

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 (equally distant) locations, (l_i, β_i) , for $i = 1, \dots, n$. Let $l_M = (l_n + l_1)/2$ be the mid-points of the locations and $l_L = (l_n - l_1)$ its length. These β -parameters defines the interpolating spline (of second order). Let Q be the scaled-precision matrix for RW2 and define the eigen decomposition, as

$$Q = \sum_{i=1}^n \lambda_i v_i v_i^T$$

where $\lambda_{n-1} = \lambda_n = 0$, assuming descreasing eigenvalues. We can use $v_n = (1, \dots, 1)^T$ and $v_{n-1} = (-0.5, \dots, 0.5)^T$. Define the matrix with scaled v_i ' as columns,

$$W = \begin{bmatrix} v_n^T & | & v_{n-1}^T & | & \frac{1}{\sqrt{\lambda_{n-2}}} v_{n-2}^T & | & \dots & | & \frac{1}{\sqrt{\lambda_1}} v_1^T \end{bmatrix}.$$

Let the columns vectors of W be $w_1, w_2, \dots$. We parameterize the spline at locations $l_1, \dots, l_n$, as $\beta = (\beta_1, \dots, \beta_n)$, as

$$\beta = \sum_{i=1}^n \theta_i w_i, \quad i = 1, \dots, n,$$

In this way, θ_1 is the overall mean, θ_2 is the (dimension-less) slope, and $\theta_3, \dots, \theta_n$ are weights for the basis-function expansion of the spline that is beyond the constant and the linear term.

Since Q is scaled, then using independent $N(0, 1)$ prior for each $\theta_3, \dots, \theta_n$ will make the prior deviation from the mean and slope, also $N(0, 1)$ (in an average sense). So only one common scaling of the prior precisions for these parameters of this prior are needed to shrink it more, or to shrink it less. For this reason the prior for θ_j , $j = 3, \dots, n$, is *defined* to be the same as the prior for θ_3 , hence only the prior for θ_3 needs to be specified.

Doing this from within R, we can evaluate the spline at any point within $[l_1, l_n]$, we can use

```
sfun <- splinefun(loc, beta, method = "natural")
new.value <- sfun(new.loc)
```

The functions `inla.scopy.summary` and `INLA::inla.scopy.define` can be consulted for further details.

We can control n and the covariate with `control.scopy` within `f()`,

```
control.scopy = list(  
  covariate = ...,  
  n = 9)
```

where

covariate gives the covariate that is used

n is the number of hyperparameters used in the spline ($n = 2$ or $5 \leq n \leq 15$).

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

The priors for the mean, slope and the deviation from them, are given by the **hyper**-argument. See also the example.

Spesification

doc Create a scopy of a model component

hyper

theta1

hyperid 36101
name mean
short.name mean
initial 1
fixed FALSE
prior normal
param 1 10
to.theta function(x) x
from.theta function(x) x

theta2

hyperid 36102
name slope
short.name slope
initial 0
fixed FALSE
prior normal
param 0 10
to.theta function(x) x
from.theta function(x) x

theta3

hyperid 36103
name spline.theta1
short.name spline
initial 0
fixed FALSE
prior laplace
param 0 10
to.theta function(x) x
from.theta function(x) x

theta4

hyperid 36104
name spline.theta2
short.name spline2
initial 0
fixed FALSE
prior none
param
to.theta function(x) x

```

    from.theta function(x) x
theta5
    hyperid 36105
    name spline.theta3
    short.name spline3
    initial 0
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta6
    hyperid 36106
    name spline.theta4
    short.name spline4
    initial 0
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta7
    hyperid 36107
    name spline.theta5
    short.name spline5
    initial 0
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta8
    hyperid 36108
    name spline.theta6
    short.name spline6
    initial 0
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta9
    hyperid 36109
    name spline.theta7

```

```

    short.name spline7
    initial 0
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta10
    hyperid 36110
    name spline.theta8
    short.name spline8
    initial 0
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta11
    hyperid 36111
    name spline.theta9
    short.name spline9
    initial 0
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta12
    hyperid 36112
    name spline.theta10
    short.name spline10
    initial 0
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta13
    hyperid 36113
    name spline.theta11
    short.name spline11
    initial 0
    fixed FALSE
    prior none

```

```

    param
    to.theta function(x) x
    from.theta function(x) x
theta14
    hyperid 36114
    name spline.theta12
    short.name spline12
    initial 0
    fixed FALSE
    prior none
    param
    to.theta function(x) x
    from.theta function(x) x
theta15
    hyperid 36115
    name spline.theta13
    short.name spline13
    initial 0
    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

pdf scopy

