictor=list(A=inla.stack.A(stk)),
control.inla=list(strategy='gaussian'))
```

-

The exponential of the intercept plus the random effect at each spacetime integration point is the relative risk at each these points. This relative risk times the spacetime volume will give the expected number of points at each these spacetime locations. Summing it will approaches the number of observations:

```
eta.at.integration.points <- burk.res$summary.fix[1,1] + burk.res$summary.ran$s$mean
c(n=n, 'E(n)'=sum(st.vol*exp(eta.at.integration.points)))
```

```
##        n     E(n)
## 188.0000 187.9949
```

-

We can plot the posterior marginal distributions for the intercept and parameters, in Figure 5.6, with

```
par(mfrow=c(2,2), mar=c(3,3,1,1), mgp=c(1.7,0.7,0))
plot(burk.res$marginals.fix[[1]], type='l', xlab='Intercept')
plot(burk.res$marginals.hy[[1]], type='l',
xlim=c(0, 10), xlab='Practical range')
plot(burk.res$marginals.hy[[2]], type='l',
xlim=c(0, 3), xlab='Standard deviation')
plot(burk.res$marginals.hy[[3]], type='l',
xlim=c(0, 1), xlab='time correlation')
```

-

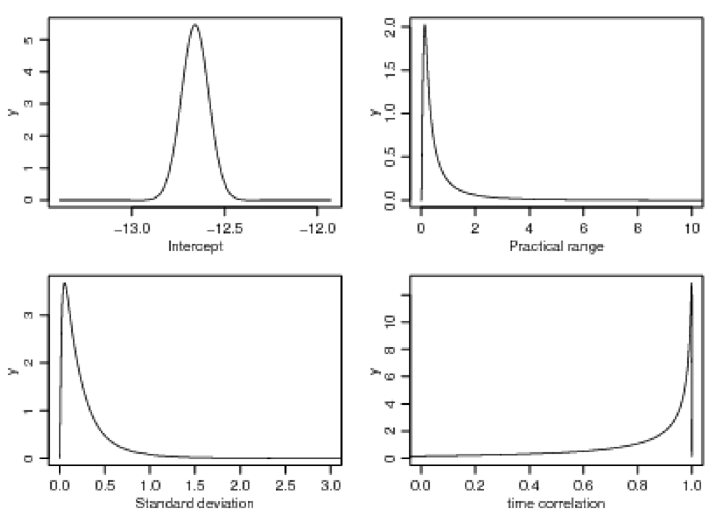


Figure 5.10: Intercept and Random Field parameters posterior marginal distributions.

The projection over a grid for each time knot can be done with

```
r0 <- diff(range(burbdy[,1]))/diff(range(burbdy[,2]))
prj <- inla.mesh.projector(mesh.s, xlim=range(burbdy[,1]),
```

```
ylim=range(burbdy[,2]), dims=c(100, 100/r0))
ov <- over(SpatialPoints(prj$lattice$loc), domainSP)
m.prj <- lapply(1:k, function(j) {
r <- inla.mesh.project(prj, burk.res$summary.ran$s$mean[1:m+(j-1)*m])
r[is.na(ov)] <- NA;   return(r)
})
```

-

The fitted latent field at each time knot is in Figure 5.6, produced with the code below. It can also be done for the standard deviation.

