

Copy of another model component: “copy”

This describes the way to copy another model component with an optional scaling. This issue arise frequently, like with a model like

$$\eta_{ij} = u_i + u_j$$

where u is the **same model component**. This is not doable with normal use of 'formula', hence we introduce an identical (up to numerical error)

$$v = \text{copy}(u)$$

so we can write

$$\eta = u + v$$

We can also enable an optional scaling, so that

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

where β is estimated as well.

Note the following.

- One model component can be copied many times.
- v will inherit the main properties from u , such as **replicate**, **nrep**, **group**, **ngroup** and **values**. So it is natural to use same kind of indexing for v as u . For example

$$y \sim f(i, \text{replicate}=r, \text{group}=g) + f(j, \text{copy}="i", \text{replicate}=rr, \text{group}=gg)$$

Here, the argument **copy**, say that this model component is a copy of "i".

- If the same indexing is **not** used, then the equivalent one-dimensional indexing is used as defined in `inla.idx()`.

Hyperparameters

The optional hyperparameter is β which is **default fixed to** $\beta = 1$.

We can estimate β by setting **fixed=FALSE**, like

$$f(j, \text{copy}="i", \text{hyper}=\text{list}(\text{beta}=\text{list}(\text{fixed}=\text{FALSE})))$$

We can control β , in two ways.

1. We can fix β to be in an interval defined by argument **range=c(r1,r2)**, where $r_1 \leq r_2$

- If $r_1 = r_2$, then there is no restrictions on β .
- If $-\infty < r_1 < r_2 < \infty$, then β is defined to be in the interval (r_1, r_2) , and the prior, initial values are defined on $\tilde{\beta}$, where

$$\beta = r_1 + (r_2 - r_1) \frac{1}{1 + \exp(-\tilde{\beta})}$$

- If $-\infty < r_1 < r_2 = \infty$, then β is defined to be in the interval (r_1, ∞) and the prior, initial values are defined on $\tilde{\beta}$, where

$$\beta = r_1 + \exp(\tilde{\beta})$$

2. We can make β the same as a β in another copy, with argument `same.as`, so that

$$\eta_{ijk} = u_i + \beta u_j + \beta u_k$$

with

```
y ~ f(i) +
f(j, copy="i", hyper=list(beta=list(fixed=FALSE))) +
f(k, copy="i", same.as="j")
```

3. How does a the copy is, is determined by the argument `precision=...`, and the default value is

```
f(j, copy="i", precision=exp(14))
```

meaning that $v = u + \epsilon$, where ϵ is iid zero mean Normal with precision $\exp(14)$.

Spesification

doc Create a copy of a model component

hyper

theta

hyperid 36001

name beta

short.name b

initial 0

fixed TRUE

prior normal

param 1 10

```
to.theta function(x, REPLACE.ME.low, REPLACE.ME.high) {
  return(x) } else if (is.finite(low) && is.infinite(high) &&
  stopifnot(low < high)
  return(log(x - low))
  stop("Condition not yet implemented")
}
```

```
from.theta function(x, REPLACE.ME.low, REPLACE.ME.high) {
  return(x) } else if (is.finite(low) && is.infinite(high) &&
  stopifnot(low < high)
  return(low + exp(x))
  stop("Condition not yet implemented")
}
```

constr FALSE

nrow.ncol FALSE

augmented FALSE

aug.factor 1

aug.constr

n.div.by

n.required FALSE

set.default.values FALSE

pdf copy

Example

Just simulate some data and estimate the parameters back.

```
## simple example to illustrate the use of 'copy'
set.seed(1234)
N <- 100
n <- 50
u <- scale(rnorm(n))
s <- 0.1

i <- sample(1:n, N, replace = TRUE)
j <- sample(1:n, N, replace = TRUE)
k <- sample(1:n, N, replace = TRUE)
y <- u[i] + u[j] + u[k] + rnorm(N, sd = s)

r <- inla(y ~ -1 +
          f(i, values = 1:n) +
          f(j, copy = "i") +
          f(k, copy = "i"),
          data = data.frame(y, i, j, k),
          control.family = list(hyper = list(
                                prec = list(initial = log(1/s^2),
                                              fixed = TRUE))))

plot(u, r$summary.random$i$mean, pch = 19)
abline(a = 0, b = 1, lwd = 3, col = "blue")

## estimate scaling parameters, assuming
## y <- u[i] + beta.j * u[j] + beta.k * u[k] + rnorm(N, sd = s)
## where the true values are beta.j=1 and beta.k=1

rr <- inla(y ~ -1 +
          f(i, values = 1:n) +
          f(j, copy = "i", hyper = list(
                                beta = list(fixed = FALSE))) +
          f(k, copy = "i", hyper = list(
                                beta = list(fixed = FALSE))),
          data = data.frame(y, i, j, k),
          control.family = list(hyper = list(
                                prec = list(initial = log(1/s^2),
                                              fixed = TRUE))))

rr$summary.hyperpar[,c("mean", "sd")]
inla.dev.new()
plot(u, rr$summary.random$i$mean, pch = 19)
abline(a = 0, b = 1, lwd = 3, col = "blue")

## now we assume that we know that beta.k = beta.j, and we estimate
## just beta.j
```

```

rrr <- inla(y ~ -1 +
            f(i, values = 1:n) +
            f(j, copy = "i", hyper = list(
                beta = list(fixed = FALSE))) +
            f(k, copy = "i", same.as = "j"),
            data = data.frame(y, i, j, k),
            control.family = list(hyper = list(
                prec = list(initial = log(1/s^2),
                            fixed = TRUE))))

rrr$summary.hyperpar[,c("mean", "sd")]
inla.dev.new()
plot(u, rrr$summary.random$i$mean, pch = 19)
abline(a = 0, b = 1, lwd = 3, col = "blue")

```

Notes