SPDE example with multiple kind of outcomes:

Joint modeling with `mdata`, `inla.surv` and counts with exposure.

Elias T Krainski

2023, March

Abstract

The **INLA** package supports different kinds of data likelihoods, including the `inla.surv`, which is a generalization of the `Surv` from the **survival** package, and `mdata`. It also support joint modeling different kind of outcomes. In this vignette we illustrate the case when we have three outcomes, one of each type, and model them jointly.

Introduction

Usually correlations between outcomes are modeled by sharing common model terms in the linear predictor for each one. These terms can be covariates, random effects or sums of these. It is possible to have cases when each likelihood for an observation unit depend on more than one quantity, for example, one likelihood is to model the survival time, which has the additional information about censoring, and other likelihood is written as a function of several quantities, as we will see in the example.

In this vignette we consider a joint model for three outcomes, simulate data from this model and illustrate how to fit the joint model accounting for shared terms. The linear predictor for each outcome includes covariates and a spatial random effect modeled using the SPDE approach.

We consider a scenario where the first outcome is known at an aggregated level, in a setting were we model it considering a kind of likelihood designed in **INLA** to account for all the information available with respect to the aggregation. The second outcome is considered to be observed censored, so that we have to consider the observed value and the censoring information. The third outcome is just the usual kind of data: one scalar per data unit.

The spatial domain

The SPDE models available in **INLA** could be applied for processes in $\mathbb{R}^d$ or $\mathbb{S}^d$, for $d = 1, 2$, and the theory in Lindgren, Rue, and Lindström (2011) include higher dimentional domains. We consider a spatial domain, $D \in \mathbb{R}^2$ to keep the example simple.

Without loss off generality we consider D as a rectangle defined as

```
bb <- rbind(c(0, 20), c(0, 12))
```

Next, we store the length of each side and its average for later use

```
rxy <- apply(bb, 1, diff)
b <- mean(rxy)
```

We sample a set of n locations, $l_1, ..., l_n$, in this domain, $l_i \in D$, completely at random as follows:

```
n <- 500
loc <- cbind(
    runif(n, bb[1, 1], bb[1, 2]),
    runif(n, bb[2, 1], bb[2, 2]))
```

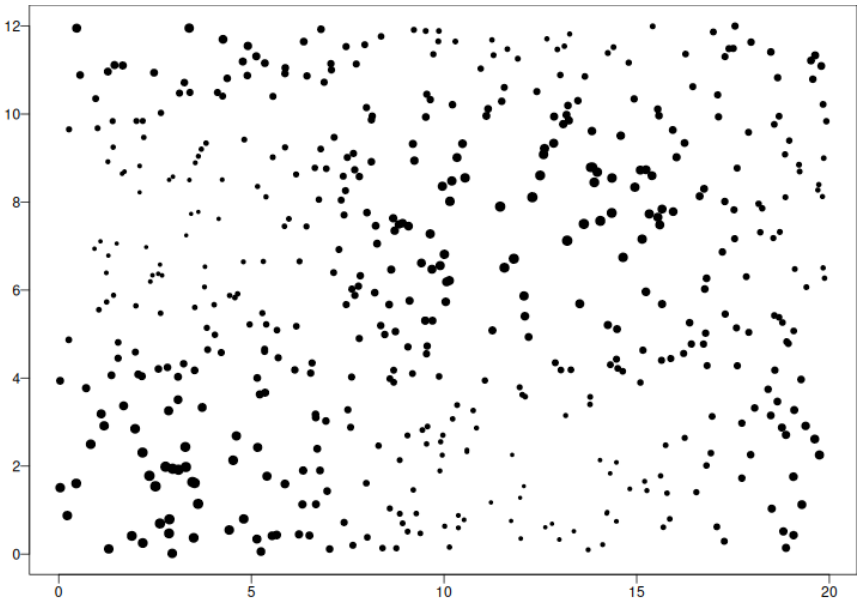
A spatial field

Instead of simulating from a model to play as a spatial smooth term, we define a smooth function on this domain as follows

```
sfn <- function(a, b)
    sin(a + b) + cos(a - b)
```

This function is evaluated at the n locations and visualized with

```
u <- sfn(2 * pi * loc[, 1] / rxy[1],
         2 * pi * loc[, 2] / rxy[2])
summary(u)
#>      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
#> -1.981431 -0.786834 -0.003074 -0.035908  0.574413  1.999551
par(mar = c(2, 2, 0, 0), mgp = c(1.5, 0.5, 0), las = 1)
plot(loc, pch = 19, cex = 0.5 + (2 + u)/4, xlab = "", ylab = "")
```



Spatial sub-domains

In order to define the first outcome, let us consider that the domain is being divided into a set of sub-regions, $r_1, r_2, ..., r_g$, so that $r_i \cap r_i = \oslash$ and $\cup_{i=1}^{g} r_j = D$.

