Scaling IGMRF-models in R-INLA

Sigrunn Holbek Sørbye

Unknown date long time ago

Introduction

Intrinsic Gaussian Markov random fields (IGMRFs) are widely used as priors to model latent dependency structures in `R-INLA`. Examples of this type of models include the `rw1`, `rw2`, `besag`, `besag2`, `bym` and the `rw2d` models, see `www.r-inla.org` for further descriptions. All of these models can be described in terms of having an improper Gaussian density with a scaled precision matrix that reflects the neighbourhood structure of the model. The scaling is represented as a random precision parameter, and applying these models the user has to define a hyperprior for this parameter. This is a rather delicate issue and such hyperpriors are often selected in a rather ad-hoc and subjective manner. One problem is that a specific fixed hyperprior for the precision parameter does not have the same interpretation for different IGMRFs.

The aim of this tutorial is to introduce and demonstrate a recently implemented option in `R-INLA` named `scale.model`, which can be added when specifying the inla formula-call for the different IGMRFs. For example, we might specify the formula as

```
formula = y ~  f(idx, model = "rw2", scale.model = TRUE, hyper=..)
```

By defining `scale.model = TRUE`, the `rw2`-model is scaled to have a generalized variance equal to 1, see Section~ for further explanations. The new option is also available using `control.group`,

```
f(idx, model="..", group=g, control.group = list(model="rw2", scale.model=TRUE))
```

which is relevant for the `rw1`, `rw2` and the `besag`-models. Note that `scale.model = FALSE` by default for all models to make the option backwards-compatible.

The purpose of scaling the models is to ensure that a fixed hyperprior for the precision parameter has a similar interpretation for different types of IGMRFs. In general, IGMRFs can be seen as models that reflect local deviation from an unspecified overall level. More specifically, the models considered here penalize local deviation from a constant level (`rw1`, `besag`, `bym`), a line (`rw2`) or a plane (`rw2d`). A priori, the hyperprior chosen for the precision parameter will influence how large we allow this local deviation to be. It seems reasonable to select hyperpriors such that different random effects in a regression model are assumed to have a similar range for this deviation. By scaling the models, this is achieved by assigning the same fixed hyperprior to the precisions of all IGMRFs in the regression model. If we do this using unscaled IGMRFs (as often done in the literature), the influence using different hyperprior choices is less controllable as the variances of the IGMRFs differ.

The hyperpriors for scaled models will also be invariant to the shape and size of the graph for a specific IGMRF and to different scalings of covariates. By mapping the random precision to the marginal variance of an IGMRF, we also get an alternative interpretation of different parameter choices for a hyperprior, as these can be viewed in terms of the local deviation, see Section~ for further details.

The new option is based on ideas introduced in [**?**] and these ideas have also been applied in [**?**]. Note that the implementation in [**?**] is slightly different as there the hyperprior is assigned to scaled precision parameters, while the current option scales the IGMRFs directly. The results are the same, but the new option makes the ideas in [**?**] very easy to apply. We demonstrate this in Section~, analysing two semiparametric regression

models. Concluding remarks are given in Section~.

Scaling the IGMRFs

An IGMRF can be defined as a vector $x = (x_1, \ldots, x_n)$, having an improper Gaussian density with a sparse precision matrix which is not of full rank. In [**?**], the order of an IGMRF is defined as the rank deficiency of its precision matrix. This implies that a zero-mean IGMRF of kth order is specified by

$$\pi x) = (2\pi)^{-(n-k)/2}(|Q|^*)^{1/2} \exp\{-\frac{1}{2}x^T Q x\},$$

where the precision matrix Q is assumed to be semi-positive definite. $|Q|^*$ denotes the generalized determinant equal to the product of the $n-k$ non-zero eigenvalues of Q. The marginal variances of x are given by the diagonal elements of the generalized inverse matrix Q^*, that is

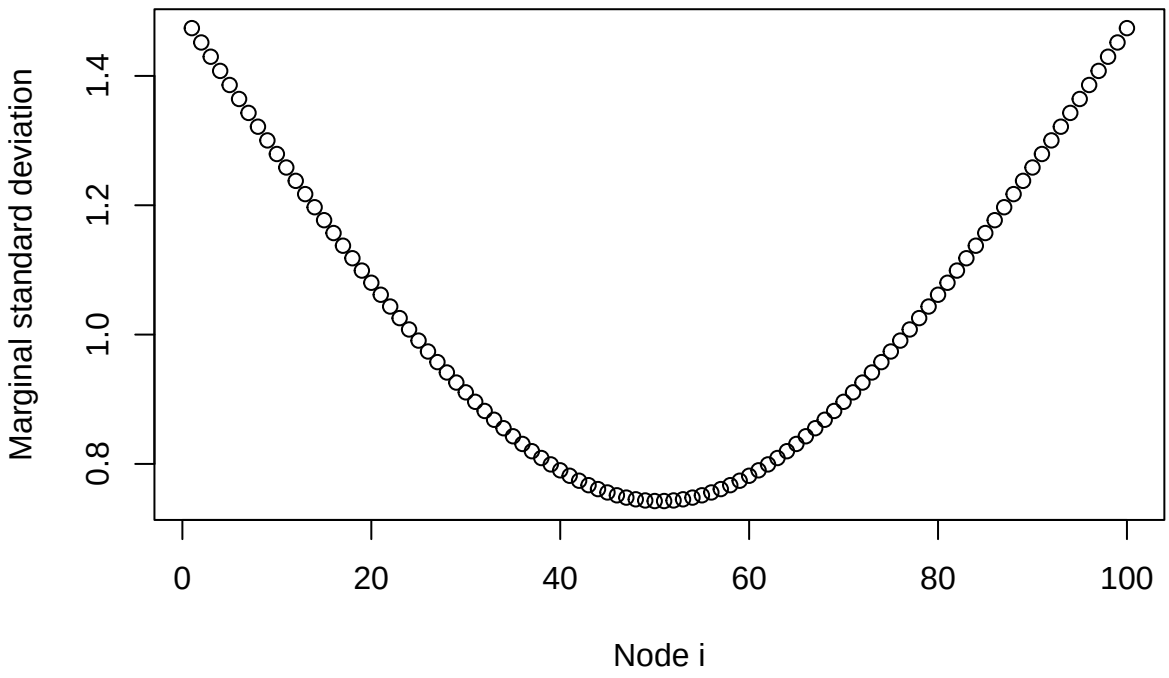
$$\sigma^2(x_i) = Q^*_{ii}, \quad i = 1, \ldots, n.$$

