The R-INLA tutorial on SPDE models

This work was further developed into the
SPDE book, see http://www.r-inla.org/spde-book

Elias T. Krainski, Finn Lindgren, Daniel Simpson and Håvard Rue

July 22, 2019

Acknowledgments To Sarah Gallup and Helen Sofaer for valuable English review in the text. To several people who brought cool problems to the discussion forum http://www.r-inla.org/comments-1 and to me individually.

Updates

```
2019 Jul 22: rerun to fix some updated code
2019 May 20: Update links and point to spde-book
2017 Oct 05: English review (thanks to SG)
2017 Mar 01: revision and new organization
2017 Feb 15: started a major revision/organization
2017 Jan 23: compile to html and small fixes
2016 May 17: fix names in copy lin. pred. example (Thx to MC) and improve text
2016 May 09: copy linear predictor example
2016 Mar 22: tiny fix of survival ex.: cor(log(sd), ...). Thx HR
2016 Feb 26: space-time (small fix) and non-stationary (s. improv.)
2016 Feb 03: small changes in dynamic and space-time simulation
2016 Feb 01: tiny fix in the rainfall example model description
2015 Dec 30: measurement error example added
2015 Sep 25: small fixes, improve dynamic example
2015 Sep 24: 1) include hyperref package. 2) improve dynamic example
2015 Sep 23: small fix on likelihood equations,
  space-time with continuous time and dynamic regression model example
2015 Aug 31: aproach for large spacetime point process
2015 Aug 26: fix the spacetime point process example
2015 Jul 17: small fixes and interpolate lin. pred. samples (Non-Gaus.)
2015 May 25: include space-time coregionalization model and tiny fix
2015 May 7: English review in Chapters 1 and 2 (thanks to HS) and
  space-time point process, space time lowering dimension and survival
2014 March 15: log-Cox example: fix weights, add covariate example
2013 December 21:
 * fix several missprints
 * add details on: likelihood, semicontinuous and spacetime examples
2013 October 08:
 * Finn's suggestions on two likelihood examples
2013 October 02:
 * mesh news: inla.mesh.2d, inla.noncovexhull, SpatialPolygons
 * toy-example improved (maybe more clear...)
 * new chapters: likelihood through SPDE, point process,
  preferential sampling, spatio temporal, data cloning
2013 March 21:
 * non-stationary example and joint covariate modelling
2013 March 01:
 * first draft: introduction, toy example, rainfall on Parana State
```

Abstract

This tutorial will show you how to fit models that contains at least one effect specified from an SPDE using the 'R-INLA'. Up to now, it can an SPDE based model can be applied to model random effects over continuous one- or two- dimensional domains. However, the theory works for higher dimensional cases. The usual application is data whose geographical location is explicitly considered in the analysis. This tutorial explores 'R-INLA' functionalities by using examples. It starts with simple models and increases in complexity.

In Chapter includes a section introducing the random field models and the Matérn class. We illustrate some features of this class in figures. The we introduce the main results in [] intuitively linking to images of the matrices involved being computed for a illustrative small case with few spatial locations. We show how to fit a geostatistical model for a simulated data, the toy example, covering from the mesh building, model definition, data preparation, showing results, doing predictions and considering results from different meshes. We also show how to build a mesh considering non-convex domains, spatial polygons objects and domains with holes or physical boundaries.

In Chapter we consider three examples. We consider the daily average rainfall from rainfall collected at 616 gauge stations in the Paraná state in Brasil over year 2011. For this data we show a detailed analysis including code to compute geographical covariates, smoothed regression an prediction. The second example in this Chapter consider survival analysis for the Leukaemia dataset, analysed in []. We show how to consider the parametric Weibull case and also the non-parametric Cox proportional hazard case. For this case we have the implementation internally considers a new structure of the data in order to perform a Poisson regression. The last example in this Chapter considers simulated data to illustrate the approach of modeling the SPDE model parameters by a regression which is the case of having covariates in the covariance, proposed in [].

In Chapter we have a collection of examples were copy random fields to model two or more outcomes jointly. It includes a measurement error model in order to account for spatially structured measurement error in a covariate. A coregionalization model consider the case for three outcomes were the fist outcome is in the linear predictor for the second one and both are in the predictor for the third outcome, as proposed in []. An example considering copying a part or the entire linear predictor from one outcome in a linear predictor to another one ends this chapter. It shows a slight different way from the coregionalization model to jointly model three outcomes.

The log Cox point process model is considered in Chapter . In this case we show how to fit a Log-cox point process using the direct approximation for the likelihood as proposed in []. We also take the opportunity to show how to consider the joint modeling of the process and the locations, under the preferential sampling as proposed in [].

