Approximating joint marginals

Cristian Chiuchiolo and Haavard Rue

KAUST, Aug 2020

Introduction

The primary use of `R-INLA` is to approximate univariate marginals of the latent field, so we can compute their marginal summaries and densities. In applications, we sometimes need more than this, as we are interested also in statistics which involve more several of the components of the latent field, and/or, a joint posterior approximation to a subset of the latent field.

The way around this issue, have earlier resolved to stochastic simulation, using the function `inla.posterior.sample`. This function samples from joint approximation to the full posterior which `INLA` construct, but do so for the whole latent field. Using these samples, we can compute the density of the relevant statistics and/or use standard methods to represent a joint marginal.

This vignette introduces a new tool which computes a deterministic approximation to the joint marginal for a subset of the latent field using `R-INLA`. This approximation is explicitly available, and constructed using skew-normal marginals and a Gaussian copula, hence restricted to a joint approximation of a modest dimension.

The key specification is using an argument `selection` in the `inla()`-call, which defines the subset, and then the joint marginal approximation is made available in `result$selection`.

Note that when using the classic-mode, like

```
inla.setOption(inla.mode="classic")
```

then linear predictors can also be used. With the default

```
inla.setOption(inla.mode="compact")
```

then the linear predictor can not be used as it is not explicitly part of the latent model.

Theory reference

For any *Latent Gaussian Model* with data $\boldsymbol{y}$ and set of parameters $(\boldsymbol{x}, \boldsymbol{\theta})$ with $\boldsymbol{x}$ being the latent field and $\boldsymbol{\theta}$ the hyperparameters, the resulting joint posterior approximation is stated as

$$\tilde{\pi}(\boldsymbol{x}, \boldsymbol{\theta}|\boldsymbol{y}) \propto \sum_k \tilde{\pi}_G(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{y}) \tilde{\pi}(\boldsymbol{\theta}_k|\boldsymbol{y}) \Delta_k \tag{1}$$

where $\tilde{\pi}_G(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{y})$ is the Gaussian approximation. This expression recalls a Gaussian mixture distribution with weights $\tilde{\pi}(\boldsymbol{\theta}_k|\boldsymbol{y})\Delta_k$ obtained in the grid exploration of the hyperparameter posterior marginals. For more insights, we suggest checking sources like Rue, Martino, and Chopin (2009), Martins et al. (2013), Blangiardo et al. (2013), or the recent review by Martino and Riebler (2019). The Gaussian approximation used in (1) is both mean and skewness corrected since it exploits Skew-Normal marginals of the latent field into a Gaussian copula structure (see Ferkingstad and Rue (2015) for details). These corrections are available in `inla.posterior.sample` as

```
inla.posterior.sample(..., skew.corr = TRUE)
```

First example

We will illustrate this new feature using a simple example.

```
n = 100
x = rnorm(n, mean = 1, sd = 0.3)
xx = rnorm(n, mean = -2, sd = 0.3)
y = rpois(n, lambda = exp(x + xx))
r = inla(y ~ 1 + x + xx,
         data = data.frame(y, x, xx),
         family = "poisson")
```

Let us compute the joint marginal for the effect of `x`, `xx` and the intercept. This is specified using the argument `selection`, which is a named list of indices to select. Names are those given the formula, plus standard names like `(Intercept)`, `Predictor` and `APredictor`. So that

```
selection = list(Predictor = 3:4, x = 1, xx = 1)
```

say that we want the joint marginal for the 3^{rd} and 4^{th} element of `Predictor` and the first element of `x` and `xx`. (Well, `x` and `xx` only have one element, so then there is not else we can do in this case.)

If we pass `selection`, then we have to rerun `inla()` in classic-mode, as Predictor in the selection is not supported in compact-mode.

```
rs = inla(y ~ 1 + x + xx,
          data = data.frame(y, x, xx),
          family = "poisson",
          inla.mode = "classic",
          control.compute = list(return.marginals.predictor = TRUE),
          control.predictor = list(compute = TRUE),
          selection = selection)
```

We obtain

```
#summary(rs$selection)
print(rs$selection)
```

```
$names
[1] "Predictor:3" "Predictor:4" "x:1"         "xx:1"

$mean
[1] -0.7122315 -1.5151309  1.6615059  1.9552146

$cov.matrix
            [,1]        [,2]       [,3]        [,4]
[1,]  0.08282097  0.02406940  0.1337194 -0.03139123
[2,]  0.02406940  0.05740947 -0.0688537 -0.08589085
[3,]  0.13371939 -0.06885370  0.4469447  0.10015670
[4,] -0.03139123 -0.08589085  0.1001567  0.32337121

$skewness
[1] -0.24584350 -0.26881620 -0.00107341 -0.02538906

$marginal.sn.par
$marginal.sn.par$xi
[1] -0.4732293 -1.3101300  1.7522519  2.1767850

$marginal.sn.par$omega
```

```
[1] 0.3740896 0.3153329 0.6746699 0.6102988

$marginal.sn.par$alpha
[1] -1.3367223 -1.4054081 -0.1710238 -0.5109801
```

The Gaussian copula is given by the `mean` and the `cov.matrix` objects, while the Skew-Normal marginals are given implicitly using the marginal mean and variance in the Gaussian copula and the listed skewness. Moreover the respective Skew-Normal mapping parameters for the marginals (ξ, ω, α) are provided in the object 'marginal.sn.par'. The `names` are given as a separate entry instead of naming each individual result, to save some storage.