Different IGMRFs can be described by expressing the precision matrix as $Q = \tau R$. Here, τ denotes the random precision parameter while the matrix R reflects the specific neighbourhood structure of the model which can be represented as a labelled graph with nodes and edges. This implies that the marginal variances/standard deviations of the IGMRF will depend on both the structure and size of R. For example, we can consider the marginal standard deviations of the `rw1` and `rw2`-models when the precision is fixed as $\tau = 1$. Due to the different structure matrices of these two models, the levels for the marginal standard deviations are quite different, see Figure~**??**. This illustrates that it seems unreasonable to assign the exact same hyperprior for the precision parameters of these two models when these parameters are random.

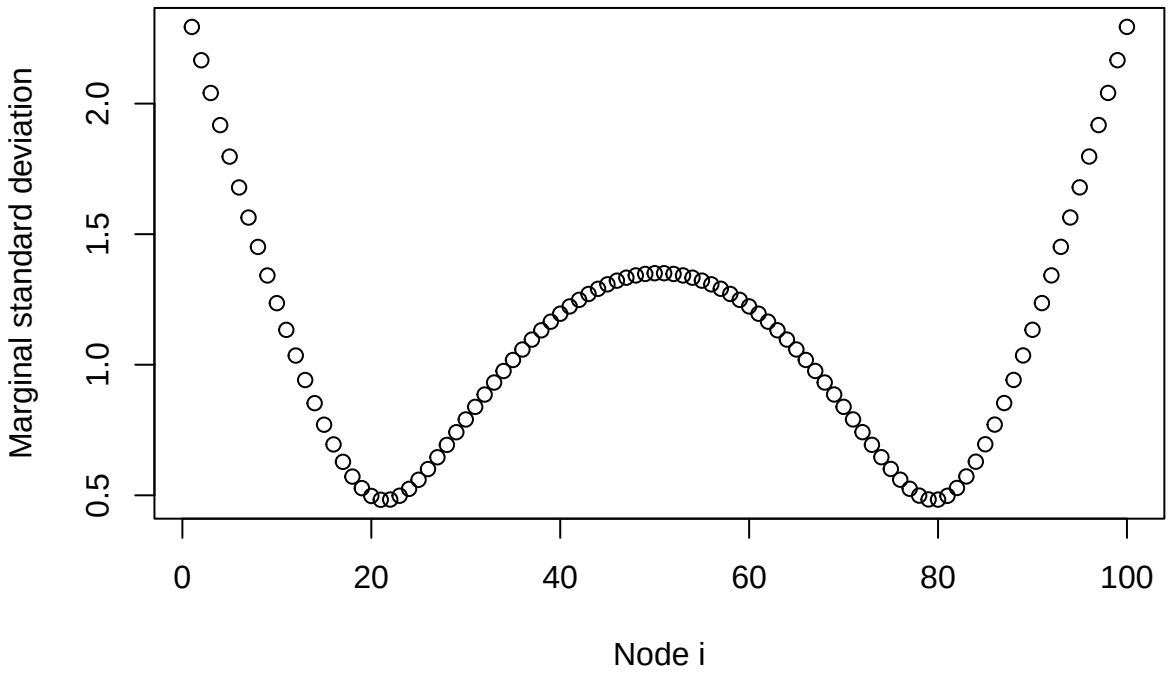
```
ss.rw1.sdref = function(values,tau) {
    n = length(values)
    stopifnot(!missing(n))
    stopifnot(n > 3)
    y = NA

    formula = y ~ -1 + f(values,
            model="rw1",
            constr =TRUE,
            values = values,
            diagonal = 1e-10,
            hyper = list(prec = list(
                                initial = log(tau),
                                fixed=TRUE)))
    r =inla(formula,data=data.frame(y,values),family = "gaussian",
            control.family = list(fixed =
                    TRUE),
            control.compute=list(return.marginals=F),verbose=F)
    sd=r$summary.random$values$sd
    return (sd)
    }

ss.rw2.sdref= function(values,tau) {
    n=length(values)
    stopifnot(!missing(n))
    stopifnot(n > 3)
    y = NA #idx =1:n

    A1 = matrix(1, n, 1)
```

```
    A2 = matrix(0, n, 1)
    for(i in 1:n) {
        A2[i, ]= i
    }

    A = rbind(c(A1), c(A2))
    e = rep(0, 2)
    extraconstr = list(A = A, e= e)

    formula = y ~ -1 + f( values, model="rw2", constr = FALSE,
            values=values, diagonal = 1e-10, extraconstr = extraconstr, hyper
            = list(prec = list(initial = log(tau), fixed=TRUE)))

    r = inla(formula, data = data.frame(y, values), family =
            "gaussian", control.family = list(fixed = TRUE),
            control.compute=list(return.marginals=F),verbose=F)

    sd=r$summary.random$values$sd
    return (sd)
}

n=100
sd.rw1=ss.rw1.sdref(1:n,1)
sd.rw2=ss.rw2.sdref(1:n,1)
```

```
plot(sd.rw1,xlab="Node i", ylab="Marginal standard deviation")
```



```
plot(sd.rw2,xlab="Node i",ylab="Marginal standard deviation")
```



