Avoiding trouble with factors in *inla.stack()*

Elias T. Krainski (elias@r-inla.org)

Aug 7th 2017

Introduction

This short note describe a way to work with factor covariates when working with *inla.stack()*. You can skip the next section if you already know how to work with factors in functions such as *lm()* or *glm()*.

Working with factors in R

Let the simple linear model

$$y_{ijk} = \mu_0 + a_i + b_j + f_k + \beta x_{ijk} + e_{ijk}$$

and the following data from it

```
dat <- expand.grid(a=0:1, b=0:1, d=0:2) ### three discrete
dat$x <- runif(12) ### one continuous covariate
dat$y <- 3*dat$a - dat$b + (dat$d-1)*2 + dat$x + rnorm(12,0,0.1)
```

A factor with only two levels can be encoded as numeric 0 or 1 or as *factor*. A factor with more than two levels always have to be encoded as factors. So we must encode d as factor

```
dat$d <- factor(dat$d)
```

The model can be fitted as

```
coef(lm(y ~ 1 + a + b + d + x, dat))
```

```
## (Intercept)           a           b          d1          d2           x
##   -2.015034    2.955168   -1.076419    2.100355    4.041895    1.046020
```

where we explicited each model term.

Let us pay attention for the fact that the default option for contrasts is the **treatment**. In this case a reference level from each covariate coded as factor is droped, the first level by default. In this case the '(Intercept)' is the scenario when each numeric covariate is zero and the factor covariates are at its reference level. The coefficient associated to a numeric covariate measures the effect of changing one unity in this covariate. For the covariates encoded as factors, there is one coefficient for each not droped level and it measures the effect of changing from the reference level to this level.

The intercept is considered by default no needing to be explicited.

```
coef(lm(y ~ a + b + d + x, dat))
```

```
## (Intercept)           a           b          d1          d2           x
##   -2.015034    2.955168   -1.076419    2.100355    4.041895    1.046020
```

It can be removed by adding '-1' in any part of the right side of the formula

```
coef(lm(y ~ -1 + a + b + d + x, dat))
```

```
##           a           b          d0          d1          d2           x
##  2.95516806 -1.07641869 -2.01503389  0.08532143  2.02686106  1.04602049
```

or writing a literal formula explicty saying 'no intercept' inserting '0' in any part of the right side of it

```
    coef(lm(y ~ 0 + a + b + d + x, dat))
```

```
##           a           b          d0          d1          d2           x
##  2.95516806 -1.07641869 -2.01503389  0.08532143  2.02686106  1.04602049
```

Notice that the '(Intercept)' from before is now 'd0' because it is the scenario for the first level in 'd' and when all the other terms (all numeric) as zero. When there is no intercept this happens to be the case for the first factor in the formula.

When the '(Intercept)' is in the model, changing the order of the terms in the formula does effect what is being fitted, only the order

```
    coef(lm(y ~ b + a + d + x, dat))
```

```
## (Intercept)           b           a          d1          d2           x
##   -2.015034   -1.076419    2.955168    2.100355    4.041895    1.046020
```

```
    coef(lm(y ~ d + a + b + x, dat))
```

```
## (Intercept)          d1          d2           a           b           x
##   -2.015034    2.100355    4.041895    2.955168   -1.076419    1.046020
```

The covariates with only two levels encoded as "0" or "1" can be encoded as factor assuming any two labels

```
dat$a <- factor(dat$a, levels=0:1, labels=c('1st', '2nd'))
dat$b <- factor(dat$b, levels=1:0, labels=c('2nd', '1st')) ## OBS: reference level changed
```

and we will have the same coefficients as when not having it as factors only when '0' is the reference level

```
coef(lm(y ~ a + b + d + x, dat))
```

```
## (Intercept)        a2nd        b1st          d1          d2           x
##   -3.091453    2.955168    1.076419    2.100355    4.041895    1.046020
```

When the intercept is not in the formula it will be the reference level of the first factor and the order matters

```
    coef(lm(y ~ 0 + a + b + d + x, dat))
```

```
##       a1st       a2nd       b1st         d1         d2          x
## -3.0914526 -0.1362845  1.0764187  2.1003553  4.0418950  1.0460205
```

```
    coef(lm(y ~ 0 + b + a + d + x, dat))
```

```
##      b2nd      b1st      a2nd        d1        d2         x
## -3.091453 -2.015034  2.955168  2.100355  4.041895  1.046020
```

```
    coef(lm(y ~ 0 + d + a + b + x, dat))
```

```
##         d0         d1         d2       a2nd       b1st          x
## -3.0914526 -0.9910973  0.9504424  2.9551681  1.0764187  1.0460205
```

It is important to notice that the way one codes de model matters. We showed the case under the **contrast parametrization**. Other contrast options are available in **R** and one can see these in *help(contrast)*.

Another important point is to notice that it is not a feature of the *lm()* or *glm()* function. The acual model matrix being prepared using the *model.matrix()* function. This fuction is used internally to create tge dummie variables. Everyone can use it directly just suplying the formula and data,

```
model.matrix(~a+b+d+x, dat)
```

and you can see what happens with each above cases.

