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, of different type, and model them jointly.

Introduction

Usually correlations between outcomes are modeled by sharing common model terms in the linear predictor. These terms can be covariates, random effects or sums of these. In some cases, the likelihood requires more information than just the vector of observations. As an example, the likelihood for survival models requires both observed times and information about censoring.

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.

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 assumed to be 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, 10), c(0, 7))
```

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

```
rxy <- apply(bb, 1, diff)
rr <- 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 <- 1000
loc <- cbind(
    runif(n, bb[1, 1], bb[1, 2]),
    runif(n, bb[2, 1], bb[2, 2]))
```

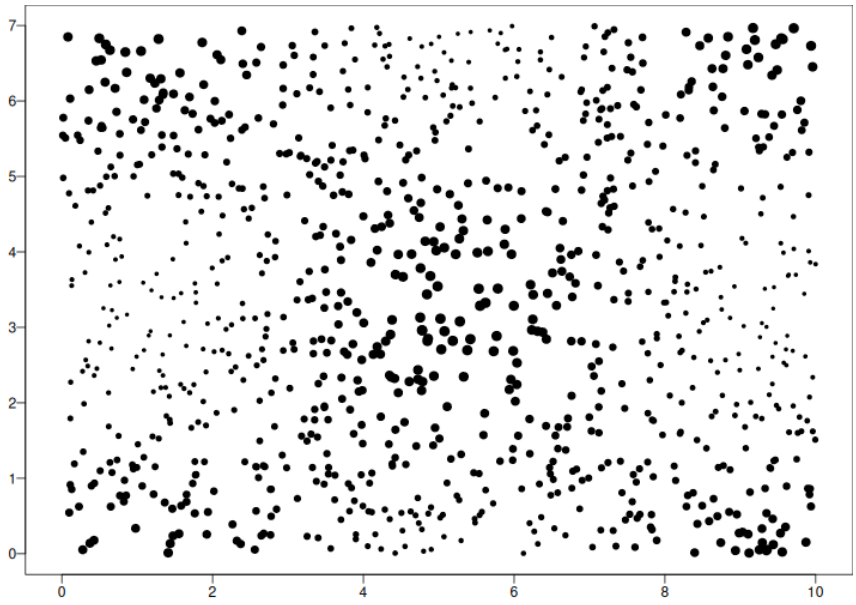
A spatial field

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] / rr,
         2 * pi * loc[, 2] / rr)
summary(u)
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> -1.98934 -0.76929 -0.01775 -0.07667  0.58622  1.99288
par(mar = c(2, 2, 0, 0), mgp = c(1.5, 0.5, 0), las = 1)
plot(loc, asp = 1, pch = 19, cex = 0.5 + (2 + u)/4, xlab = "", ylab = "")
```



Spatial sub-domains

In order to define the first outcome, let us partition our domain 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 these small sub-regions to be pixels based in the following set of centers:

```
h <- 1 ## 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] 70
```

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':    70 obs. of  4 variables:
#>  $ g : num  1 2 3 4 5 6 7 8 9 10 ...
#>  $ x1: num  0.542 0.453 0.413 0.501 0.58 ...
#>  $ x2: num  0.515 0.435 0.39 0.638 0.46 ...
#>  $ x3: num  0.548 0.558 0.607 0.557 0.49 ...
```

Outcomes

We will consider three outcomes.

First outcome: aggregated Gaussian

The first outcome is Gaussian distributes, with unknown mean and variance, and a known scaling factor, s, such that

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

We assume $\boldsymbol{\eta}_y = \{\mu_1, ..., \mu_n\}$ and

$$\boldsymbol{\eta}_y = \mathbf{X}\boldsymbol{\beta}_y + \mathbf{u}.$$

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

```
beta.y <- 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)])) %*% beta.y) + 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.
#>  0.3739  1.7863  2.2935  2.3933  2.9033  6.0581
```

```
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.
#>   1.090   4.075   4.964   4.947   5.841   8.347
```

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)
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 [1:70, 1:5] 0.218 0.418 0.15 0.389 0.242 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : chr [1:70] "init" "" "" "" ...
#>   ..$ : chr [1:5] "Y1" "Y2" "Y3" "Y4" ...
```

Second outcome: Survival

For the second outcome we choose a Weibull distribution with the following parametrization

$$f_W(w_i) = \alpha w_i^{\alpha-1} \lambda_i^{\alpha} \mathrm{e}^{-(\lambda_i w_i)^{\alpha}}.$$

We define the linear predictor as $\boldsymbol{\eta}_w = \{\log(\lambda_1), ..., \log(\lambda_n)\}$ and

$$\boldsymbol{\eta}_w = \mathbf{X}\beta_w + \gamma_w \mathbf{u}$$

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

```
beta.w <- c(1, -1, 0, 1)
alpha <- 2
gamma.w <- 0.5
lambdaw <- exp(cbind(1, xxx) %*% beta.w + gamma.w * u)
we0 <- rweibull(n, shape = alpha, scale = 1/lambdaw)
summary(we0)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.01604 0.15938 0.29232 0.40070 0.54882 2.32019
```