Notice that for a fixed precision τ, the marginal variances of the components of x can be expressed, as a function of τ, by

$$\sigma^2_\tau(x_i) = \frac{\sigma^2_{\{\tau=1\}}(x_i)}{\tau}, \quad i = 1, \ldots, n. \tag{1}$$

To characterize the level of the marginal variances when $\tau = 1$, we define the generalized variance of x as the geometric mean

$$\sigma^2_{\text{GVx}} = \exp\left(\frac{1}{n}\sum_{i=1}^{n} \log(\sigma^2_{\{\tau=1\}}(x_i))\right).$$

When `scale.model = TRUE`, IGMRFs are scaled such that $\sigma^2_{\text{GVx}} = 1$. This implies that the precision parameters of different models have similar interpretation which makes it reasonable to assign the same fixed hyperprior to precision parameters of different IGMRFs in a regression model. Note that in [**?**], the generalized variance was referred to as the reference variance $\sigma^2_{\text{ref x}}$. The hyperprior was then assigned to the scaled precision $\tau/\sigma^2_{\text{ref x}}$ to achieve the same interpretation for different models.

As already mentioned, IGMRFs penalize local deviation from an unspecified overall level. The marginal standard deviation of the models, integrating out the random precision, give information on how large we allow this local deviation to be. Following [**?**], we apply the identity in (1) also when τ is random and define an upper limit U for the marginal standard deviation of a model by

$$P(\sigma(x_i) > U) \approx P\left(\tau < \frac{\sigma^2_{\text{GVx}}}{U^2}\right) = \alpha, \tag{2}$$

where α is a fixed small probability. For scaled models, this implies that the parameter choices for the hyperprior of τ has the same interpretation in terms of U, for all models x.

In R-INLA, precision parameters are assigned Gamma distributions by default (but any other distribution can be specified by the user). Assume that $\tau \sim \text{Gamma}(a, b)$, where a denotes the shape parameter and b the inverse-scale parameter. Then $b\tau \sim \text{Gamma}(a, 1)$ and (2) implies that

$$\frac{b\sigma^2_{\text{GVx}}}{U^2} = F^{-1}(\alpha, a, 1), \tag{3}$$

where $F^{-1}(\cdot)$ denotes the inverse cumulative distribution function of the Gamma distribution. For scaled models, we again notice that the interpretation of a and b stays the same for all x and the upper limit U is easily calculated as %%

```
U = sqrt(b/qgamma(alpha,a,1))
```

```
func.u = function(a,b,alpha,sigma.ref) {
    upper.limit = sqrt(b)*sigma.ref/sqrt(qgamma(alpha,a,1))
    return(upper.limit)
}
a=1
b=5*10^{-5}
alpha=0.001
upper.limit=func.u(a,b,alpha,1)
```

For example, if we choose the default parameter values in R-INLA, $a=$1 and $b = 5 \cdot 10^{-5}$, the upper limit equals $U \approx 0.22$ when $\alpha = 0.001$. Notice that the explicit value of U has to be interpreted with care as this depends on the selected value of α. Also, U is not uniquely defined and different combinations of a and b giving the same values of U, do not in general give the same estimated results.

Examples

The new option makes it very easy to compare results using unscaled and scaled models. Here we consider two examples, analysing semiparametric regression models. Other examples are given in [?] and [?].

```
func<-function(x,t)
{
    f.x=x/t+sin(2*pi* x/t)-0.5
    return(f.x)
}

f.est <- function(y,x,a,b,option)
{
    formula = y~1+f(x,model="rw2",hyper=list(prec=list(prior="loggamma",param=c(a,b))),scale.model=opti
    result = inla(formula,family="gaussian",data=data.frame(y,x))
    f.est=result$summary.random$x$mean
    return(f.est)
}
```

```
### Same values of the function, same observations, but  defined on different intervals ###
set.seed(89236)
x2=0:n
x1=x2/100
x3=x2*10

sigma=0.5
f.x=func(x2,length(x2))
y=f.x+rnorm(length(f.x),0,sigma)
```

```
a=1
b=0.00005

mat1=cbind(f.x,f.est(y,x1,a,b,T),f.est(y,x1,a,b,F))
mat2=cbind(f.x,f.est(y,x2,a,b,T),f.est(y,x2,a,b,F))
mat3=cbind(f.x,f.est(y,x3,a,b,T),f.est(y,x3,a,b,F))

# Generalized marginal variances
v1=exp(mean(log(ss.rw2.sdref(x1,1)^2)))
v2=exp(mean(log(ss.rw2.sdref(x2,1)^2)))
v3=exp(mean(log(ss.rw2.sdref(x3,1)^2)))

u1=func.u(a,b,alpha,sqrt(v1))
u2=func.u(a,b,alpha,sqrt(v2))
u3=func.u(a,b,alpha,sqrt(v3))
```

Semiparametric regression with different scalings of the covariate

We first consider a simple semiparametric regression model in which n observations are generated by

$$y_i = f(x_i) + \epsilon_i, \quad i = 1, \ldots, n. \tag{4}$$

The noise terms, ϵ_i, are assumed independent and normally distributed with mean 0 and a fixed standard deviation σ. The aim is to estimate the underlying smooth function $f(.)$. Applying the `rw2`-model as a prior, the formula-call using the scaled model is defined as

```
formula = y ~  f(x, model = "rw2", scale.model = TRUE, hyper=...)
```

where the hyperprior assigned to the precision parameter is specified in `hyper`. Here, we just select the default hyperprior, that is $\tau \sim \text{Gamma}(1, 5 \cdot 10^{-5})$.