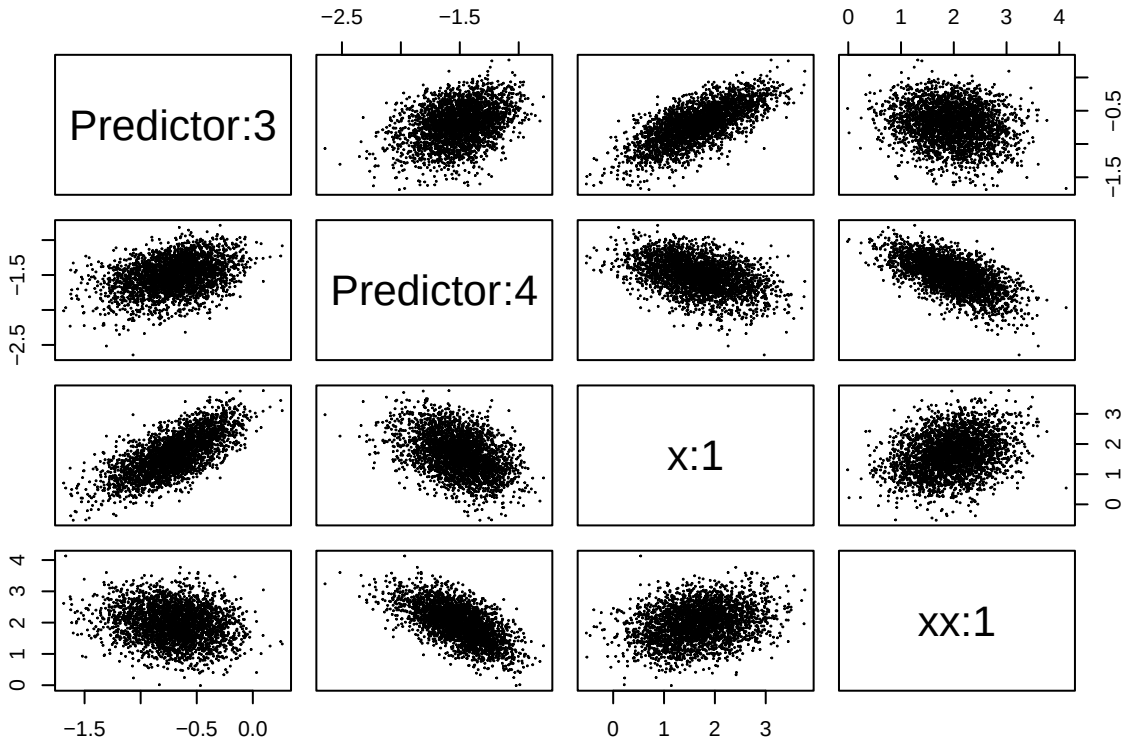
There are utility functions to sample and evaluate samples from this joint marginal, similar to `inla.posterior.sample` and `inla.posterior.sample.eval`.

```
ns = 10000
xs = inla.rjmarginal(ns, rs) ## or rs$selection
str(xs)
```

```
List of 2
 $ samples    : num [1:4, 1:10000] -0.708 -1.371 1.352 1.632 -0.916 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:4] "Predictor:3" "Predictor:4" "x:1" "xx:1"
  .. ..$ : chr [1:10000] "sample:1" "sample:2" "sample:3" "sample:4" ...
 $ log.density: Named num [1:10000] 5.213 4.293 4.439 -0.331 4.86 ...
  ..- attr(*, "names")= chr [1:10000] "sample:1" "sample:2" "sample:3" "sample:4" ...
```

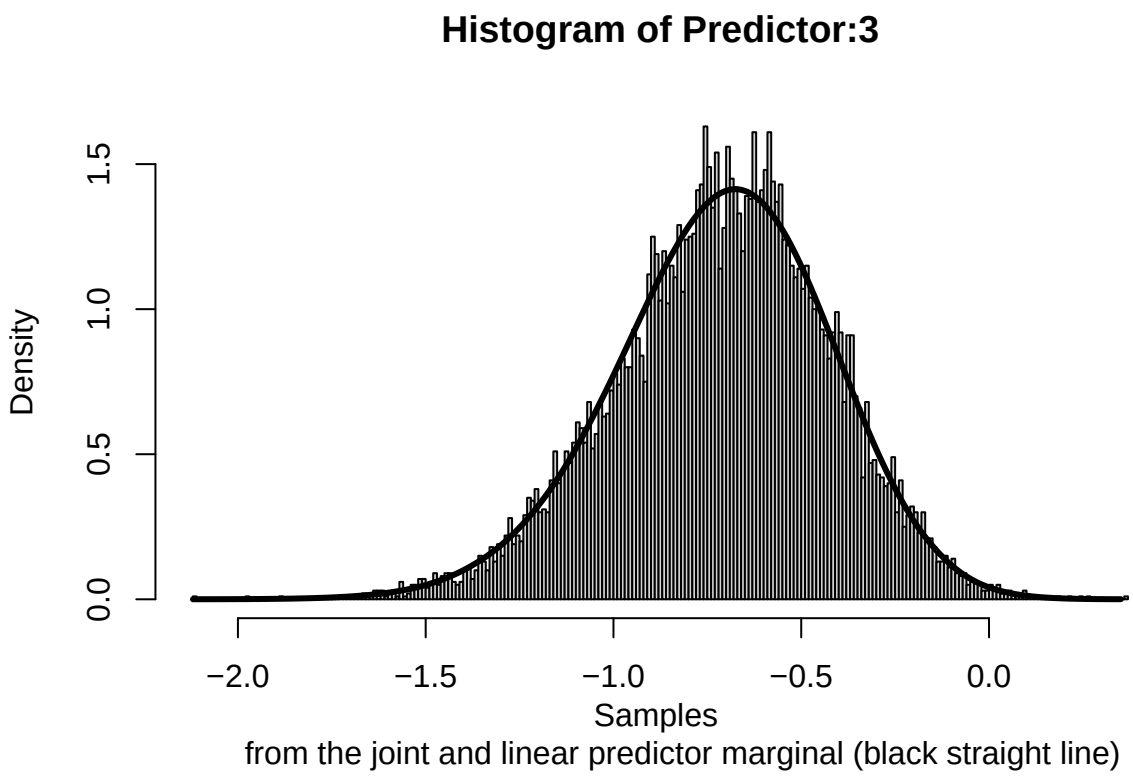
whose output is a matrix where each row contains the samples for the variable in each column

```
pairs(t(xs$samples[, 1:3000]), cex = 0.1)
```



We can compare the approximation of `Predictor:3` to the one computed by the `inla()` call,

```
hist(xs$samples["Predictor:3",], n = 300, prob = TRUE,
     main = 'Histogram of Predictor:3', xlab = 'Samples
     from the joint and linear predictor marginal (black straight line)')
lines(inla.smarginal(rs$marginals.linear.predictor[[3]]), lwd = 3)
```



These marginals are not *exactly* the same (as they are different approximations), but should be very similar.