```
igr <- apply(abs(outer(burkitt$t, mesh.t$loc, '-')), 1, which.min)
zlm <- range(unlist(m.prj), na.rm=TRUE)
par(mfrow=c(2,3), mar=c(0,0,0,0))
for (j in 1:k) {
image(x=prj$x, y=prj$y, z=m.prj[[j]], asp=1,
xlab='', zlim=zlm, axes=FALSE, col=tim.colors(64))
points(burkitt[igr==j, 1:2], pch=19)
}; image.plot(legend.only=TRUE, zlim=zlm, legend.mar=5)
```

-

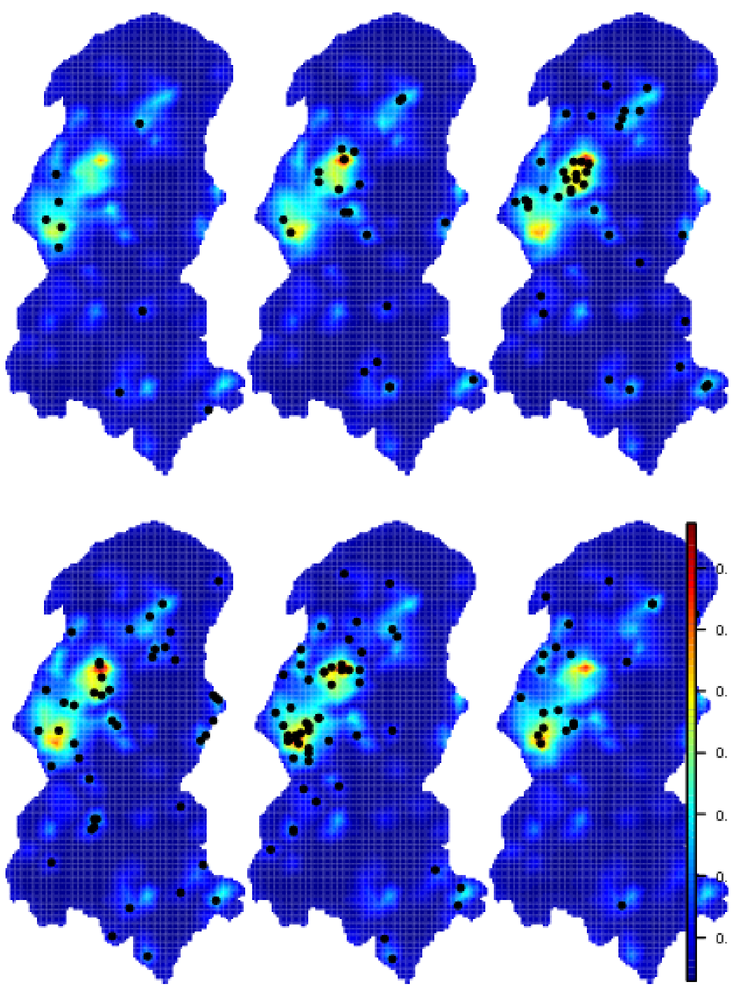


Figure 5.11: Fitted latent field at each time knot overlayed by the points closer in time.

5.7 Large point process data set

In this chapter we show how an approach to fit a spatio temporal log-Cox point process model for a large data sets. We are going to drawn samples from a separable space time intensity function. The R source for this file is available at http://www.math.ntnu.no/inla/r-_inla.org/tutorials/spde/R/spde-_tutorial-_stpp.R

First we define the spatial domain as follows

```
x0 <- seq(0, 4*pi, length=15)
domain <- data.frame(x=c(x0, rev(x0), 0))
domain$y <- c(sin(x0/2)-2, sin(rev(x0/2))+2, sin(0)-2)
```

-

and convert it into the `SpatialPolygons` class

```
library(sp)
domainSP <- SpatialPolygons(list(Polygons(list(Polygon(domain)), '0')))
```

-

We choose to sample a dataset using the **lgcp**, [Taylor et al., 2013], package as follows

```
library(lgcp)
ndays <- 15
n <- (xyt <- lgcpSim(
owin=spatstat:::owin(poly=domain), tlim=c(0,ndays),
model.parameters=lgcppars(1,0.5,0.1,0,0.5), cellwidth=0.1,
spatial.covmodel='matern', covpars=c(nu=1)))$n
```

-

In order to fit the model, we do need to define a discretization over space and over time. For the time domain, we define a temporal mesh based on a number of time knots:

```
k <- 7; tmesh <- inla.mesh.1d(seq(0, ndays, length=k))
```

-

The spatial mesh is defined using the domain polygon:

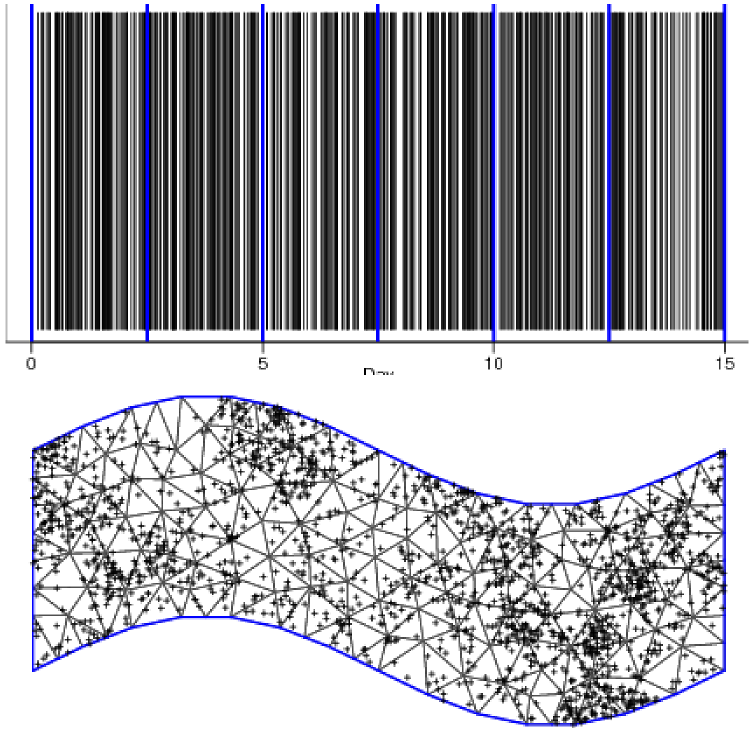
```
smesh <- inla.mesh.2d(boundary=inla.sp2segment(domainSP),
                      max.edge=1, cutoff=0.3)
```

-

We can have a look in Figure 5.7 to see a plot of a sample of the data over time, the time knots and over space and the spatial mesh as well with the commands below

```
par(mfrow=c(2,1), mar=c(1.5,0,0,0), mgp=c(1,0.5,0))
plot(sample(xyt$t,500), rep(1,500), type='h', ylim=0:1,
     xlab='Day', ylab='', axes=FALSE); box(); axis(1)
abline(v=tmesh$loc, col=4, lwd=3)
par(mar=c(0,0,0,0))
plot(smesh, asp=1, main='')
points(xyt$x, xyt$y, cex=0.5, pch=3)
```

-



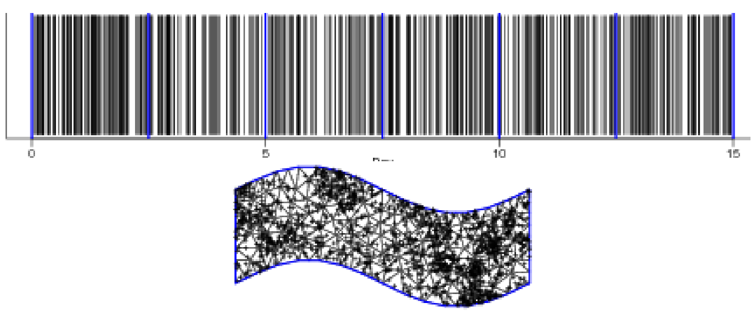


Figure 5.12: Time for a sample of the events (black), time knots (blue) in the upper plot. Spatial locations of a sample on the spatial domain (botton plot).

5.7.1 Space-time aggregation

For large datasets it can be computationally demanding to fit the model. The problem is because the dimention of the model would be $n + m*k$, where n is the number of data points, m is the number of nodes in the mesh, k is the number of time knots. In this section we choose to aggregate the data in a way that we have a problem with dimention $2 *m*k$. So, this approach really makes sence when $n >> m * k$.

We choose to aggregate the data in accord to the integration points to make the fitting process easier. We also consider the dual mesh polygons, as shown in Chapter 4.

So, firt we find the Voronoi polygons for the mesh nodes