Let the unknown true function $f(.)$ be defined as

$$f(x) = \frac{x}{t} + \sin(\frac{2\pi x}{t}) - 0.5, \quad x \in (0, t).$$

We generate $n = 101$ observations based on (4) in which $\sigma = 0.5$ and call `inla`:

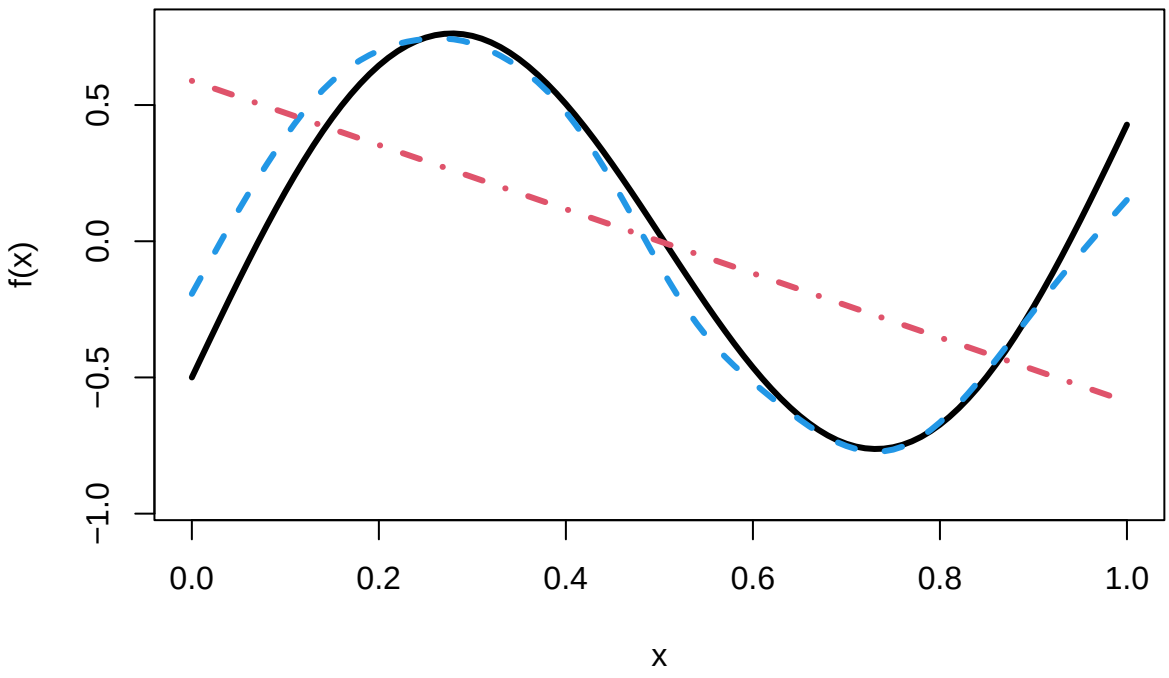
```
result = inla(formula, family = "gaussian", data = data.frame(y,x),verbose=T,safe=F)
```