Deterministic Joint approximation

As a conclusion to this vignette we show an additional joint posterior related tool. The following INLA function computes a deterministic approximation for the joint posterior sampler and must be considered experimental. The function is strictly related to the selection type INLA setting. Deterministic posterior marginals for the previous example can be obtained as follows

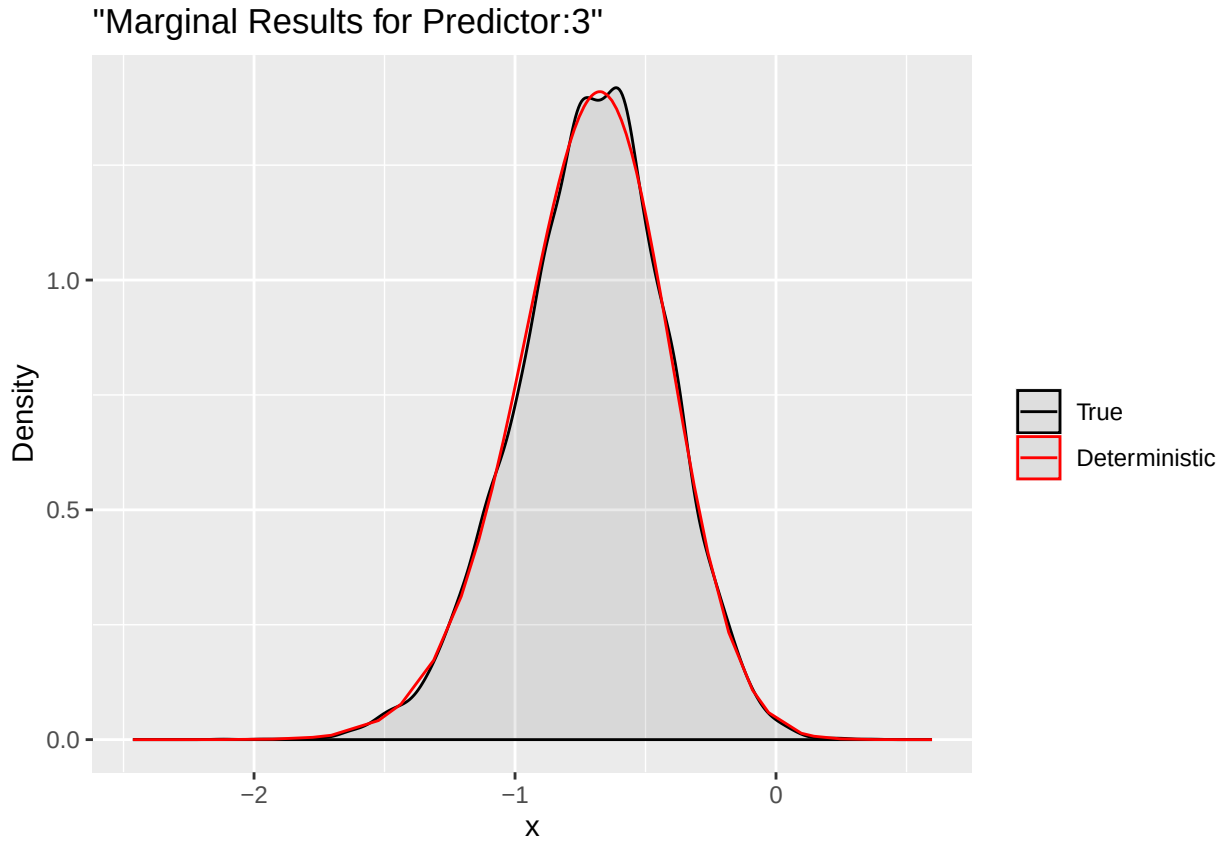
```
dxs = inla.1djmarginal(jmarginal = rs$selection)
str(dxs)
```

```
List of 4
 $ Predictor:3: num [1:63, 1:2] -2.46 -2.3 -2.13 -1.93 -1.78 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "x" "y"
 $ Predictor:4: num [1:63, 1:2] -2.99 -2.85 -2.7 -2.54 -2.41 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "x" "y"
```

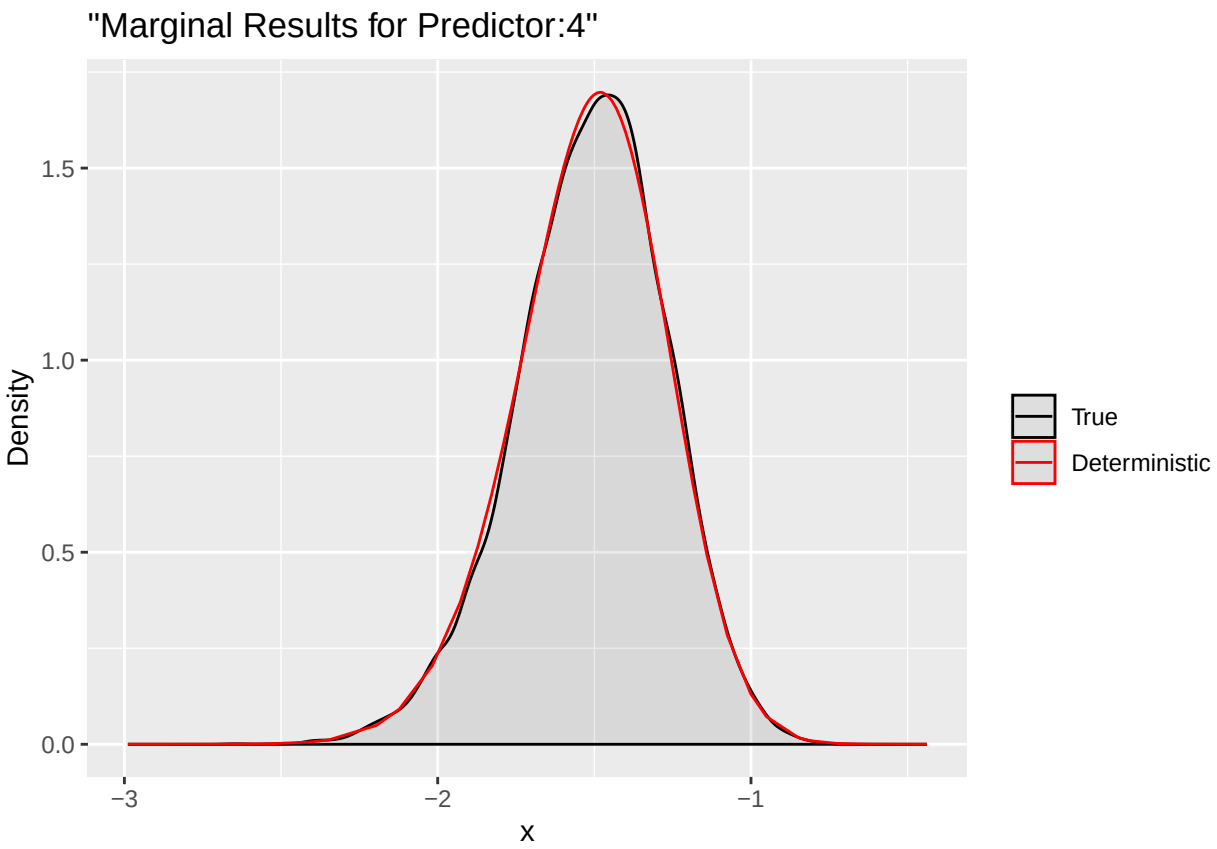
```
$ x:1        : num [1:63, 1:2] -1.818 -1.519 -1.192 -0.826 -0.54 ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : NULL
 .. ..$ : chr [1:2] "x" "y"
$ xx:1       : num [1:63, 1:2] -1.0736 -0.8079 -0.5178 -0.1949 0.0574 ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : NULL
 .. ..$ : chr [1:2] "x" "y"
```