```
library(deldir)
dd <- deldir(smesh$loc[,1], smesh$loc[,2])
tiles <- tile.list(dd)
```

-

Convert it into `SpatialPolygons`:

```
polys <- SpatialPolygons(lapply(1:length(tiles), function(i)
    { p <- cbind(tiles[[i]]$x, tiles[[i]]$y)
      n <- nrow(p)
```

```
Polygons(list(Polygon(p[c(1:n, 1),])), i)
}))
```

-

Find to which polygon belongs each data point:

```
area <- factor(over(SpatialPoints(cbind(xyt$x, xyt$y)),
                    polys), levels=1:length(polys))
```

-

Fint to which part of the time mesh belongs each data point:

```
t.breaks <- sort(c(tmesh$loc[c(1,k)],
                   tmesh$loc[2:k-1]/2 + tmesh$loc[2:k]/2))
table(time <- factor(findInterval(xyt$t, t.breaks),
                     levels=1:(length(t.breaks)-1)))
```

```
##
##   1   2   3   4   5   6   7
## 183 327 260 282 321 288 146
```

-

Use these both identification index sets to aggregate the data

```
agg.dat <- as.data.frame(table(area, time))
for(j in 1:2) ### set time and area as integer
agg.dat[[j]] <- as.integer(as.character(agg.dat[[j]]))
str(agg.dat)
```

```
## 'data.frame': 1064 obs. of  3 variables:
##  $ area: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ time: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Freq: int  1 2 0 0 0 0 0 0 0 3 ...
```

-

We need to define the expected number of cases (at least) proportional to the area of the Polygons times the width length of the time knots. Compute the intersection area of each polygon with the domain (show the sum).

```
library(rgeos)
sum(w.areas <- sapply(1:length(tiles), function(i)
  { p <- cbind(tiles[[i]]$x, tiles[[i]]$y)
    n <- nrow(p)
pl <- SpatialPolygons(list(Polygons(list(Polygon(p[c(1:n, 1),])), i)))

if (gIntersects(pl, domainSP))
return(gArea(gIntersection(pl, domainSP)))
else return(0)
}))
```

```
## [1] 50.26548
```

-

A summary of the polygons area is

```
summary(w.areas)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.06293 0.21780 0.35040 0.33070 0.41160 0.69310
```

-

and the area of the spatial domain is

```
gArea(domainSP)
```

```
## [1] 50.26548
```

-

The time length (domain) is 365 and the width of each knot is

```
(w.t <- diag(inla.mesh.fem(tmesh)$c0))
```

```
## [1] 1.25 2.50 2.50 2.50 2.50 2.50 1.25
```

-

where the knots at boundary are with less width than the internal ones.

Since the intensity function is the number of cases per volumn unit, with n cases the intensity varies around the average number of cases (intensity) by unit volumn

```
(i0 <- n / (gArea(domainSP) * diff(range(tmesh$loc))))
```

```
## [1] 2.396608
```

-

and this value is related to an intercept in the model we fit below. The space-time volumn (area unit per time unit) at each polygon and time knot is

```
summary(e0 <- w.areas[agg.dat$area] * (w.t[agg.dat$time]))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.07866 0.45690 0.64780 0.70860 0.97130 1.73300
```

-

5.7.2 Model fit

The projector matrix, SPDE model object and the space-time index set definition:

```
A.st <- inla.spde.make.A(smesh, smesh$loc[agg.dat$area,],
                         group=agg.dat$time, mesh.group=tmesh)
spde <- inla.spde2.matern(smesh)
idx <- inla.spde.make.index('s', spde$n.spde, n.group=k)
```

-

Defining the data stack

```
stk <- inla.stack(data=list(y=agg.dat$Freq, exposure=e0),
                  A=list(A.st, 1),
                  effects=list(idx,
                  list(b0=rep(1, nrow(agg.dat)))))
```

-

the formula

```
formula <- y ~ 0 + b0 +
f(s, model=spde, group=s.group, control.group=list(model='ar1'))
```

-

and fitting the model

```
res <- inla(formula, family='poisson',
      data=inla.stack.data(stk), E=exposure,
```