We then plot the estimated function

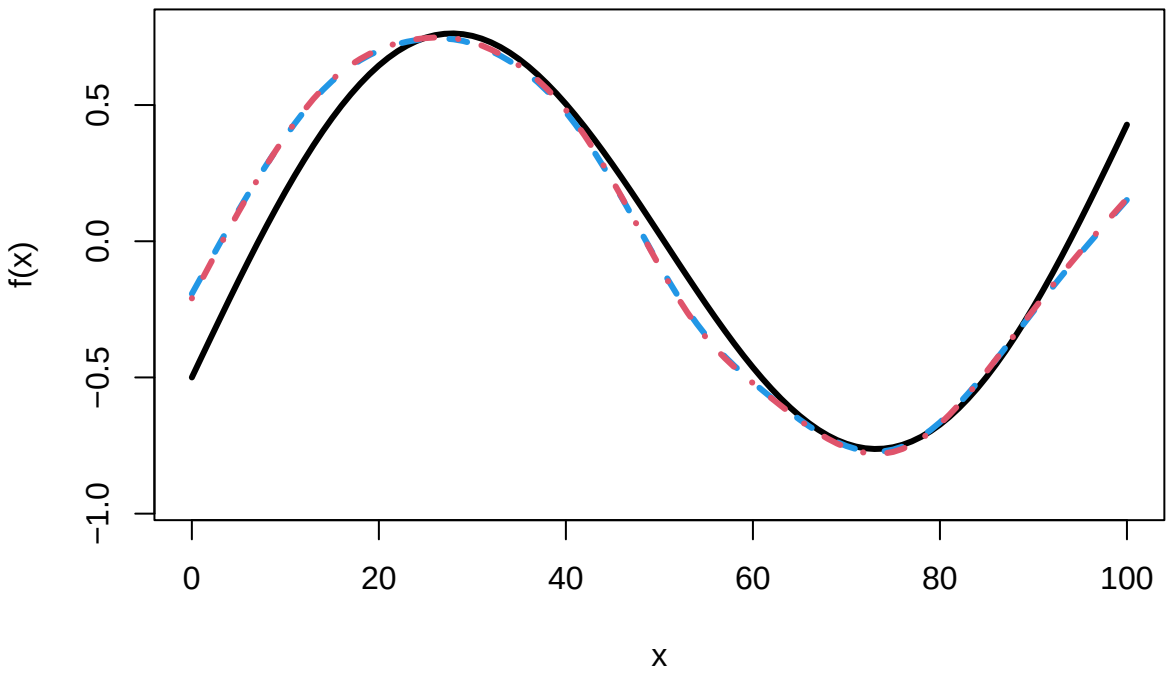
```
result$summary.random$x$mean
```

both using the unscaled and scaled models for three different intervals of x, see Figure **??**.

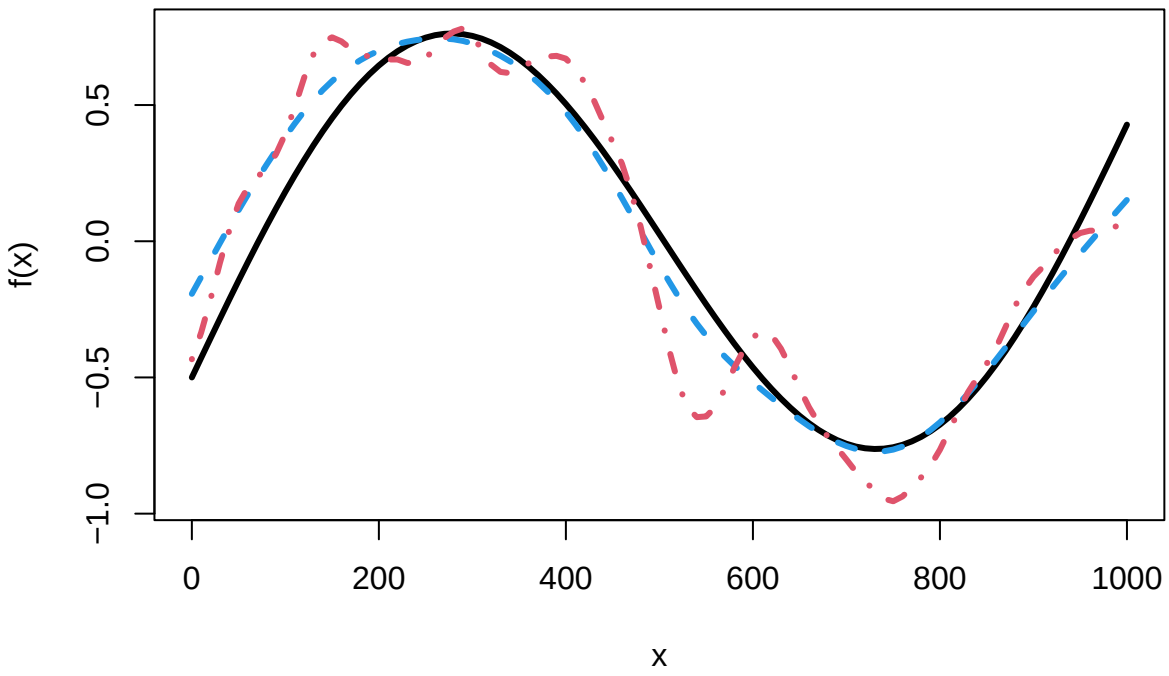
```
r=range(c(mat1,mat2,mat3))
matplot(x1,mat1,type="lll",lty=c(1,2,4),col=c(1,4,2),xlab="x",ylab="f(x)",ylim=r,lwd=3)
```




```
matplot(x2,mat2,type="lll",lty=c(1,2,4),col=c(1,4,2),xlab="x",ylab="f(x)",ylim=r,lwd=3)
```




```
matplot(x3,mat3,type="lll",lty=c(1,2,4),col=c(1,4,2),xlab="x",ylab="f(x)",ylim=r,lwd=3)
```



Using the scaled model, the results using a fixed hyperprior is invariant to different scalings of the covariate and we get exactly the same estimated curve in all three cases. For the unscaled model, the marginal variance will decrease/increase as the distance between nodes decreases/increases. By using the same fixed hyperprior for the three cases, we implicitly assume very different values for the upper limit U in (3), see Table~1. In the extreme case of assuming a very low value of U, the resulting estimate is just a straight line.

Interval length	$t = 1$	$t = 100$	$t = 1000$
U (unscaled model)	0.224	0.2	0.2

Table 1: The upper limit U as defined in (3) for the unscaled `rw2`-model, calculated when $a = 1$, $b = 5 \cdot 10^{-5}$ and $\alpha = 0.001$.

As noted in [**?**] and [**?**], a given hyperprior for the precision of unscaled models can be adjusted to account for different scalings of the covariate, adjusting the inverse-scale parameter of the Gamma distribution. However, by scaling the models, we don't have to think about this at all as a given fixed prior will give the same estimated curve.