Now we prepare ourselves to understand how to work with factors in *inla.stack()*.

Dealing with factors in *inla.stack()*

The *inla.stack()* function helps to organize data when the model has components with different projection matrices. Let us consider the *Tokyo* data which is data about rain each day over two years grouped by day of the year.

```
data(Tokyo)
str(Tokyo)
```

```
## 'data.frame':    366 obs. of  3 variables:
##  $ y   : int  0 0 1 1 0 1 1 0 0 0 ...
##  $ n   : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ time: int  1 2 3 4 5 6 7 8 9 10 ...
```

y is 0, 1 or 2, n is 1 (February 29) or 2 and *time* is 1, 2, ..., 366.

Let us have a set of knots over time in order to build a model, see Lindgren and Rue (2015).

```
knots <- seq(1, 367, length = 25)
mesh <- inla.mesh.1d(knots, interval = c(1, 367), degree = 2, boundary = "cyclic")
spde <- inla.spde2.pcmatern(mesh,
    prior.sigma=c(1, 0.01), ## P(sigma > 1) = 0.01
    prior.range=c(1, 0.01)) ## P(range < 1) = 0.01
A <- inla.spde.make.A(mesh, loc = Tokyo$time)
time.index <- inla.spde.make.index("time", n.spde = spde$n.spde)
```

Let us add two factor covariates to Tokyo data and a numeric one.

```
Tokyo$a <- factor(rbinom(366, 1, 0.5))
Tokyo$b <- factor(rbinom(366, 2, 0.5))
Tokyo$x <- runif(366)
```

When working with factor covariates it is better to to build the design matrix and supply it to *inla.stack()*. We can include the other covariates as well.

```
abx <- model.matrix(~a+b+x, Tokyo)[, -1]
```

The automatic intercept at the first column was droped. When supplying it in *inla.stack()* we will join an explict intercept as we usually do when working with SPDE models.

```
stack <- inla.stack(
  data = list(y = Tokyo$y, link = 1, Ntrials = Tokyo$n),
  A = list(A, 1),
  effects = list(time.index, data.frame(mu0=1, abx)),
  tag = "est")
formula <- y ~ 0 + mu0 + a1 + b1 + b2 + x + f(time, model = spde)
data <- inla.stack.data(stack)
result <- inla(formula, family = "binomial",
               data = data,
               Ntrials = data$Ntrials,
               control.predictor = list(
                 A = inla.stack.A(stack),
                 link = data$link,
```

```
                compute = TRUE))
result$summary.fixed[, 1:5]
```

```
##            mean        sd 0.025quant    0.5quant  0.975quant
## mu0 -0.86191324 0.3748084 -1.6037616 -0.86155976 -0.12166814
## a1  -0.08429582 0.1765061 -0.4303519 -0.08433415  0.26197701
## b1  -0.35000860 0.2099586 -0.7619228 -0.34995745  0.06161657
## b2  -0.29757028 0.2408365 -0.7699136 -0.29756482  0.17474217
## x    0.06289912 0.3141183 -0.5528724  0.06280194  0.67921968
```

When there is a prediction scenario

Let us build a prediction scenario

```
pred.sc <- expand.grid(a1=0:1, b1=0:1, b2=0, x=c(0.5))
pred.sc
```

```
##   a1 b1 b2   x
## 1  0  0  0 0.5
## 2  1  0  0 0.5
## 3  0  1  0 0.5
## 4  1  1  0 0.5
```

For the random effect, over time, we do need to build a projection as well.

```
A.pred <- inla.spde.make.A(mesh, loc=rep(180, nrow(pred.sc)))
```

This scenario can be supplied in a new data stack as

```
stack.pred <- inla.stack(
  data = list(y = NA, link = 1, Ntrials = 2),
  A = list(A.pred, 1),
  effects = list(time.index, data.frame(mu0=1, pred.sc)),
  tag = "pred")
stack.full <- inla.stack(stack, stack.pred)
data <- inla.stack.data(stack.full)
result <- inla(formula, family = "binomial",
               data = data,
               Ntrials = data$Ntrials,
               control.predictor = list(
                 A = inla.stack.A(stack.full),
                 link = data$link,
                 compute = TRUE),
               control.mode=list(theta=result$mode$theta,
                                 restart=FALSE))
```

Getting the predictions

```
idx.pred <- inla.stack.index(stack.full, tag='pred')$data
result$summary.fitted.val[idx.pred, 1:5]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.APredictor.367 0.5581193 0.08194158  0.3978885 0.5584685  0.7164533
## fitted.APredictor.368 0.5378094 0.08421885  0.3751193 0.5374762  0.7023801
## fitted.APredictor.369 0.4731606 0.07610981  0.3304369 0.4712395  0.6267411
## fitted.APredictor.370 0.4528254 0.07887628  0.3066827 0.4502262  0.6135467
```

References

Lindgren, F., and H. Rue. 2015. "Bayesian Spatial and Spatio-Temporal Modelling with R-INLA." *Journal of Statistical Software* 63 (19).