```

Example 1

```
if (FALSE) {
  inla.setOption(smtp = 'taucs', safe = FALSE, num.threads = "1:1")
}

N <- 200
s <- 0.1
x <- 1:N
eta <- 1 + x / N + sin(x * 0.1) * exp(-2*x/N)
y <- eta + rnorm(N, sd = s)
m <- 15

Y <- matrix(NA, N + 1, 2)
Y[1:N, 1] <- y
Y[N+1, 2] <- mean(y)
r <- inla(Y ~ -1 +
  ## this model will just define the 'overall level',
  ## but with one value for each i.
  ## We need this as as can then scale this one with
  ## the spline. We add a point with
  ## the second likelihood to lock-it in place
  f(idx,
    model = "rw1",
    scale.model = TRUE,
    constr = FALSE,
    values = 1:N,
    hyper = list(prec = list(initial = 15,
                              fixed = TRUE))) +
  ## the 'overall level' is scaled by a spline
  f(idx.scopy, scopy = "idx",
    hyper = list(mean = list(param = c(1, 0.1)),
                  slope = list(param = c(0, 0.1)),
                  spline = list(param = c(0, 20))),
    control.scopy = list(covariate = x, n = m)),
  ##
  data = list(idx = c(rep(NA, N), 1),
              idx.scopy = c(1:N, NA),
              x = c(x,1),
              m = m),
  family = c("gaussian", "gaussian"),
  control.family = list(
    list(hyper =
      list(prec = list(
        initial = log(1/s^2),
        fixed = TRUE))),
    list(hyper =
      list(prec = list(
        initial = 15,
        fixed = TRUE))))),
  control.compute = list(config = TRUE, residuals = 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.
beta <- inla.scopy.summary(r, "idx.scopy", range = c(1, N))
```

```

s.mean <- mean(r$summary.random$idx$mean)
lines(beta$x, s.mean * (beta$mean), lwd = 3, col = "blue")
lines(beta$x, s.mean * (beta$mean + 2 * beta$sd), lwd = 2,
      lty = 2, col = "black")
lines(beta$x, s.mean * (beta$mean - 2 * beta$sd), lwd = 2,
      lty = 2, col = "black")

```

Example 2

```

if (FALSE) {
  inla.setOption(smtp = 'taucs', safe = FALSE, num.threads = "1:1")
}
N <- 100
s <- 0.5
x <- 1:N
eta <- 1 + x / N
y <- eta + rnorm(N, sd = s)
m <- 2

Y <- matrix(NA, N + 1, 2)
Y[1:N, 1] <- y
Y[N+1, 2] <- mean(y)
r <- inla(Y ~ -1 +
  ## this model will just define the 'overall level',
  ## but with one value for each i.
  ## We need this as as can then scale this one with
  ## the spline. We add a point with
  ## the second likelihood to lock-it in place
  f(idx,
    model = "rw1",
    scale.model = TRUE,
    constr = FALSE,
    values = 1:N,
    hyper = list(prec = list(initial = 15, fixed = TRUE))) +
  ## the 'overall level' is scaled by a spline
  f(idx.scopy, scopy = "idx",
    hyper = list(mean = list(param = c(1, 0.1)),
      slope = list(param = c(0, 0.1)),
      spline = list(param = c(0, 20))),
    control.scopy = list(covariate = x, n = m)),
  ##
  data = list(idx = c(rep(NA, N), 1),
    idx.scopy = c(1:N, NA),
    x = c(x,1),
    m = m),
  family = c("gaussian", "gaussian"),
  control.family = list(
    list(hyper = list(prec = list(
      initial = log(1/s^2),
      fixed = TRUE))),
    list(hyper = list(prec = list(
      initial = 15,
      fixed = TRUE)))),
  control.compute = list(config = TRUE, residuals = TRUE))

plot(1:N, y, pch = 19, main = "SCOPY")

```

```

## 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.
beta <- inla.scopy.summary(r, "idx.scopy", range = c(1, N))
s.mean <- mean(r$summary.random$idx$mean)
lines(beta$x, s.mean * (beta$mean), lwd = 3, col = "blue")
lines(beta$x, s.mean * (beta$mean + 2 * beta$sd), lwd = 2, lty = 2, col = "black")
lines(beta$x, s.mean * (beta$mean - 2 * beta$sd), lwd = 2, lty = 2, col = "black")

## this is just a check
rr <- inla(y ~ 1 + x,
           data = data.frame(y, x),
           control.family = list(hyper = list(
               prec = list(
                   initial = log(1/s^2),
                   fixed = TRUE))))

inla.dev.new()
plot(1:N, y, pch = 19, main = "y ~ 1 + x")
lines(1:N, rr$summary.linear.predictor$mean, lwd = 3, col = "blue")
lines(1:N, rr$summary.linear.predictor$"0.025quant", lwd = 2,
      lty = 2, col = "black")
lines(1:N, rr$summary.linear.predictor$"0.975quant", lwd = 2,
      lty = 2, col = "black")

```

Notes

This model is experimental.