However, we suppose that some observations are censored

```
summary(u.ev <- runif(n, 0.3, 1)) ## censoring factor
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.3006  0.4792  0.6429  0.6466  0.8066  0.9985
table(event <- rbinom(n, 1, 0.5)) ## censored (=0) or event (=1)
#>
#>   0   1
#> 519 481
summary(we <- ifelse(event == 1, we0, we0 * u.ev)) ## censored outcome
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.01096 0.11519 0.23098 0.32351 0.41337 2.18450
```

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.
#>   1.779   3.875   4.833   4.989   5.933   9.679
```

We assume an outcome following a Poisson distribution with

$$O_i \sim \text{Poisson}(\delta_i E_i).$$

We assume the linear predictor as $\boldsymbol{\eta}_p = \{\log(\delta_1), \ldots, \log(\delta_n)\}$ and

$$\boldsymbol{\eta}_p = \mathbf{X}\beta_p + \gamma_p \mathbf{u}$$

This outcome is being simulated with

```
beta.p <- c(2, 1, -1, 0)
gamma.p <- -0.3
delta.p <- exp(cbind(1, xxx) %*% beta.p + gamma.p * u)
po <- rpois(n, delta.p * ee)
summary(po)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    4.00   23.00   35.00   42.27   55.00  186.00
```

Data model setup

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

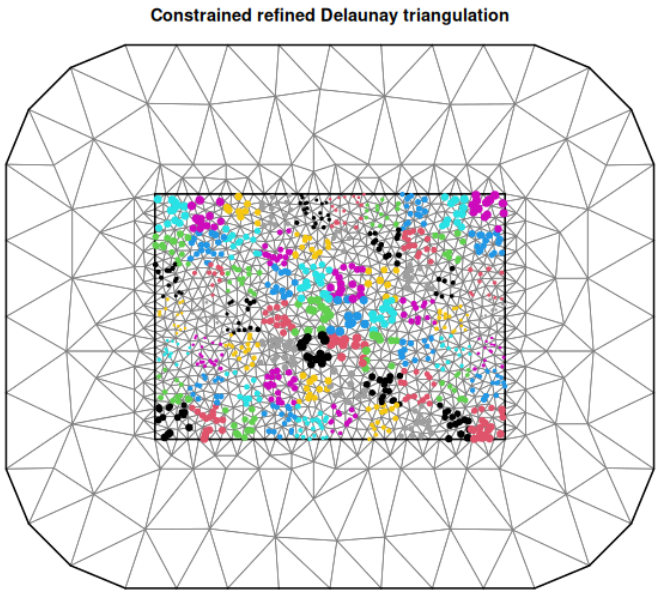
SPDE model setup

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),
    offset = c(.0001, .5) * rr, ## extensions size
    max.edge = c(.05, .3) * rr,
    cutoff = 0.025 * rr)
mesh$n
#> [1] 1157
```

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 meant to me used when working with SPDE models. However, it also helps when considering multiple likelihood data as detailed below.

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, in this vignette we are dealing with more complex cases. Therefore, he need 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 include the same covariate but 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)
```

Building 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.

For the first outcome, we need to supply a five column matrix, each with one of the five statistics previously computed for the `ng` groups. For the effects we define an intercept, 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.

We will include a different `tag` for each stack to keep track. This is useful to collect results associated to each outcome. For each outcome we can also define an indicator vector to identify it. This is useful for the case of having some missing data. In this case the indicator can be used to identify which link to be used for making the predictions.

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 (observed time and censoring) 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))
)
```

Fitting the model and some results

We now fit the model and look at some results.

Model fitting

To fit the model we start by joining all the data into one data stack

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

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()),
   control.compute = list(dic = TRUE, waic = TRUE, cpo = TRUE)
)
#> 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.
#> Warning in sqrt(ee): NaNs produced
result$cpu
#>         Pre     Running        Post       Total
#>  0.77612972 64.03728771  0.09999871 64.91341615
```

Summaries

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

```
round(cbind(
    true = c(beta.y, beta.w, beta.p),
    result$summary.fix), 2)
#>     true  mean   sd 0.025quant 0.5quant 0.975quant  mode kld
```

```
#> a1     5  5.59 1.14       3.31     5.58       7.92  5.58   0
#> x1y    0 -0.01 0.43      -0.85    -0.01       0.83 -0.01   0
#> x2y   -3 -3.39 0.36      -4.09    -3.39      -2.69 -3.39   0
#> x3y    3  3.09 0.37       2.36     3.09       3.81  3.09   0
#> a2     1  1.02 0.52      -0.03     1.01       2.08  1.00   0
#> x1w   -1 -1.06 0.07      -1.20    -1.06      -0.91 -1.06   0
#> x2w    0  0.02 0.07      -0.11     0.02       0.16  0.02   0
#> x3w    1  1.03 0.07       0.89     1.03       1.17  1.03   0
#> a3     2  1.86 0.33       1.18     1.86       2.52  1.86   0
#> x1p    1  1.01 0.02       0.98     1.01       1.05  1.01   0
#> x2p   -1 -1.01 0.02      -1.04    -1.01      -0.98 -1.01   0
#> x3p    0 -0.01 0.02      -0.04    -0.01       0.03 -0.01   0
```

For the hyper-parameters, we have one for the `agaussian` likelihood, one for the `weibulsuv`, two for the SPDE model and two that define the scaling for the shared SPDE terms in the second and third outome.