Semiparametric regression with several random effects

The linear predictor of a latent Gaussian model often consists of a combination of random effects modelled by different IGMRFs. As an example we will take a look at the Munich rental data analysed in Section 4.2.2 in [**?**], also given as an example in Volume I at `www.r-inla.org`. The response variable in this dataset is rent per square meter (in Euros) for a flat and the covariates include spatial location, floor space, year of construction and various indicator variables. Smooth effects of the covariates are modelled using the `rw2`-model, while the spatially structured effect for location is modelled using the `besag`-model. Previously, this data has been analysed using a fixed Gamma$(1, 0.01)$ for all the precision parameters in the model. The inverse-scale parameter of the Gamma distribution can be adjusted to account for different types of IGMRFs

by scaling this parameter with the generalized variance, see [**?**], but it is easier to just use the new option.

To simplify notation let "'{r echo=TRUE,eval=F} %chunk 17 hyper.prec = list(prec = list(prior = "pc.prec", param = c(0.5, 0.01)))

```
 The formula-call at  \texttt{www.r-inla.org} is then specified by
'''{r echo=TRUE,eval=F} %chunk 18
data(Munich)
g = system.file("demodata/munich.graph", package="INLA")

## Note that here we what to have an estimator of the effect of year
## also the for years where we have no observation, therefore we give a
## vector with all possible values assumed by the covariate year, that
## is seq(1918,2001)

formula = rent ~
   f(location, model = "besag", graph = g, initial = 1, hyper = hyper.prec) +
   f(year, model = "rw2", values = seq(1918,2001), hyper = hyper.prec) +
   f(floor.size, model = "rw2", hyper = hyper.prec) +
   Gute.Wohnlage + Beste.Wohnlage + Keine.Wwv + Keine.Zh +
   Kein.Badkach  + Besond.Bad + Gehobene.Kueche +
   zim1 + zim2 + zim3 + zim4 + zim5 + zim6 -1
```

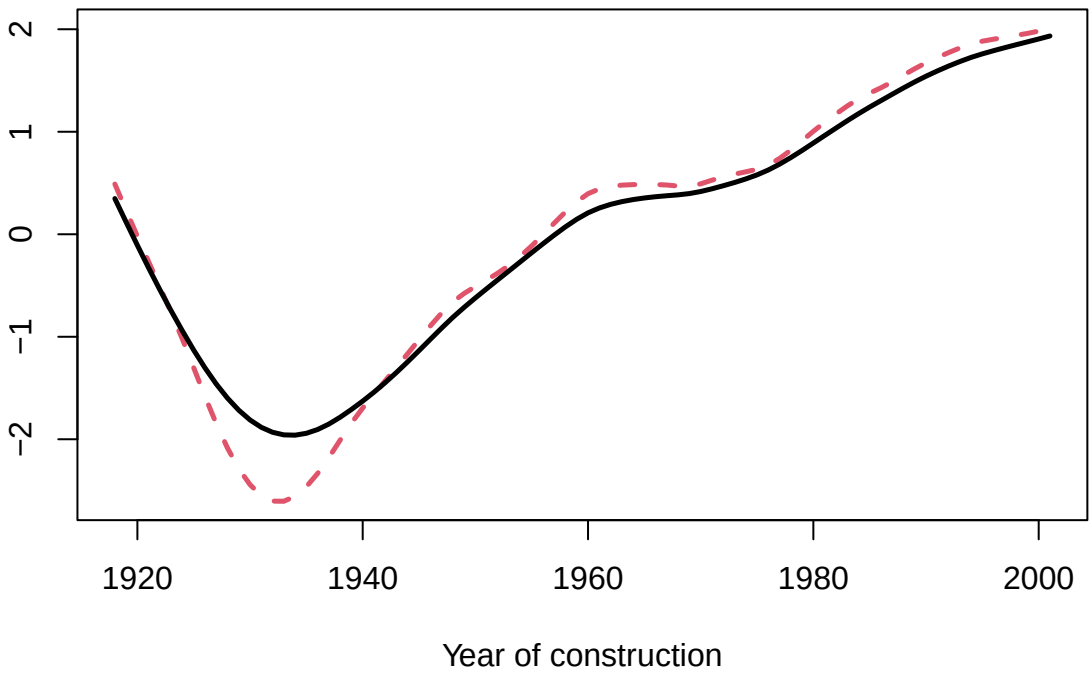
Further, the call to inla is given as {r echo=TRUE,eval=F} %chunk 19 mod = inla(formula, data = Munich, control.fixed=list(prec=1))

```
data(Munich)
g = system.file("demodata/munich.graph", package="INLA")

func.munich <- function(a,b,option)
{
    hyper.prec=list(prec=list(param=c(a,b)))
    formula= rent ~ f(location,model="besag",graph=g,initial=1, hyper=hyper.prec, scale.model=option) +
        f(year,model="rw2",values=seq(1918,2001), hyper=hyper.prec, scale.model=option) +
            f(floor.size,model="rw2", hyper=hyper.prec, scale.model=option) +
                Gute.Wohnlage + Beste.Wohnlage + Keine.Wwv + Keine.Zh +
                    Kein.Badkach  + Besond.Bad + Gehobene.Kueche +
                        zim1 + zim2 + zim3 + zim4 + zim5 + zim6 -1

    return (inla(formula, data=Munich, control.fixed=list(prec=1, prec.intercept = 1)))
}
```

