The z-model

Parametrization

The z-model is an implementation of the "classical" way to define the "random effect" part of a mixed model, through

$$\eta = \ldots + Zz$$

where Z is a $n \times m$ matrix and z a vector of length m representing zero-mean "random effects". The z-model is defined as the augmented model

$$\widetilde{z} = \begin{pmatrix} v \\ z \end{pmatrix}$$

where $v \sim \mathcal{N}_n(Zz, \kappa I)$, where κ is a high fixed precision, and where the precision matrix for z is τC where $C > 0$ is a $m \times m$ (fixed) matrix and τ is the precision parameter.

Hyperparameters

The precision parameter of the z-model is represented as

$$\theta = \log(\tau)$$

and prior is assigned to θ. The parameter κ is kept fixed at all times.

Specification

The z-model is specified inside the `f()` function as

```
f(<whatever>, model="z", Z = <Z>, Cmatrix = <Cmat>, hyper = <hyper>,
  precision = <precision>)
```

where the required `Z`-matrix argument defines the Z matrix. The (optional) `Cmatrix` defines the C matrix and is by default taken to the the diagonal matrix with dimension m. The `precision` parameter defines the value of κ, and `hyper` the hyperparameter spesification for τ.

If Z is a $n \times m$ matrix then the C matrix must be $m \times m$ matrix, and $\widetilde{z}$ has length $n + m$. The n first terms of $\widetilde{z}$ is v and the last m terms of $\widetilde{z}$ is z.

If `constr=TRUE` is given, then this is defined as $\sum_{i=1}^{m} z_i = 0$. If `extraconstr` is given, then it is applied to $\widetilde{z}$, hence `extraconstr$A` must be a $k \times (n + m)$ matrix where k is the number of linear constraints.

Hyperparameter spesification and default values

hyper

theta

name log precision

short.name prec

initial 4

fixed FALSE

prior loggamma

param 1 5e-05

to.theta `function(x) log(x)`

from.theta `function(x) exp(x)`

constr FALSE

nrow.ncol FALSE

augmented FALSE

aug.factor 1

aug.constr

n.div.by

n.required TRUE

set.default.values TRUE

pdf z

status experimental

Example 1

```
## Example for implementing the standard mixed model
##    y = X%*%beta + Z%*%b+e
## with b ~N(0,tau * Q) With Q a precision matrix

library(INLA)
set.seed(123)
n = 100
nz = 5
nb = 4
tmp = matrix(rnorm(nz^2), nz, nz)
Qz = tmp %*% t(tmp)
Z = matrix(runif(n*nz), n, nz)
b = inla.qsample(1, Q=Qz)
beta = 5 + rnorm(nb)
X = matrix(runif(n*nb), n, nb)
X[,1] = 1   # want an intercept

eta = X %*% beta + Z %*% b
y = eta + rnorm(n, sd=0.01)
Qx = diag(nb)

formula = y ~ -1 + X +  f(id.z, model="z", Cmatrix=Qz,  Z=Z)
result = inla(formula, data = list(y=y, id.z = 1:n, X=X))

plot(b, result$summary.random$id.z$mean[-(1:n)], pch=19)
abline(0, 1)
```