```
round(cbind(true = c(1/sigma.y^2, alpha, NA, NA, gamma.w, gamma.p),
            result$summary.hy), 2)
#>                                              true  mean   sd 0.025quant 0.5quant
#> Precision for the AggGaussian observations    1.0  1.06 0.04       0.98     1.06
#> alpha parameter for weibullsurv[2]            2.0  2.51 0.29       2.14     2.44
#> Range for s                                    NA  8.72 2.54       5.65     8.29
#> Stdev for s                                    NA  1.14 0.18       0.90     1.12
#> Beta for sc1                                  0.5  0.49 0.03       0.44     0.49
#> Beta for sc2                                 -0.3 -0.31 0.01      -0.33    -0.31
#>                                              0.975quant  mode
#> Precision for the AggGaussian observations         1.13  1.06
#> alpha parameter for weibullsurv[2]                 3.24  2.22
#> Range for s                                       15.42  6.25
#> Stdev for s                                        1.59  1.02
#> Beta for sc1                                       0.55  0.48
#> Beta for sc2                                      -0.30 -0.31
```

Results considering each outcome

To collect outputs associated with each data, we need to consider the `tag` from each data stack to get the data index from the joint stack. For example, the data index for the third outcome is

```
idx1 <- inla.stack.index(stacked, tag = "p")$data
```

We can see the summary for the fitted values for the first few observations of this outcome with

```
head(cbind(true = delta.p,
           result$summary.fitted.values[idx1, ]), 3)
#>                              true      mean        sd 0.025quant  0.5quant
#> fitted.APredictor.1071  9.798490  9.937648 0.4202856   9.140347  9.928204
#> fitted.APredictor.1072 11.039282 10.834178 0.4280471  10.017856 10.826024
#> fitted.APredictor.1073  8.644928  8.302297 0.3707498   7.589743  8.297172
#>                        0.975quant      mode
#> fitted.APredictor.1071   10.788834  9.756684
#> fitted.APredictor.1072   11.696891 10.649086
#> fitted.APredictor.1073    9.044317  8.169296
```

For access the DIC, WAIC and log score can consider the family indicator stored in the DIC output as follows

```
str(result$dic)
#> List of 14
#>  $ dic               : num 8450
#>  $ p.eff             : num 119
#>  $ mean.deviance     : num 8331
#>  $ deviance.mean     : num 8212
#>  $ dic.sat           : num 2053
#>  $ mean.deviance.sat: num 1934
#>  $ deviance.mean.sat: num 1755
#>  $ family.dic        : num [1:3] 2046.5 -15.8 6419.8
#>  $ family.dic.sat    : num [1:3] 92.1 990.6 1030.1
#>  $ family.p.eff      : num [1:3] 33.6 15 70.5
#>  $ family            : num [1:2070] 1 1 1 1 1 1 1 1 1 1 ...
#>  $ local.dic         : num [1:2070] 27.1 31.6 18.8 31.2 34.1 ...
#>  $ local.dic.sat     : num [1:2070] 2.856 1.043 0.902 1.172 2.254 ...
#>  $ local.p.eff       : num [1:2070] 0.645 0.44 0.678 0.549 0.68 ...
str(result$waic)
#> List of 4
#>  $ waic       : num 8443
#>  $ p.eff      : num 99.5
#>  $ local.waic : num [1:2070] 27.9 31.4 18.3 30.9 34.5 ...
#>  $ local.p.eff: num [1:2070] 1.105 0.148 0.127 0.201 0.814 ...
str(result$cpo)
#> List of 3
#>  $ cpo    : num [1:2070] 6.14e-07 1.51e-07 1.04e-04 1.87e-07 2.38e-08 ...
#>  $ pit    : num [1:2070] 0.00308 0.35458 0.5059 0.25368 0.01484 ...
#>  $ failure: num [1:2070] 0 0 0 0 0 0 0 0 0 0 ...
result$dic$family.dic ### DIC for each outcome
#> [1] 2046.53053  -15.79101 6419.75156
tapply(result$waic$local.waic,
       result$dic$family, ## taken from dic family id
       sum) ## WAIC for each outcome
#>          1          2          3
#> 2039.80740  -17.64628 6420.65501
tapply(result$cpo$cpo,
       result$dic$family, ## taken fron dic family id
        function(x)
         c(nlCPO = -sum(log(x)))) ## -sum log CPO for each outcome
#>           1           2           3
#> 1024.524871   -8.723326 3211.134888
```

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.