The output is a list with all computed marginals and a matrix summary output in INLA style. Marginal can be accessed and plotted by using the respective names in the selection

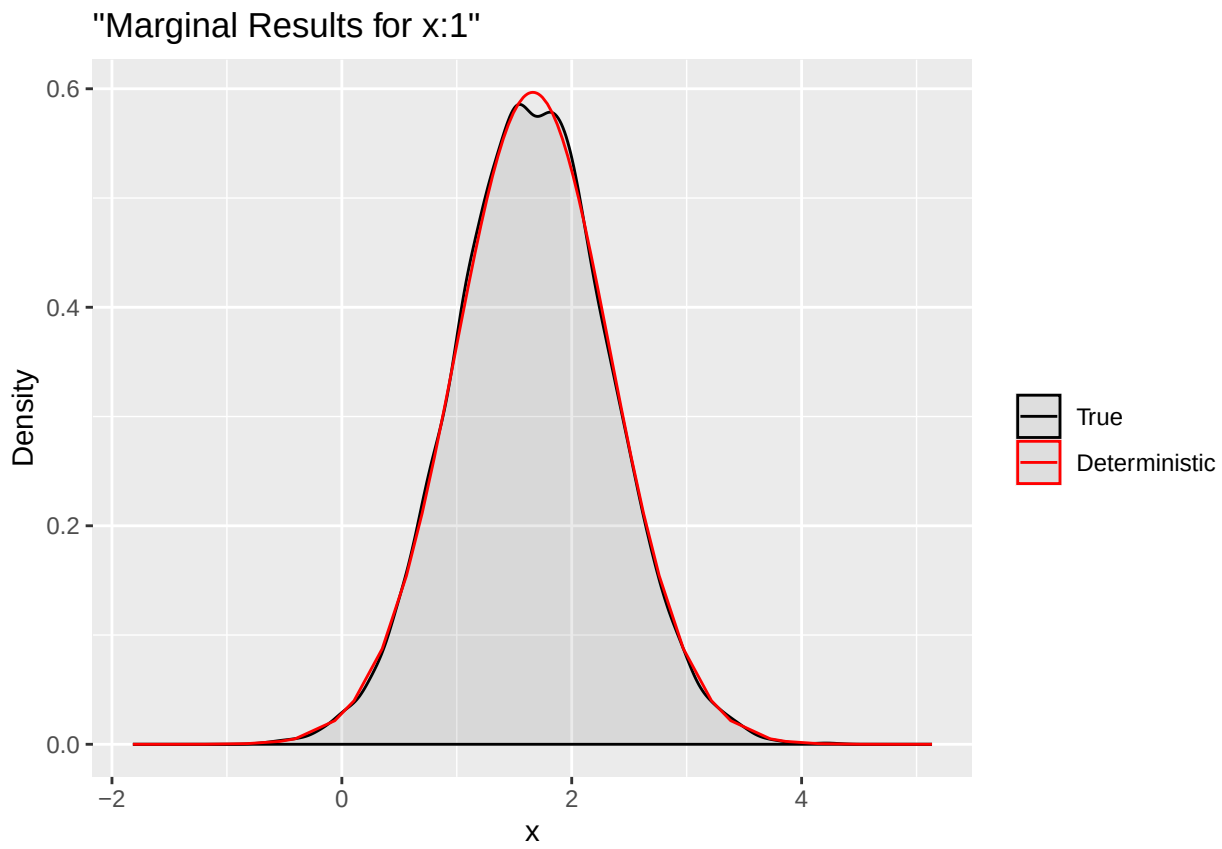
```
ggplot(data = data.frame(y = xs$samples["Predictor:3",]), aes(y, after_stat(density), colour = "True"))
  stat_density(alpha = .1) +
  geom_line(data = as.data.frame(dxs$`Predictor:3`), aes(x = x, y = y, colour = "Deterministic"))+
  labs(title= '"Marginal Results for Predictor:3"', x='x', y='Density') +
  scale_colour_manual("",
                      breaks = c("True", "Deterministic"),
                      values = c("black", "red"))
```