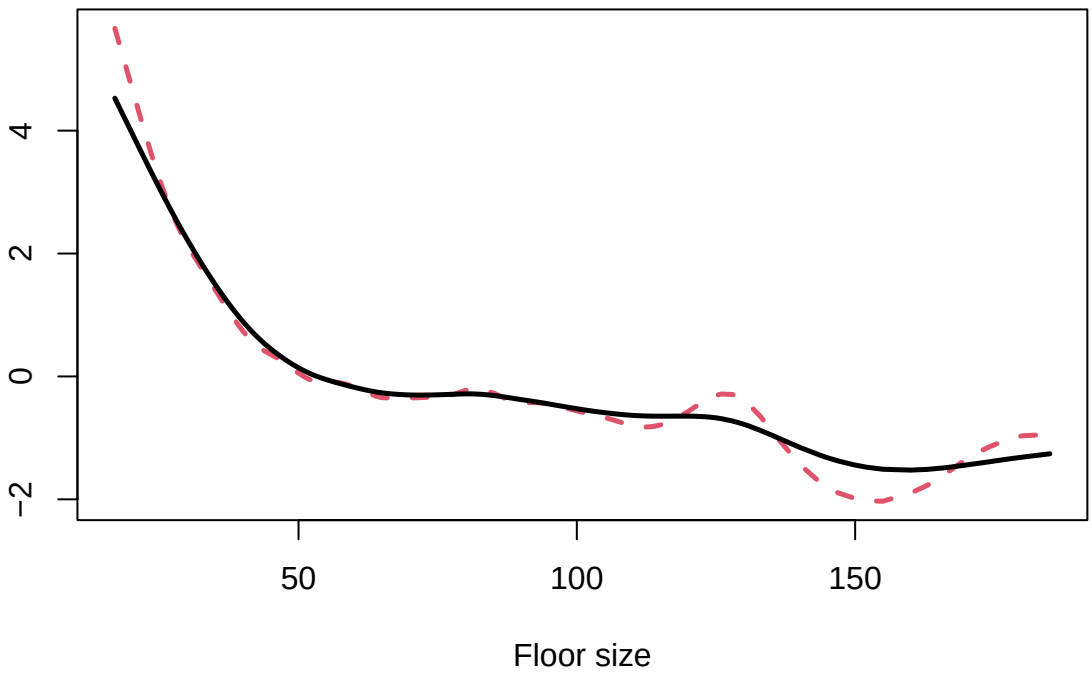
```
a=1
b=0.01
u.munich=func.u(a,b,alpha,1)
mod=func.munich(a,b,FALSE)
mod.scaled=func.munich(a,b,TRUE)
```

Figure~**??** illustrates the resulting estimated effects of the covariates 'Year of construction' and 'Floor size' using the scaled and unscaled models. Applying the given hyperprior for the scaled models, we have assumed that the upper limit in (3) is $U \approx 3.2$. This seems to give quite reasonable results, while the estimated curves are more wiggly using the unscaled models. The results for the spatially structured effects are very similar using the scaled and unscaled models (results not shown).

```
mat=cbind(mod$summary.random$year$mean,mod.scaled$summary.random$year$mean)
matplot(mod$summary.random$year$ID,mat,type="ll",lty=c(2,1),col=c(2,1),xlab="Year of construction",ylab
```



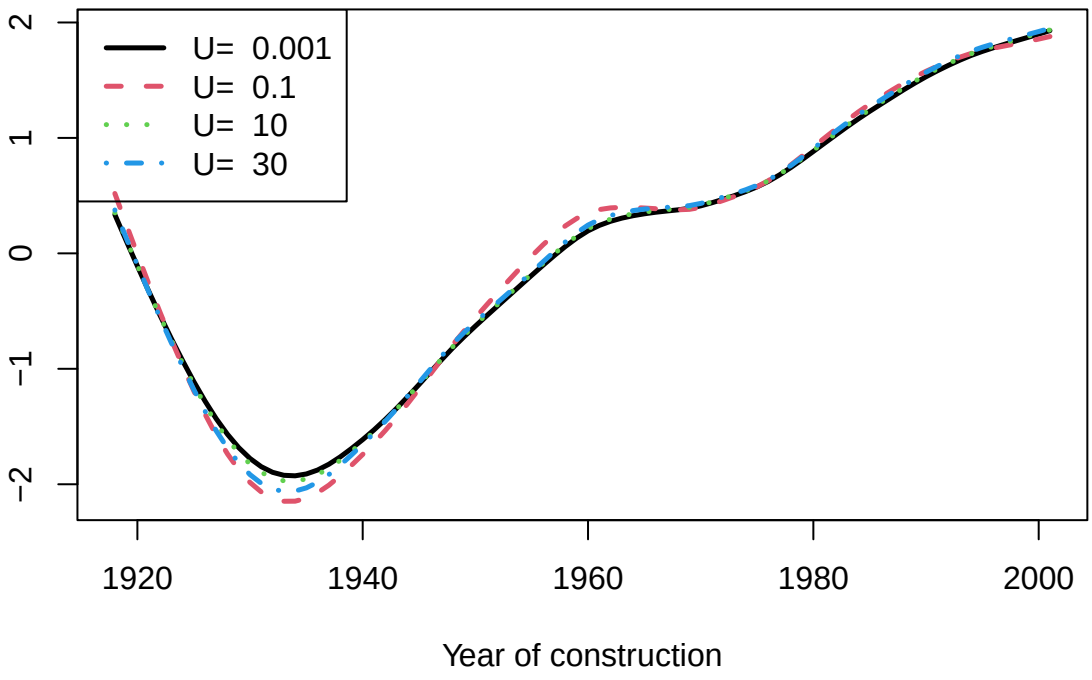

```
mat=cbind(mod$summary.random$floor.size$mean,mod.scaled$summary.random$floor.size$mean)
matplot(mod$summary.random$floor.size$ID,mat,type="ll",lty=c(2,1),col=c(2,1),xlab="Floor size",ylab="",l
```