```
control.predictor=list(A=inla.stack.A(stk)),
  control.inla=list(strategy='gaussian'))
```

-

The log of the average intensity and the intercept summary:

```
round(cbind(true=log(i0), res$summary.fixed),4)
##      true  mean     sd 0.025quant 0.5quant 0.975quant  mode kld
## b0 0.8741 0.702 0.1553     0.3933   0.7019     1.0107 0.702   0
```

-

The expected number of cases at each integration point can be used to compute the total expected number of cases

```
eta.i <- res$summary.fix[1,1] + res$summary.ran$s$mean
c(n=xyt$n, 'E(n)'=sum(rep(w.areas, k)*rep(w.t, each=smesh$n)*exp(eta.i)))
```

```
##        n     E(n)
## 1807.000 1805.903
```

-

The spatial surface at each time knot can be computed by

```
r0 <- diff(range(domain[,1]))/diff(range(domain[,2]))
prj <- inla.mesh.projector(smesh, xlim=bbox(domainSP)[1,],
                 ylim=bbox(domainSP)[2,], dims=c(r0*200, 200))
g.no.in <- is.na(over(SpatialPoints(prj$lattice$loc), domainSP))
t.mean <- lapply(1:k, function(j) {
z <- inla.mesh.project(prj, res$summary.ran$s$mean[idx$s.group==j])
z[g.no.in] <- NA
return(z)
})
```

-

and is visualized in Figure 5.7.2 is visualized by

```
zlims <- range(unlist(t.mean), na.rm=TRUE)
library(fields)
par(mfrow=c(4,2), mar=c(0.1,0.1,0.1,0.1))
for (j in 1:k) {
image(prj$x, prj$y, t.mean[[j]],
axes=FALSE, zlim=zlims, col=tim.colors(30))
points(xyt$x[time==j], xyt$y[time==j], cex=0.1)
}
image.plot(prj$x, prj$y, t.mean[[j]]+1e9, axes=FALSE, zlim=zlims, xlab='',
  legend.mar=10, legend.width=5, col=tim.colors(30), horizontal=T)
```

-

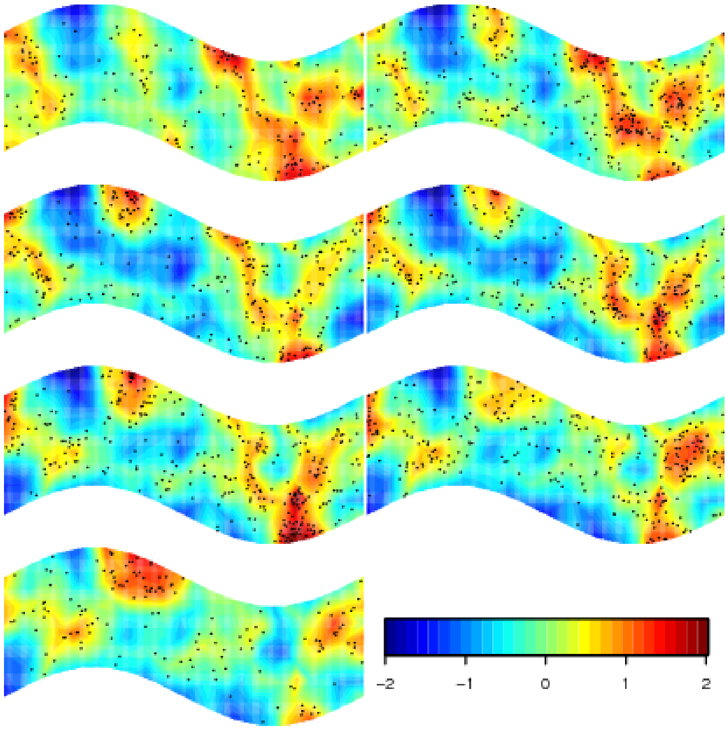


Figure 5.13: Spatial surface fitted at each time knot overlayed by the point pattern formed by the points nearest to each time knot.