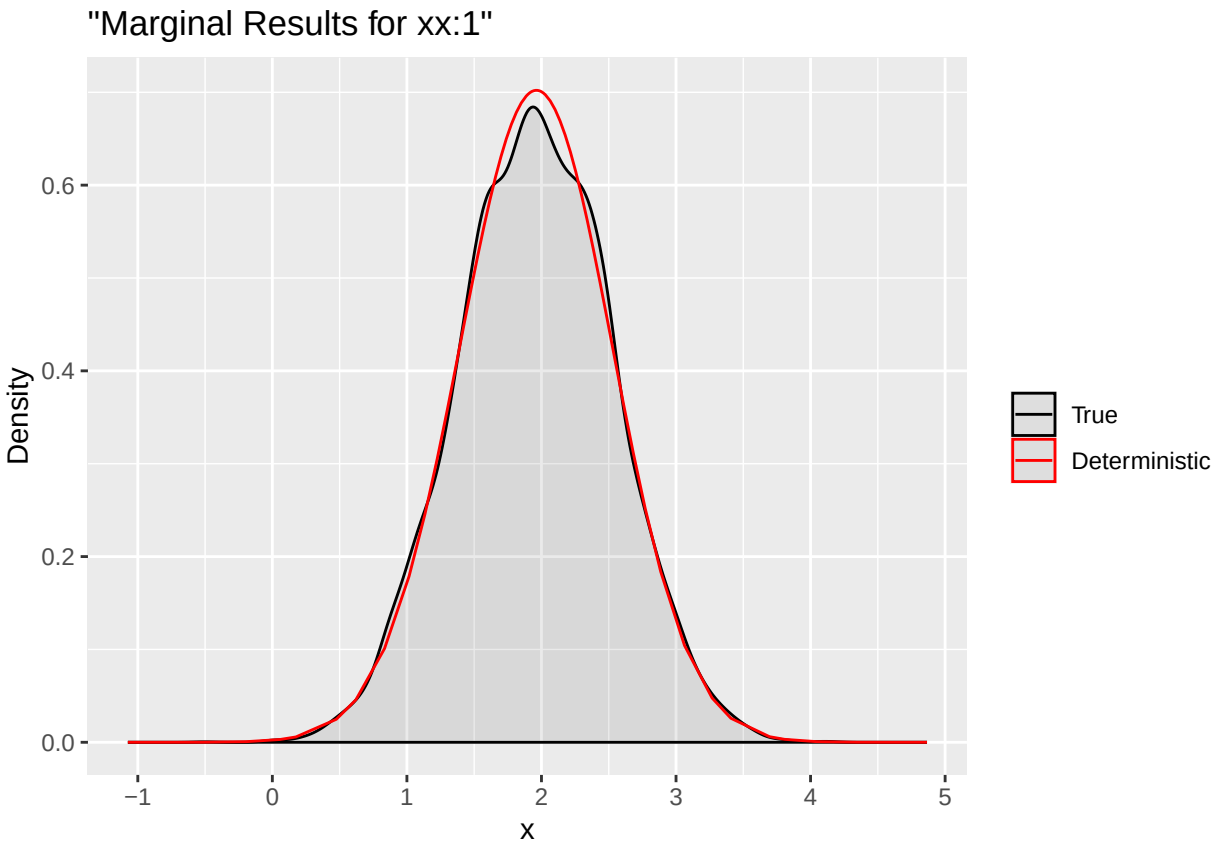

```
ggplot(data = data.frame(y = xs$samples["Predictor:4",]), aes(y, after_stat(density), colour = "True"))
  stat_density(alpha = .1) +
  geom_line(data = as.data.frame(dxs$`Predictor:4`), aes(x = x, y = y, colour = "Deterministic"))+
  labs(title= '"Marginal Results for Predictor:4"', x='x', y='Density') +
  scale_colour_manual("",
                      breaks = c("True", "Deterministic"),
                      values = c("black", "red"))
```




```
ggplot(data = data.frame(y = xs$samples["x:1",]), aes(y, after_stat(density), colour = "True")) +
  stat_density(alpha = .1) +
  geom_line(data = as.data.frame(dxs$`x:1`), aes(x = x, y = y, colour = "Deterministic"))+
  labs(title= '"Marginal Results for x:1"', x='x', y='Density') +
  scale_colour_manual("",
                      breaks = c("True", "Deterministic"),
                      values = c("black", "red"))
```




```
ggplot(data = data.frame(y = xs$samples["xx:1",]), aes(y, after_stat(density), colour = "True")) +
  stat_density(alpha = .1) +
  geom_line(data = as.data.frame(dxs$`xx:1`), aes(x = x, y = y, colour = "Deterministic"))+
  labs(title= '"Marginal Results for xx:1"', x='x', y='Density') +
  scale_colour_manual("",
                      breaks = c("True", "Deterministic"),
                      values = c("black", "red"))
```



Here we compare the deterministic marginals with their sampling version. They are quite accurate and can provide more informations. Indeed, a complete summary based on these deterministic results is achievable with a personalized `summary` function

```
summary(rs$selection)
```

```
[1] "Joint marginal is computed for: "
                  mean        sd 0.025quant   0.5quant 0.975quant       mode
Predictor:3 -0.7122315 0.2877863 -1.3116415 -0.6999207  -0.181140 -0.6746879
Predictor:4 -1.5151309 0.2396027 -2.0168833 -1.5038648  -1.075660 -1.4806947
x:1          1.6615059 0.6685392  0.3508489  1.6616255   2.971483  1.6618579
xx:1         1.9552146 0.5686574  0.8335860  1.9576274   3.063142  1.9624603
```

where posterior estimates and quantiles are computed for all the selected marginals. Along the same line, we can easily compute multiple deterministic linear combinations through the function `inla.tjmarginal`and a matrix object A with the respective indexes

```
A = matrix(c(1,1,0,0,0,0,1,1), nrow = 2, ncol = 4, byrow = T)
rownames(A) <- c("Test1", "Test2")
A
```

