

Smooth-Copy of another model component: "scopy"

This model is a generalization of `copy`, please refer to `inla.doc("copy")` first.

This describes the way to copy another model component with an optional smooth/spline scaling, like with

$$\eta = u + v$$

where v is a smooth copy of u (component-wise)

$$v = \beta(z) \times \text{copy}(u)$$

where $\beta(z)$, a smooth/spline function of the covariate z . The smooth scaling is done **component-wise** for u , so if u are defined with domain $(1, 2, \dots, m)$, i.e. $u = (u_1, u_2, \dots, u_m)$, then z must be $z = (z_1, z_2, \dots, z_m)$, so that

$$v_i = \beta(z_i)u_i, \quad i = 1, 2, \dots, m.$$

Hyperparameters

The hyperparameters are the value of the spline at n fixed locations, (l_i, β_i) , for $i = 1, \dots, n$. The function $\beta(z)$, is defined as (using z as the covariate)

```
zr <- range(z)
l <- seq(zr[1], zr[2], len=n)
beta.z <- splinefun(l, beta, method = "natural")
```

We can control β and its prior distribution using argument `control.scopy` within `f()`,

```
control.scopy = list(
  covariate = ...,
  n = 5,
  model = "rw2",
  mean = 1.0,
  prec.mean = 1.0,
  prec.betas = 10.0)
```

where

covariate gives the covariate that is used

n is the number of hyperparameters used in the spline ($3 \leq n \leq 15$).

model the prior model for $\{\beta_i\}$, either `rw1` or `rw2`. This model is scaled (like with `scale.model=TRUE`).

mean The prior mean for the (weighted-)mean¹ of $\{\beta_i\}$

prec.mean The prior precision for the (weighted-)mean of $\{\beta_i\}$

prec.betas The prior precision for the `rw1/rw2` model for $\{\beta_i\}$

Note that the prior mean and both prior precisions, are *fixed* and not *random*.

The `f()`-argument **precision**, defines how close the copy is, is similar as for model `copy`.

¹The mean of $\{\beta_i\}$ is defined to approximate the integral of the RW, hence its $\left(\frac{1}{2}(\beta_1 + \beta_n) + \sum_{j=2}^{n-1} \beta_j\right) / (n-1)$.

Spesification

doc Create a scopy of a model component

hyper

theta1

hyperid 36101
name beta1
short.name b1
initial 0.1
fixed FALSE
prior none
param
to.theta function(x) x
from.theta function(x) x

theta2

hyperid 36102
name beta2
short.name b2
initial 0.1
fixed FALSE
prior none
param
to.theta function(x) x
from.theta function(x) x

theta3

hyperid 36103
name beta3
short.name b3
initial 0.1
fixed FALSE
prior none
param
to.theta function(x) x
from.theta function(x) x

theta4

hyperid 36104
name beta4
short.name b4
initial 0.1
fixed FALSE
prior none
param
to.theta function(x) x

```

    from.theta function(x) x
theta5
    hyperid 36105
    name beta5
    short.name b5
    initial 0.1
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta6
    hyperid 36106
    name beta6
    short.name b6
    initial 0.1
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta7
    hyperid 36107
    name beta7
    short.name b7
    initial 0.1
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta8
    hyperid 36108
    name beta8
    short.name b8
    initial 0.1
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta9
    hyperid 36109
    name beta9

```

```

    short.name b9
    initial 0.1
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta10
    hyperid 36110
    name beta10
    short.name b10
    initial 0.1
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta11
    hyperid 36111
    name beta11
    short.name b11
    initial 0.1
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta12
    hyperid 36112
    name beta12
    short.name b12
    initial 0.1
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta13
    hyperid 36113
    name beta13
    short.name b13
    initial 0.1
    fixed FALSE
    prior none

```

```

    param
    to.theta function(x) x
    from.theta function(x) x
theta14
    hyperid 36114
    name beta14
    short.name b14
    initial 0.1
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta15
    hyperid 36115
    name beta15
    short.name b15
    initial 0.1
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x

constr FALSE

nrow.ncol FALSE

augmented FALSE

aug.factor 1

aug.constr

n.div.by

n.required FALSE

set.default.values FALSE

status experimental

pdf scopy

```

Example

Just simulate some data and estimate the parameters back.

```
## Use the spline part of scopy to estimate a spline.
## This example is rather artificial, but illustrate
## well the idea.
N <- 100
s <- 0.1
x <- 1:N
y <- 1 + sin(x * 0.25) * exp(-2*x/N) + rnorm(N, sd = s)
m <- 15
r <- inla(y ~ -1 +
  ##
  ## this model will just define the overall mean level, but
  ## with one value for each i. We need this as as can then
  ## scale this one with the spline
  ##
  f(idx,
    model = "rw1",
    scale.model = TRUE,
    constr = FALSE,
    values = 1:N,
    hyper = list(prec = list(initial = 20, fixed = TRUE))) +
  ##
  ## the 'overall level' is scaled by a spline
  ##
  f(idx.scopy,
    scopy = "idx",
    control.scopy = list(covariate = x,
                        n = m,
                        mean = 1,
                        prec.mean = 1,
                        prec.betas = 10,
                        model = "rw2")),
  ##
  data = list(idx = rep(NA, N),
    idx.scopy = 1:N,
    x = x,
    m = m),
  ##
  control.family = list(hyper = list(
    prec = list(
      initial = log(1/s^2),
      fixed = TRUE))),
  ##
  control.compute = list(config = TRUE),
  verbose = TRUE)
plot(x, y, pch = 19)
## note that the locations are not stored in the results, hence we can set them here. This is
## just for the ease of the output, the results are unchanged. 1. we can just rescale
beta <- inla.summary.scopy(r, "idx.scopy", range = c(1, N))
s <- mean(r$summary.random$idx$mean)
lines(beta$x, s * beta$mean, lwd = 3, col = "blue")
lines(beta$x, s * beta$mean + 2 * s * beta$sd, lwd = 1, col = "black")
lines(beta$x, s * beta$mean - 2 * s * beta$sd, lwd = 1, col = "black")
```

Notes

This model is experimental.