For simplicity of coding, but without loss of generality, we assume that these small sub-regions to be pixels based in the following set of centers:

```
h <- 2 ## size of the side of each sub-region
group.ce <- as.matrix(expand.grid(
    seq(bb[1, 1] + h/2, bb[1, 2], h),
    seq(bb[2, 1] + h/2, bb[2, 2], h)))
(ng <- nrow(group.ce))
```

```
#> [1] 60
```

We can avoid the need of dealing with the general spatial objects and operations available in **R** and identify each location l_i to each sub-region r_j with an integer vector as

```
group.id <- ceiling(loc[,1]/h)+ceiling(rxy[1]/h) *
  (ceiling(loc[,2]/h)-1)
```

This will be used as a grouping index, so that each group is made of the observations falling in each of the pixels. We will visualize the locations with color depending on this grouping later.

Covariates

Let us consider a set of three covariates for each location as

```
xxx <- cbind(x1 = runif(n), x2 = runif(n), x3 = runif(n))
```

We average the covariates by group as we will consider that we have data at the group level to model the first outcome.

```
xxx.g <- aggregate(xxx, list(g = group.id), mean)
str(xxx.g)
#> 'data.frame':    60 obs. of  4 variables:
#>  $ g : num  1 2 3 4 5 6 7 8 9 10 ...
#>  $ x1: num  0.388 0.471 0.436 0.469 0.561 ...
#>  $ x2: num  0.357 0.585 0.593 0.466 0.436 ...
#>  $ x3: num  0.598 0.49 0.491 0.388 0.567 ...
```

Outcomes

We will consider three outcomes.

First outcome: aggregated Gaussian

We assume that one of the outcomes is a continuous random variable, to be considered as Gaussian, with unknown mean and variance, and a known scaling factor, s, such that

$$y_i \sim \mathrm{N}(\mu_i, s_i\sigma^2).$$

The first outcome being simulated considering that the expected value as $\eta_1 = \{\mu_1, ..., \mu_n\}$ as

$$\eta_1 = \mathbf{X}\beta_1 + \mathbf{u}.$$

This is being evaluated considering the covariate and random effect averaged per group:

```
betas1 <- c(5, 0, -3, 3)
u.g <- tapply(u, group.id, mean)
eta.y.g <- drop(cbind(1, as.matrix(xxx.g[, 2:ncol(xxx.g)])) %*% betas1) + u.g
```

We will simulate the scaling factor for each observation with

```
s <- rgamma(n, 7, 3)
summary(s)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.3769  1.6935  2.2326  2.3795  3.0328  5.4808
```

```
sigma.y <- 1
y <- rnorm(n, eta.y.g[group.id], sqrt(sigma.y/s))
summary(y)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   1.515   4.009   4.915   4.917   5.744   8.807
```

We simulated y_i for each location l_i. However, we consider that observations of this variable are available only at the aggregated level, at each of the sub-regions r_j. Furthermore, for each sub-region r_j, we know

- how many observations were taken in each sub-region, $n_j = \sum_{i=1}^{n} I(l_i \in r_j)$
- the weighted average $\overline{y}_j = (n_j)^{-1} \sum_{i,l_i \in r_j} s_i y_i$,
- the sum o the scaling factors, $m_j = \sum_{i,l_i \in r_j} s_i$,
- the half of the sum of the log of the scaling factors, $\frac{1}{2} \sum_{i,l_i \in r_j} \log(s_i)$,
- and the sample variance within each unit, defined as $v_j = \frac{1}{n_j} \sum_{i,l_i \in r_j}^{n} (s_i y_i^2 - \overline{y}_j^2)$