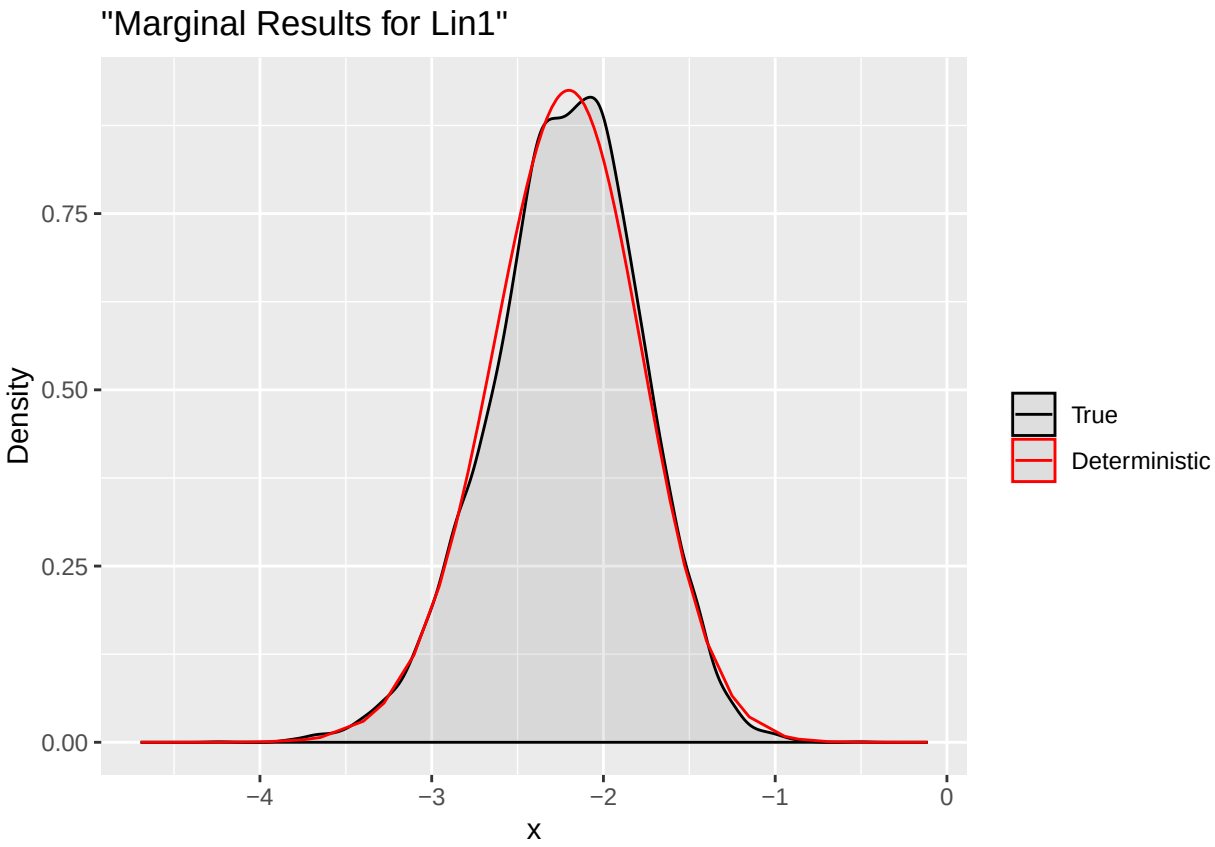
```
      [,1] [,2] [,3] [,4]
Test1    1    1    0    0
Test2    0    0    1    1
```

We define two linear combinations: `Predictor:3+Predictor:4` and `x:1+xx:1` respectively. Then we can use the cited function which has the same class of `selection` object

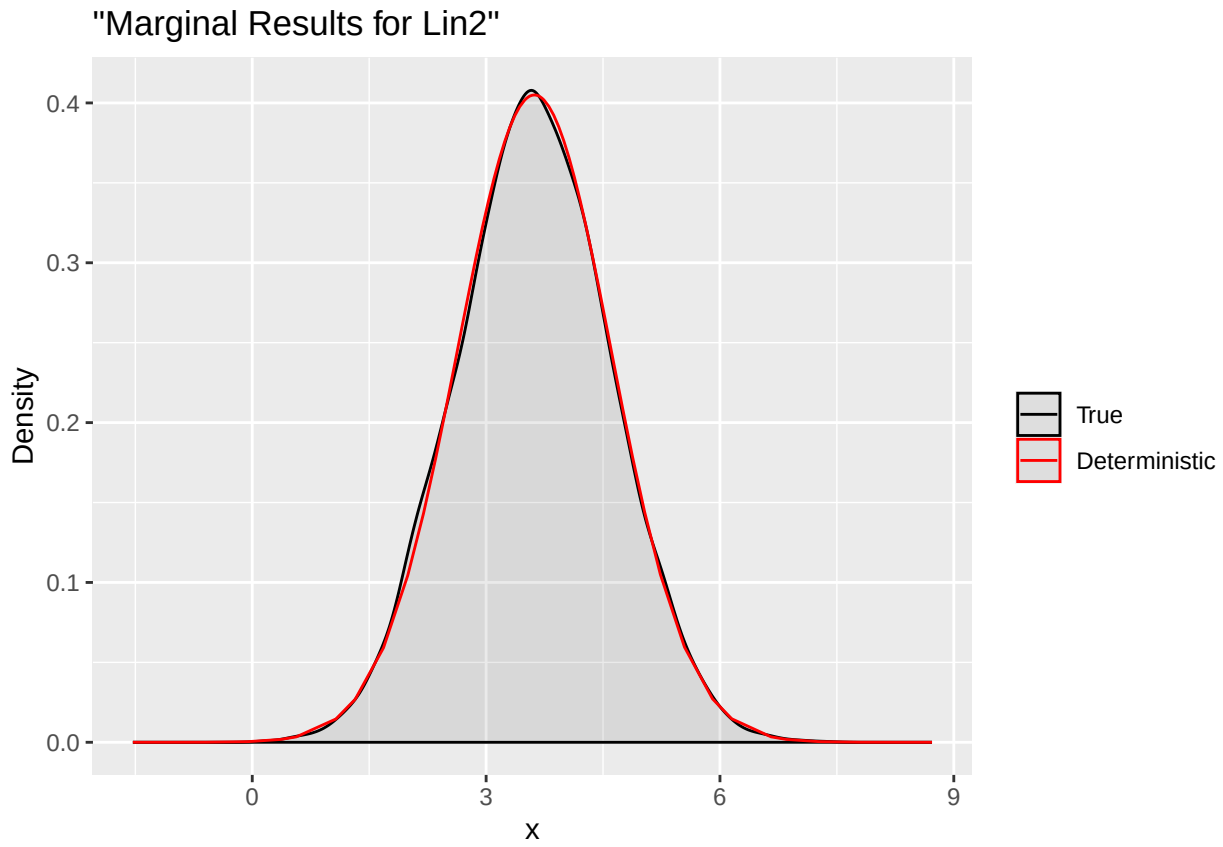
```
m = inla.tjmarginal(jmarginal = rs, A = A)
m
class(m)
```

```
$names
[1] "Test1" "Test2"

$mean
           [,1]
Test1 -2.227362
Test2  3.616720

$cov.matrix
           Test1      Test2
Test1  0.1883692 -0.0524164
Test2 -0.0524164  0.9706293

$skewness
[1] -0.116901652 -0.005217636

$marginal.sn.par
$marginal.sn.par$xi
    Test1     Test2
-1.946027  3.843253

$marginal.sn.par$omega
    Test1     Test2
0.5172221 1.0109135

$marginal.sn.par$alpha
[1] -0.9318074 -0.2926290


[1] "inla.jmarginal"
```

```
dxs.lin = inla.1djmarginal(jmarginal = m)
str(dxs.lin)

fun1 <- function(...) {Predictor[1]+Predictor[2]}
fun2 <- function(...) {x+xx}

xs.lin1 = inla.rjmarginal.eval(fun1, xs)
xs.lin2 = inla.rjmarginal.eval(fun2, xs)

ggplot(data = data.frame(y = xs.lin1[1, ]), aes(y, after_stat(density), colour = "True")) +
  stat_density(alpha = .1) +
  geom_line(data = as.data.frame(dxs.lin$Test1), aes(x = x, y = y, colour = "Deterministic"))+
  labs(title= '"Marginal Results for Lin1"', x='x', y='Density') +
  scale_colour_manual("",
                      breaks = c("True", "Deterministic"),
                      values = c("black", "red"))
```




```
ggplot(data = data.frame(y = xs.lin2[1, ]), aes(y, after_stat(density), colour = "True")) +
  stat_density(alpha = .1) +
  geom_line(data = as.data.frame(dxs.lin$Test2), aes(x = x, y = y, colour = "Deterministic"))+
  labs(title= '"Marginal Results for Lin2"', x='x', y='Density') +
  scale_colour_manual("",
                      breaks = c("True", "Deterministic"),
                      values = c("black", "red"))
```




```
List of 2
 $ Test1: num [1:63, 1:2] -4.69 -4.48 -4.23 -3.96 -3.75 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "x" "y"
 $ Test2: num [1:63, 1:2] -1.5314 -1.0875 -0.6016 -0.0595 0.3657 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "x" "y"
```