Finally, Chapter presents several cases to example analysis of space-time data. We start by an example having discrete time domain as in []. We extend

this example considering the time as continuous by considering time knots along with temporal function basis functions for projection. We also extend the coregionalization example for the space-time in this Chapter. The space-time model is also applied for modeling regression coefficients in a dynamic regression example having the regression coefficients varying over space-time. We consider the space-time version of the log-Cox point process model for a dataset and also illustrates an approach to deal whit the case of having a large space-time point process data.

Since this tutorial is more a collection of examples, one should start with the tutorial marked as **Read this first!** at the tutorials link in the R-INLA web page, `http://www.r-inla.org`, more precisely at `http://www.r-inla.org/examples/tutorials/spde-tutorial-from-jss`. If you are in a rush to fit a simple geostatistical model, we made a short tutorial without the details as a vignette in the **INLA**. Thus one can have it just typing `vignette(SPDEhowto)` for a two dimensional example or `vignette(SPDE1d)` for a one dimensional example. We built a Shiny application to help one to understand the mesh building. It depends on the **shiny** package. This application opens by typing `demo(mesh2d)`.

This content is part of the book available at `http://www.r-inla.org/spde-book`, whose Gitbook version is freely available along all the code and datasets.

Contents

Also, we join both stacks by

```
stk5.jp <- inla.stack(stk5, stk5p.rf)
```

and fit the model again with the full stack setting `compute=TRUE` on `control.predictor`

```
res5p <- inla(resp ~ 0 + m + f(i, model=spde5),
              data=inla.stack.data(stk5.jp),
              control.predictor=list(A=inla.stack.A(stk5.jp), compute=TRUE))
```

To access the posterior marginal distribution of the random field at the target locations, we extract the index from the full stack using the adequate `tag`.

```
(indd5p <- inla.stack.index(stk5.jp, tag='prd5r')$data)

## [1] 201 202 203
```

The summary of the posterior distributions of the random field on the target locations is

```
round(res5p$summary.linear.pred[indd5p,], 4)

##                     mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## APredictor.201  0.1226 0.6354    -1.1047   0.1122     1.4117  0.0936   0
## APredictor.202  2.9730 0.8163     1.3825   2.9677     4.5943  2.9575   0
## APredictor.203 -2.7016 1.0888    -4.8496  -2.6988    -0.5694 -2.6934   0
```

that includes the posterior mean, standard deviation, quantiles and mode.

Because it is a full bayesian analysis, we also we have the marginal distributions. We extract the marginals posterior distributions with

```
marg3 <- res5p$marginals.linear[indd5p]
```

and get the 95% HPD interval for the random field at the second target location by

```
inla.hpdmarginal(0.95, marg3[[2]])

##                  low     high
## level:0.95 1.370391 4.580516
```

and see that around the point (0.5,0.5) the random field has positive values, see Figure .

After the estimation process

If we need just the prediction we can do the prediction after the estimation process with a very small computational cost. It is just a matrix operation in way that we just project the posterior mean of the the random field on mesh nodes to target locations, using the correspondent projector matrix.

So, we 'project' the posterior mean of the latend random field to the target locations by

```
drop(A5pts3%*%res5$summary.random$i$mean)

## [1]  0.1222806  2.9728843 -2.7014696
```

or using the `inla.mesh.projector()` function

```
inla.mesh.project(inla.mesh.projector(mesh5, loc=pts3),
                  res5$summary.random$i$mean)

## [1]  0.1222806  2.9728843 -2.7014696
```

and see that for the mean we have similar values than those on previous subsection. Also, we can get the standard deviation

```
drop(A5pts3%*%res5$summary.random$i$sd)

## [1] 0.7389992 0.9920801 1.2677818
```

and we have a little difference.

```
sqrt(drop((A5pts3^2)%*%(res5$summary.random$i$sd^2)))

## [1] 0.4894069 0.6144520 0.8630550
```

Projection on a grid

The approach by the projection of the posterior mean random field is computationaly cheap. So, it can be used to get the map of the random field on a fine grid. The `inla.mesh.projector()` function get the projector matrix automatically for a grid of points over a square that contains the mesh.

To get projection on a grid at the domain $(0, 1) \times (0, 1)$ we just inform these limits

```
pgrid0 <- inla.mesh.projector(mesh5, xlim=0:1, ylim=0:1, dims=c(101,101))
```

and we project the posterior mean and the posterior standard deviation on the both grid with

```
prd0.m <- inla.mesh.project(pgrid0,  res5$summary.ran$i$mean)
prd0.s <- inla.mesh.project(pgrid0,  res5$summary.ran$i$s)
```

We visualize this values projected on the grid on Figure .

1.2.6 Prediction of the response

Another commom result that we want on spatially continuous modelling is the prediction of the response on a target locations that we don't have data observed. In similar way that on past subsection, it is possible to find the marginal distribution or to make a projection of some functional of the response.