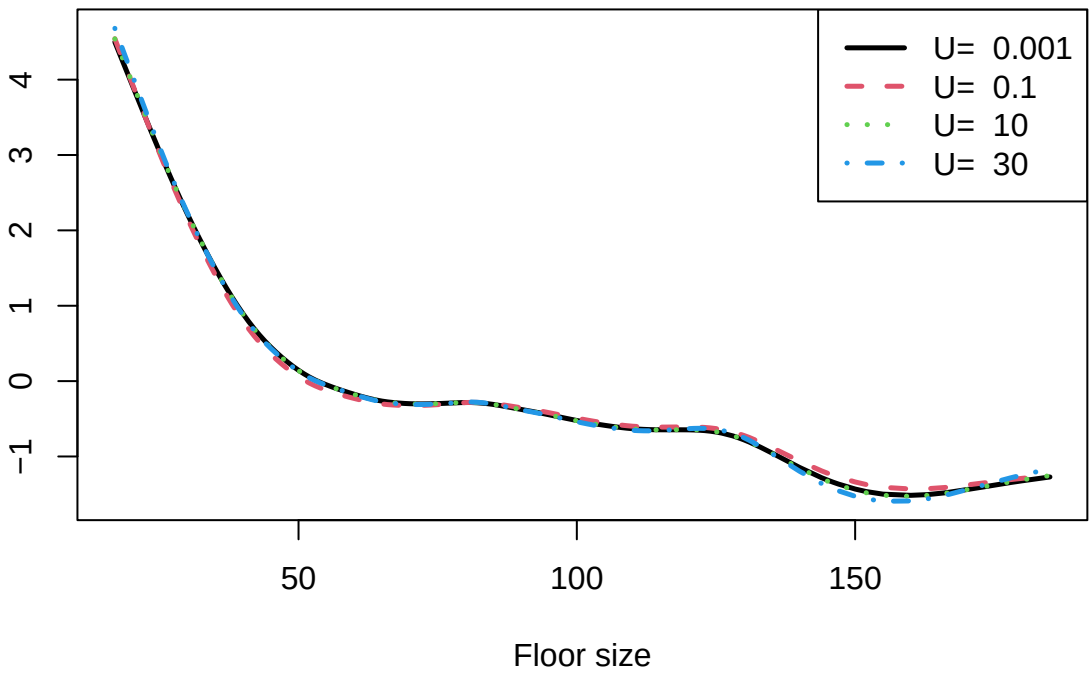

```
alpha=0.001
#u.vec=c(0.01,1,10,100)
u.vec=c(0.001,0.1,10,30)
a.vec=c(1,1,1,1)
b.vec=u.vec^2*qgamma(alpha,a.vec,1)

option=TRUE
c1=func.munich(a.vec[1],b.vec[1],option)
c2=func.munich(a.vec[2],b.vec[2],option)
c3=func.munich(a.vec[3],b.vec[3],option)
c4=func.munich(a.vec[4],b.vec[4],option)
```

Using the scaled models, we can now study sensitivity to the parameters of the Gamma-prior simultaneously for the different random effects modelled by IGMRFs, instead of doing this separately for each of the effects. For example, we can choose to keep the shape parameter fixed at $a = 1$, while the inverse-scale parameter b is calculated to give specific values of U. The estimated functions for the effects of the two covariates 'Year of construction' and 'Floor size' are given in Figure~**??**. Here, the value of U ranges from 0.001 to 30 which corresponds to values of b in the range $(10^{-9}, 0.9)$. We conclude that the results are not very sensitive to changes in the inverse-scale parameter when $a = 1$. This also applies to the spatially structured effect which does not change significantly for these different parameter choices (results not shown). Naturally, we could also consider other values of a.

```
mat=cbind(c1$summary.random$year$mean,c2$summary.random$year$mean,c3$summary.random$year$mean,c4$summary
matplot(mod$summary.random$year$ID,mat,type="llll",lty=c(1,2,3,4),col=c(1,2,3,4),xlab="Year of construc
legend("topleft",paste("U= ",u.vec),lty=1:4,col=1:4,lwd=2.5)
```




```
mat=cbind(c1$summary.random$floor.size$mean,c2$summary.random$floor.size$mean,c3$summary.random$floor.si
matplot(mod$summary.random$floor.size$ID,mat,type="llll",lty=c(1,2,3,4),col=c(1,2,3,4),xlab="Floor size
legend("topright",paste("U= ",u.vec),lty=1:4,col=1:4,lwd=2.5)
```



Concluding remarks

By scaling all IGMRFs to a unit generalized variance, we avoid that the precision parameters of these models have different interpretation. This makes it reasonable to assign the same hyperprior to precision parameters of different IGMRF-models, which greatly simplifies the issue of selecting hyperpriors and performing sensitivity analysis. These ideas were introduced in [**?**], in which the precision parameter was scaled by the generalized variance. The new option in `R-INLA` makes these ideas easy to apply as we don't have to calculate the generalized variance for each model. We simply set the `scale.model`-option equal to `TRUE` in the inla formula-call for the relevant terms.

If you are now convinced that all IGMRF-models should in fact be scaled, you can do this in an even simpler way than already described. By specifying the following global option

```
inla.setOption(scale.model.default = TRUE)
```

all the IGMRF-models are scaled automatically. This also includes the situation using the `control.group` argument mentioned in the introduction. Hyperpriors still have to be selected with care using scaled models, but the issue of selecting hyperpriors is simplified and we believe that hyperpriors can be chosen in a more meaningful and controlled way.