Example 2

```
## An example demonstrating two ways to implement the model eta = Z*z,
## where z ~ N(0, tau*Q).

## Simulate data
n = 100
m = 10
Z = matrix(rnorm(n*m), n, m)
rho = 0.8
```

```
Qz = toeplitz(rho^(0:(m-1)))
prec.fixed = FALSE  ## the precision parameter for z
z = inla.qsample(1, Q=Qz)
eta = Z %*% z
s = 0.1 ## noise stdev
s.fixed = TRUE
y = eta + rnorm(n, sd = s)

## This is normally not needed at all, but it demonstrate how to set
## the 'high precisions': in the z-model, in the A-part of the linear
## predictor, and in the linear predictor iteself.
precision = exp(15)

## The first approach use the z-model.
r = inla(y ~ -1 + f(idx, model="z", Z=Z,
                    precision=precision,
                    Cmatrix=Qz,
                    hyper = list(
                            prec = list(
                                    initial = 0,
                                    fixed = prec.fixed,
                                    param = c(1, 1)))),
        data = list(y=y, idx=1:n),
        control.family = list(
                hyper = list(
                        prec = list(
                                initial = log(1/s^2),
                                fixed=s.fixed))),
        control.predictor = list(
                compute=TRUE,
                precision=precision,
                initial = log(precision)))

## The second one uses the A-matrix
rr = inla(y ~ -1 + f(idx, model="generic",
                     precision = precision,
                     Cmatrix=Qz,
                     hyper = list(
                             prec = list(
                                     initial = 0,
                                     fixed = prec.fixed,
                                     param = c(1, 1)))),
        data = list(y=y, idx=1:m),
        control.family = list(
                hyper = list(
                        prec = list(
                                initial = log(1/s^2),
                                fixed=s.fixed))),
        control.predictor = list(
                compute=TRUE,
                A=Z,
                precision=precision,
                initial = log(precision)))

## Plot some results
par(mfrow=c(2, 2))
plot(r$summary.linear.predictor$mean[1:n], eta,
```

```
     main="z-model: (eta.estimated, eta)")
plot(r$summary.linear.predictor$mean[1:n], eta,
     main="generic-model: (eta.estimated, eta)")
plot(r$internal.marginals.hyperpar[[1]],
     main="Prec.param (both)")
lines(rr$internal.marginals.hyperpar[[1]])

## compare (log) marginal likelihood. recall to add the missing part,
## see inla.doc("generic")
print(r$mlik - (rr$mlik + 0.5*log(det(Qz))))


r = inla.hyperpar(r)
rr = inla.hyperpar(rr)
plot(r$internal.marginals.hyperpar[[1]],
     main="Prec.param (improved, both)")
lines(rr$internal.marginals.hyperpar[[1]])

## compare (log) marginal likelihood. recall to add the missing part,
## see inla.doc("generic")
print(r$mlik - (rr$mlik + 0.5*log(det(Qz))))
```

Example 3

```
## This example demonstrate how to use the z-model with intrinsic
## models. The z-model must be proper, which we have to mimic if we
## are using an intrinsic model

## Simulate some data
n = 100
idx = 1:n
x = sin(idx / n * 4 * pi)
s = 0.1
y = x + rnorm(n, sd=s)

## Parameters for the loggamma prior
prior = c(1, 0.001)

## A small constant we add to the diagonal to prevent the model to be
## intrinsic.
d = 1e-8

## RW1
r = inla(y ~ -1 + f(idx, model="rw1", param=prior,
                    constr=TRUE, diagonal=d),
        data = data.frame(y, idx),
        control.family = list(
                hyper = list(
                        prec = list(
                                initial = log(1/s^2),
                                fixed = TRUE))))
C = toeplitz(c(2, -1, rep(0, n-2)))
C[1, 1] = C[n, n] = 1
## We must add the extra diagonal contribution here, as otherwise it
## applies to the hole model (v, z)
diag(C) = diag(C) + d
```

```
Z = diag(n)
rr = inla(y ~ -1 + f(idx, model="z", Z=Z,
                     Cmatrix = C, constr=TRUE, param=prior),
        data = data.frame(y, idx),
        control.family = list(
                hyper = list(
                        prec = list(
                                initial = log(1/s^2),
                                fixed = TRUE))))
##
par(mfrow=c(2, 2))
plot(idx, r$summary.random$idx$mean)
lines(idx, rr$summary.random$idx$mean[1:n])
title("rw1: idx")
plot(r$internal.marginals.hyperpar[[1]])
lines(rr$internal.marginals.hyperpar[[1]])
title("rw1: log.prec")

## RW2
r = inla(y ~ -1 + f(idx, model="rw2",
        ## we cannot define the rankdef for the z-model, but it will
        ## be set to 1 as constr=TRUE. so we're using the same here,
        ## even though the rankdef is 0, since we added 'd' on the
        ## diagonal.
        rankdef = 1,
        param=prior, constr=TRUE, diagonal = d),
        data = data.frame(y, idx),
        control.family = list(
                hyper = list(
                        prec = list(
                                initial = log(1/s^2),
                                fixed = TRUE))))

C = toeplitz(c(2, -1, rep(0, n-3), -1))
C = C[-c(1, n), ]
C = t(C) %*% C
## We must add the extra diagonal contribution here, as otherwise it
## applies to the hole model (v, z)
diag(C) = diag(C) + d
Z = diag(n)
rr = inla(y ~ -1 + f(idx, model="z", Z=Z, Cmatrix = C,
        constr=TRUE, param=prior),
        data = data.frame(y, idx),
        control.family = list(
                hyper = list(
                        prec = list(
                                initial = log(1/s^2),
                                fixed = TRUE))))
##
plot(idx, r$summary.random$idx$mean)
lines(idx, rr$summary.random$idx$mean[1:n])
title("rw2: idx")
plot(r$internal.marginals.hyperpar[[1]])
lines(rr$internal.marginals.hyperpar[[1]])
title("rw2: log.prec")
```

Notes

None.