By the posterior distribution

In this case, we want to define a adequate predictor of the response and build the model again. This is similar to the stack to predict the random field, but here we add the intercept on the list of predictor matrix and on the list of effects

```
stk5.presp <- inla.stack(data=list(resp=NA), A=list(A5pts3,1),
                         effects=list(i=1:spde5$n.spde, m=rep(1,3)),
                         tag='prd5.resp')
```

and join with the data stack to build the model again

```
stk5.full <- inla.stack(stk5, stk5.presp)
r5presp <- inla(resp ~ 0 + m + f(i, model=spde5),
                data=inla.stack.data(stk5.full),
                control.predictor=list(A=inla.stack.A(stk5.full), compute=TRUE))
```

We find the index of the predictor that corresponds the predicted values of the response on the target locations. We extract the index from the full stack by

```
(indd3r <- inla.stack.index(stk5.full, 'prd5.resp')$data)

## [1] 201 202 203
```

To get the summary of the posterior distributions of the response on target locations we do

```
round(r5presp$summary.fitted.values[indd3r,], 3)

##                           mean    sd 0.025quant 0.5quant 0.975quant   mode
## fitted.APredictor.201  9.681 0.344      9.008    9.680     10.358  9.679
## fitted.APredictor.202 12.532 0.625     11.308   12.531     13.762 12.529
## fitted.APredictor.203  6.857 0.976      4.948    6.853      8.788  6.845
```

Also, we extract the marginals posterior distributions with

```
marg3r <- r5presp$marginals.fitted.values[indd3r]
```

and get the 95% HPD interval for the response at second target location by

```
inla.hpdmarginal(0.95, marg3r[[2]])

##                  low     high
## level:0.95 11.30451 13.75794
```

and see that around the point (0.5,0.5) we have the values of the response significantly larger than β_0, see Figure .

By sum of linear predictor components

A computational cheap approach is to (naively) sum the projected posterior mean to the regression term. In this toy example we just sum the posterior mean of the intercept to the posterior mean of the random field to get the posterior mean of the response.

If there are covariates, the prediction also can be made in similar way, see . That approach can be used here considering just the intercept

```
res5$summary.fix[1,1] + drop(A5pts3%*%res5$summary.random$i$mean)

## [1]  9.681211 12.531815  6.857461
```

For the standard error, we need to take into account the error of the covariate values and regression coefficients.

```
summary(rvar <- res5$summary.random$i$sd^2)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3385  0.7888  1.1684  1.9654  2.1542  8.2618

sqrt(1^2+res5$summary.fix[1,2]^2 + drop(A5pts3%*%rvar))

## [1] 1.353252 1.509747 1.700649
```

Response on a grid

The computation of all marginal posterior distributions on a grid is computationally expensive. But, we usually not uses the marginal distributions. We usually uses just the mean and standard deviation. So, we don't need the storage of all the marginal distributions! Also, we don't need the quantiles of the marginal distributions.

On the code below, we build the model again but we disable the storage of the marginal posterior distributions to random effects and to posterior predictor values. Also, we disable the computation of the quantiles. Only the mean and standard defiation are stored.

We use the projector matrix on the projector object that we use to project the posterior mean on the grid

```
stkgrid <- inla.stack(data=list(resp=NA), A=list(pgrid0$proj$A,1),
                     effects=list(i=1:spde5$n.spde,
                       m=rep(1,101*101)), tag='prd.gr')
stk.all <- inla.stack(stk5, stkgrid)
res5g <- inla(resp ~ 0 + m + f(i, model=spde5),
              data=inla.stack.data(stk.all),
              control.predictor=list(A=inla.stack.A(stk.all),
                compute=TRUE), quantiles=NULL,
              control.results=list(return.marginals.random=FALSE,
                return.marginals.predictor=FALSE))
res5g$cpu

##        Pre    Running       Post      Total
## 0.38548446 6.41950345 0.09945011 6.90443802
```

We get the indexes

```
igr <- inla.stack.index(stk.all, 'prd.gr')$data
```

and use it to visualize, together the prediction of the random field on previous subsection, on Figure with the commands bellow

```
library(gridExtra)
grid.arrange(levelplot(prd0.m, col.regions=topo.colors(99), main='latent field mean',
                       xlab='', ylab='', scales=list(draw=FALSE)),
             levelplot(matrix(res5g$summary.fitt[igr,1], 101),
                       xlab='', ylab='', main='response mean',
                       col.regions=topo.colors(99), scales=list(draw=FALSE)),
             levelplot(prd0.s, col.regions=topo.colors(99), main='latent field SD',
                       xlab='', ylab='', scales=list(draw=FALSE)),
             levelplot(matrix(res5g$summary.fitt[igr,2], 101),
                       xlab='', ylab='', main='response SD',
```