The function `inla.agaussian` computes these five statistics, given y_i and s_i for each group. We use this function to compute these statistics, as follows

```
library(INLA)
#> Loading required package: Matrix
#> Loading required package: foreach
#> Loading required package: parallel
#> Loading required package: sp
#> This is INLA_23.03.26 built 2023-03-26 19:04:09 UTC.
#>  - See www.r-inla.org/contact-us for how to get help.
#>  - To enable PARDISO sparse library; see inla.pardiso()
#>  - Save 290.9Mb of storage running 'inla.prune()'
agg <- lapply(1:ng, function(g) {
    ig <- which(group.id==g)
    if(length(ig)>0)
        return(inla.agaussian(y[ig], s[ig]))
    return(inla.mdata(list(NA, 0, 1, 1, NA))) ### a deal with missing
})
str(YY <- Reduce('rbind', lapply(agg, unlist))) ## five columns matrix
#>  num [1:60, 1:5] 0.163 0.355 0.332 0.286 0.144 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : chr [1:60] "init" "" "" "" ...
#>   ..$ : chr [1:5] "Y1" "Y2" "Y3" "Y4" ...
```

Second outcome: Survival

One can fit survival modes using **INLA** considering several available likelihoods. One is the `weibullsurv` and we will now consider that we have an outcome that follows a Weibull distribution, considering the following parametrization

$$f_W(w) = \alpha w^{\alpha-1} \lambda^{\alpha} \mathrm{e}^{-(\lambda w)^{\alpha}}.$$

We assume that $\lambda = \exp(\eta_2)$, where η_2 is the linear predictor and is defined as

$$\eta_2 = \mathbf{X}\beta + \gamma_1 \mathbf{u}$$

and we consider the following code to sample from a Weibull distribution

```
betas2 <- c(1, -1, 0, 1)
alpha <- 2
gamma1 <- 0.5
lambdaw <- exp(cbind(1, xxx) %*% betas2 + gamma1 * u)
```

```
we0 <- rweibull(n, shape = alpha, scale = 1/lambdaw)
summary(we0)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.01662 0.15374 0.29795 0.40216 0.50582 3.40239
```

However, suppose we do not observe the survival times for some of the observations.

```
summary(u.ev <- runif(n, 0.3, 1)) ## censoring factor
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.3019  0.5204  0.6867  0.6723  0.8290  0.9988
table(event <- rbinom(n, 1, 0.5)) ## censored (=0) or event (=1)
#>
#>   0   1
#> 238 262
summary(we <- ifelse(event == 1, we0, we0 * u.ev)) ## censored outcome
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.01337 0.11886 0.22933 0.34113 0.43561 3.26612
```

Third outcome: Count under exposure

We consider that we have different exposure at each location

```
summary(ee <- rgamma(n, 10, 2))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   1.267   3.894   4.807   5.008   5.905  11.055
```

We assume an outcome following a Poisson distribution with

$$o_i \sim \text{Poisson}(\lambda_i E_i)$$

where λ_i depends on the covariates and assuming that

$$\log \lambda = \eta_3 = \mathbf{X}\beta_e + \gamma_2 \mathbf{u}$$

This outcome is being simulated with

```
betas3 <- c(2, 1, -1, 0)
gamma2 <- -0.3
lambdap <- cbind(1, xxx) %*% betas3 + gamma2 * u
po <- rpois(n, exp(lambdap) * ee)
summary(po)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>       3      23      34      41      52     196
```

Data model setup

We now have to setup the objects in order to fit the model.

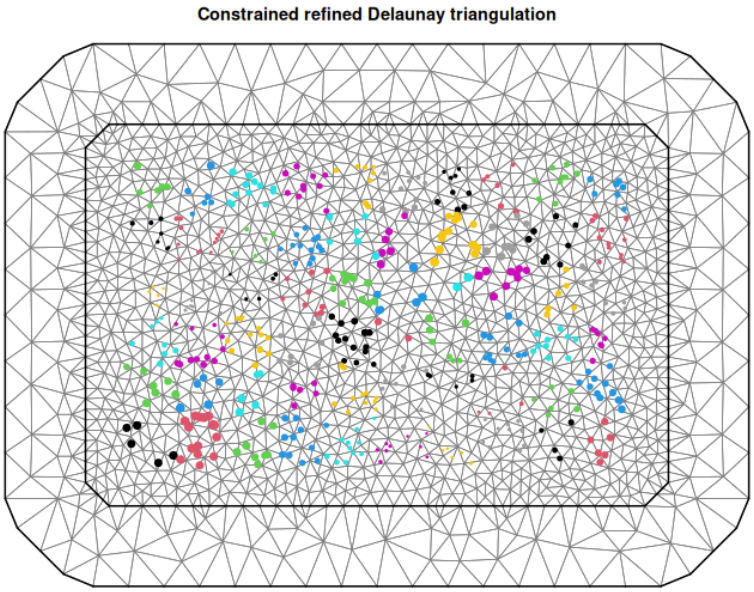
SPDE model setup

We have to define a mesh over the domain. We do it considering the following code

```
mesh <- inla.mesh.2d(
    loc.domain = matrix(bb[c(1,3,3,1,1, 2,2,4,4,2)], ncol = 2),
    max.edge = c(.05, .15) * b,
    offset = c(.1, .2) * b)
```