and accomplish the job with summaries

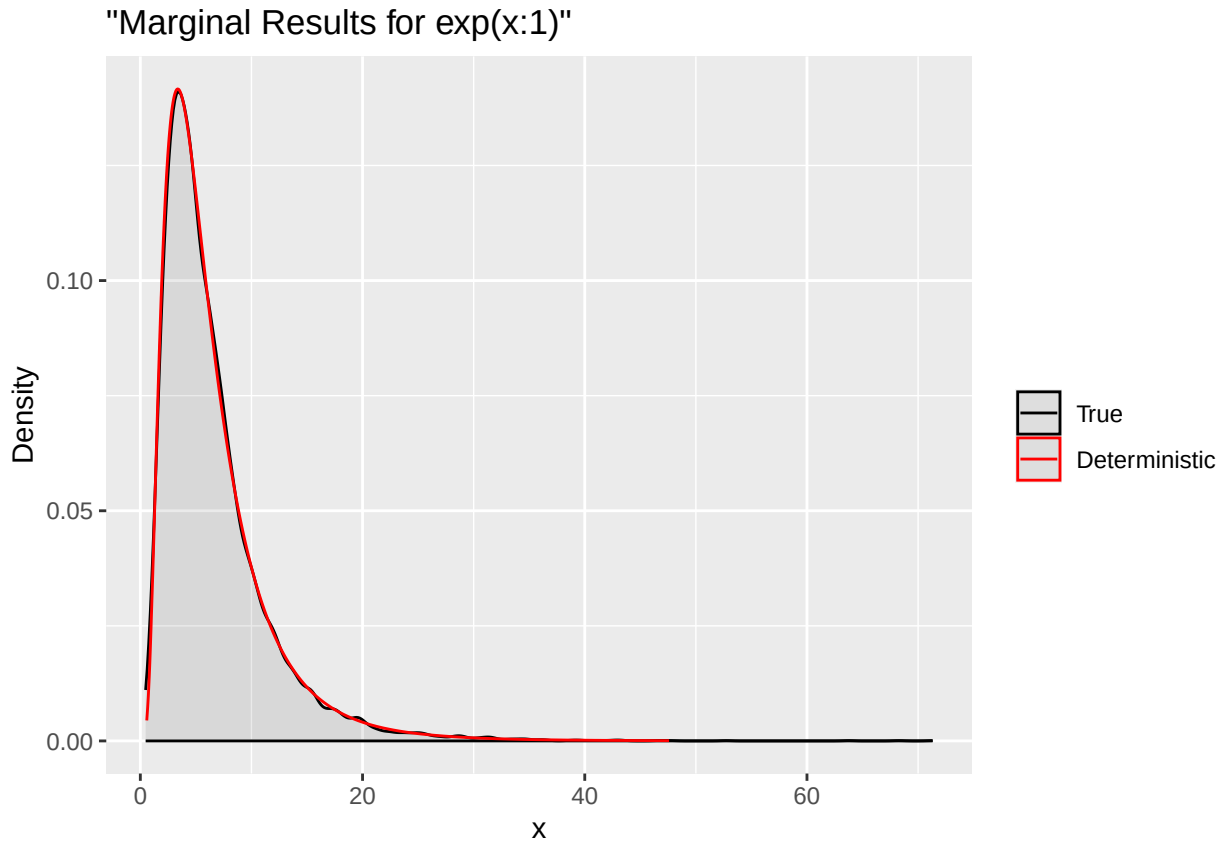
```
summary(m)
```

```
[1] "Joint marginal is computed for: "
           mean        sd 0.025quant  0.5quant 0.975quant      mode
Test1 -2.227362 0.4340152  -3.103351 -2.218762  -1.399910 -2.201415
Test2  3.616720 0.9852052   1.683267  3.617578   5.545304  3.619281
```