This mesh and the locations colored by the group defined for the first outcome can be visualized with

```
par(mar = c(0, 0, 1, 0), mgp = c(1.5, 0.5, 0))
plot(mesh, asp = 1, edge.color = gray(0.5))
points(loc, pch = 19, col = group.id, cex = 0.3 + (2 + u)/4)
```



The SPDE model is defined with

```
spde <- inla.spde2.pcmatern(
    mesh,
    prior.range=c(1, 0.1),
    prior.sigma=c(1, 0.1))
```

Linear predictor components

We will fit a joint model considering the linear predictor with terms as how it was simulated. The `inla.stack()` function in ment to me used when working with SPDE models. However, it also helps when considering multiple likelihood data as detailed bellow.

One way to deal with multiple likelihoods in **INLA** is defining `matrix` data for the outcome where each column is used for each likelihood. However, we do have more complex cases. Therefore, **the way** to deal with more complex multiple likelihood data is to use `list` in the left hand side of the `formula`. This is the key point to implement this example.

The right hand side of the formula includes each term in the linear predictor. When working with multiple likelihood, each term in each likelihood should be named uniquely. For example, if two likelihoods does includes the same covariate and each one with a different coefficient, each one has to have its own name. Therefore, we will pass the covariate `x1` as `x1y`, for the first outcome, `x1w` for the second outcome, and `x1p` for the third outcome. Similarly for the other ones, including the intercept.

In our example, the SPDE model is in the linear predictor for the first outcome, a copy from this random effect in in the linear predictor for the second outcome, and another copy of this term is in the linear predictor for the third outcome.

The model formula considering all of these details is defined as

```
fff <- list(
  inla.mdata(Y),
```

```
  inla.surv(w, ev),
  o) ~ 0 +
  a1 + x1y + x2y + x3y + f(s, model = spde) +
  a2 + x1w + x2w + x3w + f(sc1, copy = "s", fixed = FALSE) +
  a3 + x1p + x2p + x3p + f(sc2, copy = "s", fixed = FALSE)
```

Data stacks

The data has to be organized so it contains all the information needed, that is all the outcomes, covariates and random effect indexes. For the SPDE models it also has to include the projector matrices.

The main work of the `inla.stack()` function is to check and organize outcomes, effects and projector matrices in order to guarantee that the dimentions match. Additionally, `inla.stack()` will deal with the problem of arranging the outcomes to work properly with multiple likelihoods, without us having to deal with the completion of lines and columns with NA. We just have to supply the data for each outcome and its linear predictor in a separated `inla.stack()` call and then it does the job properly when joining all the data stacks.

For the first outcome, we need to supply a five column matrix, each column with each of the five statistics computed for each of the `ng` groups, previously computed. For the effects we define an intercept, each one of the covariates with names according to what the `formula` specify and the index for the SPDE model. The projector matrices is one for the fixed effect, which is just `1`, and the one for the SPDE model.

The stack for the first outcome can be defined as

```
stackY <- inla.stack(
    tag = 'y',
    data = list(Y = YY),
    effects= list(
        data.frame(a1 = 1,
                   x1y = xxx.g$x1,
                   x2y = xxx.g$x2,
                   x3y = xxx.g$x3),
        s = 1:mesh$n),
    A=list(1,
           inla.spde.make.A(mesh, group.ce))
)
```

For the second outcome, we have to supply two vectors to build the `inla.surv` data, the covariates, with names according to the terms in the formula, and the projector matrices, the single `1` for the covariates and the projector for the shared SPDE term.

The stack for the second outcome can be defined as

```
stackW <- inla.stack(
    tag = 'w',
    data = list(w = we,
                ev = event),
    effects = list(
        data.frame(a2 = 1,
                   x1w = xxx[, 1],
                   x2w = xxx[, 2],
                   x3w = xxx[, 3]),
        sc1 = 1:mesh$n),
    A = list(1,
             inla.spde.make.A(mesh, loc))
)
```

For the third outcome we have to supply the vector of counts and the vector with the exposure. The effects and projector are similar with the previous others, just that we have to account for its names.

The stack for the third outcome can be defined as

```
stackP <- inla.stack(
    tag = 'p',
    data = list(o = po,
                E = ee),
    effects = list(
        data.frame(a3 = 1,
                   x1p = xxx[, 1],
                   x2p = xxx[, 2],
                   x3p = xxx[, 3]),
        sc2 = 1:mesh$n),
    A = list(1,
             inla.spde.make.A(mesh, loc))
)
```

Joining all the data into one data stack

```
stacked <- inla.stack(stackY, stackW, stackP)
```

Fitting the model

We now supply this to the `inla()` function in order to fit the model

```
result <- inla(
   formula = fff,
    family = c("agaussian", "weibullsurv", "poisson"),
    data = inla.stack.data(stacked),
    E = E,
    control.predictor = list(
        A=inla.stack.A(stacked)),
    control.family = list(
        list(),
        list(variant = 1),
        list())
)
#> Warning in inla.model.properties.generic(inla.trim.family(model), mm[names(mm) == : Model 'agaussian
#>   Use this model with extra care!!! Further warnings are disabled.
result$cpu
#>        Pre    Running       Post      Total
#>  1.1859901 75.0485556  0.2110319 76.4455776
```

We now compare the true value for each fixed effect with its posterior summary.

```
round(cbind(
    true = c(betas1, betas2, betas3),
    result$summary.fix), 2)
#>     true  mean   sd 0.025quant 0.5quant 0.975quant  mode kld
#> a1     5  5.08 1.18       2.66     5.07       7.51  5.06   0
#> x1y    0 -0.14 0.39      -0.91    -0.14       0.63 -0.14   0
#> x2y   -3 -3.09 0.43      -3.93    -3.09      -2.26 -3.09   0
#> x3y    3  3.35 0.48       2.42     3.35       4.29  3.34   0
```

```
#> a2     1  0.89 0.60      -0.35     0.88       2.14  0.88   0
#> x1w   -1 -1.00 0.10      -1.20    -1.00      -0.80 -1.00   0
#> x2w    0 -0.02 0.10      -0.21    -0.02       0.17 -0.02   0
#> x3w    1  1.04 0.10       0.85     1.04       1.24  1.04   0
#> a3     2  1.95 0.34       1.23     1.95       2.66  1.95   0
#> x1p    1  0.96 0.03       0.91     0.96       1.01  0.96   0
#> x2p   -1 -0.97 0.03      -1.02    -0.97      -0.92 -0.97   0
#> x3p    0  0.02 0.03      -0.03     0.02       0.07  0.02   0
```

For the hyper-parameters, we do have fitted one for the `agaussian` likelihood, one for the `weibulsuv` and we have those for the SPDE model assumed for the spatial smooth effect (defined from the sine and cosine functions) and the fitted parameters for this term shared from the first to the second and third outcomes.

```
round(cbind(true = c(1/sigma.y^2, alpha, NA, NA, gamma1, gamma2),
            result$summary.hy), 2)
#>                                                 true  mean   sd 0.025quant 0.5quant
#> Precision for the AggGaussian observations       1.0  0.96 0.06       0.85     0.96
#> alpha parameter for weibullsurv[2]               2.0  2.18 0.10       1.99     2.18
#> Range for s                                       NA 15.38 2.69      10.78    15.14
#> Stdev for s                                       NA  1.39 0.27       0.94     1.37
#> Beta for sc1                                     0.5  0.52 0.03       0.45     0.52
#> Beta for sc2                                    -0.3 -0.30 0.01      -0.33    -0.30
#>                                                 0.975quant  mode
#> Precision for the AggGaussian observations            1.08  0.96
#> alpha parameter for weibullsurv[2]                    2.38  2.18
#> Range for s                                          21.46 14.63
#> Stdev for s                                           2.00  1.31
#> Beta for sc1                                          0.58  0.52
#> Beta for sc2                                         -0.28 -0.30
```

References

Lindgren, F., H. Rue, and J. Lindström. 2011. "An Explicit Link Between Gaussian Fields and Gaussian Markov Random Fields: The Stochastic Partial Differential Equation Approach (with Discussion)." *J. R. Statist. Soc. B* 73 (4): 423–98.