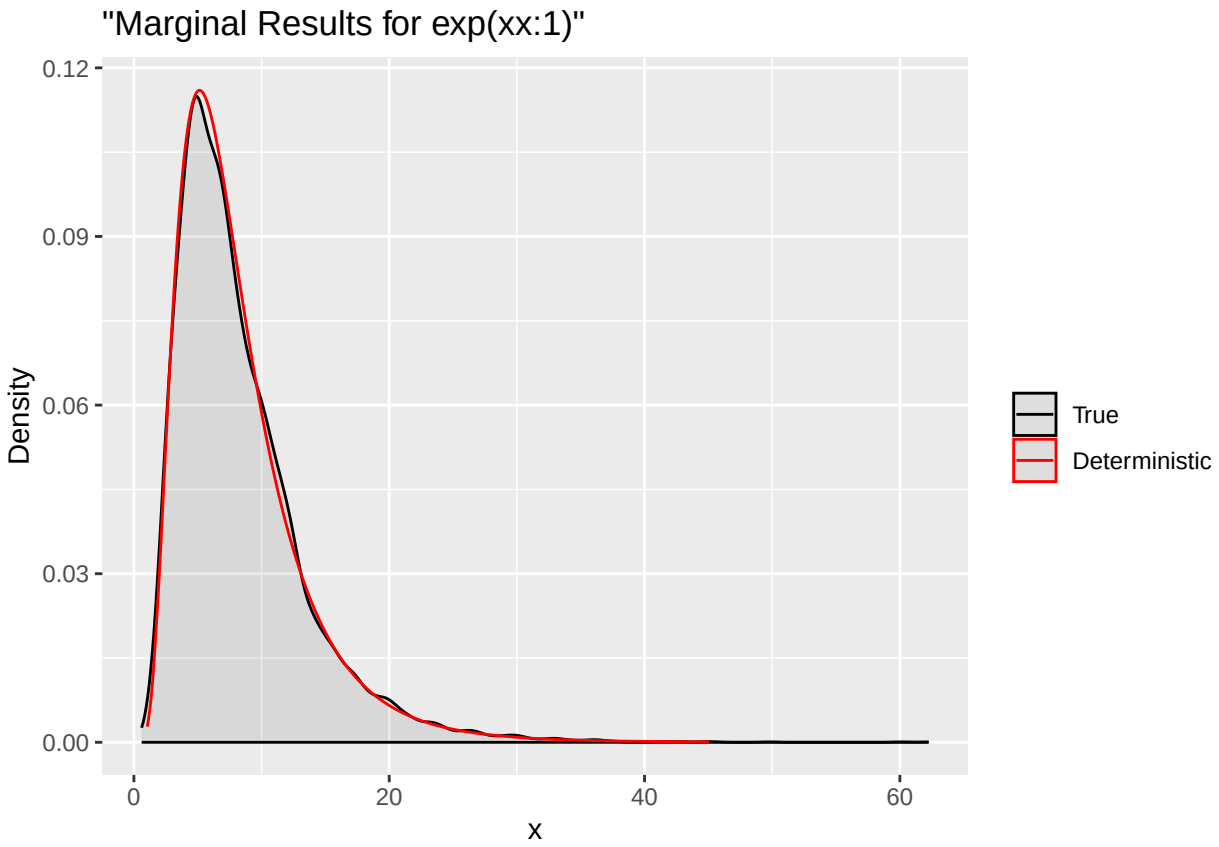
Transformations of the marginal terms or linear combinations are possible as well. We just need to use `inla.tmarginal` as follows

```
fun.exp <- function(x) exp(x)

fun5 <- function(...) {exp(x)}
fun6 <- function(...) {exp(xx)}
fun7 <- function(...) {exp(x+xx)}
```

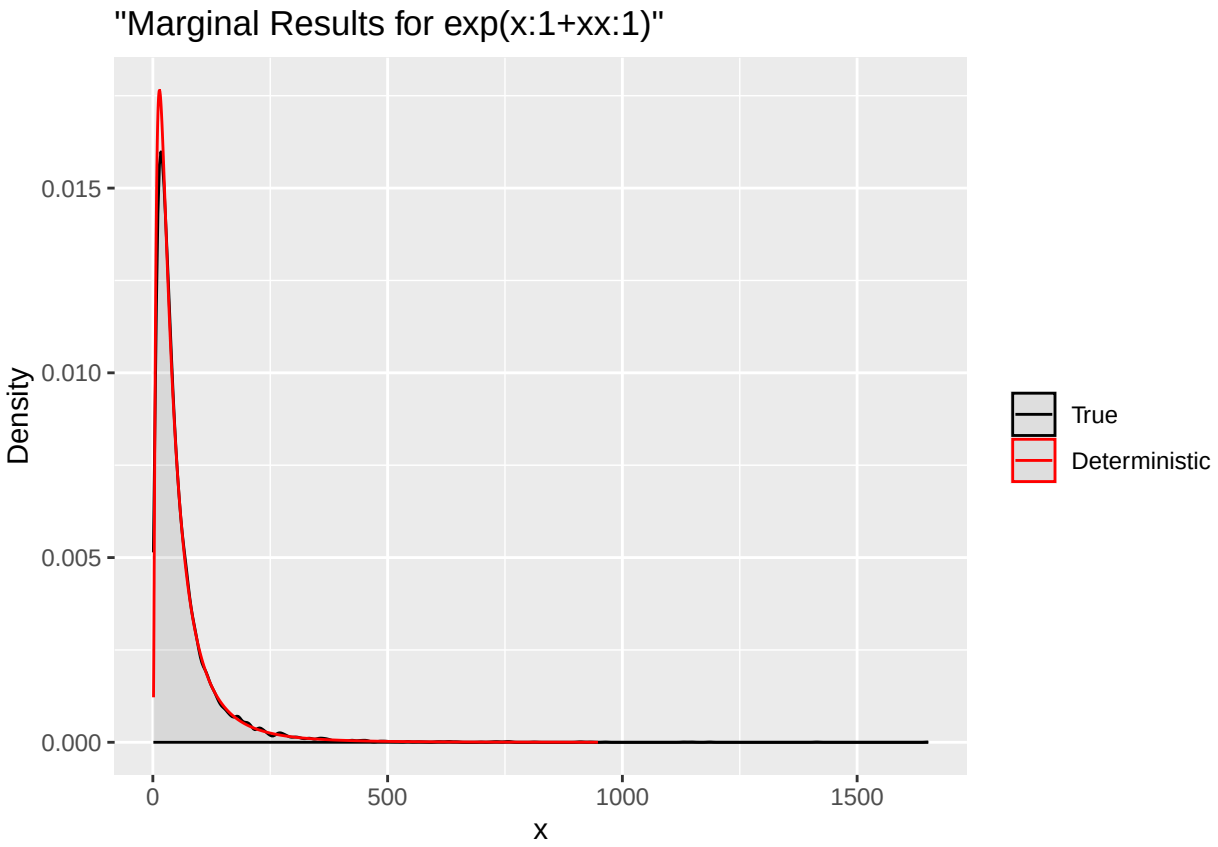
```
tdx <- inla.tmarginal(fun = fun.exp, marginal = dxs$`x:1`)
tdxx <- inla.tmarginal(fun = fun.exp, marginal = dxs$`xx:1`)
tdx.lin <- inla.tmarginal(fun = fun.exp, marginal = dxs.lin$Test2)

tx = inla.rjmarginal.eval(fun5, xs)
txx = inla.rjmarginal.eval(fun6, xs)
tx.lin = inla.rjmarginal.eval(fun7, xs)

ggplot(data = data.frame(y = tx[1, ]), aes(y, after_stat(density), colour = "True")) +
  stat_density(alpha = .1) +
  geom_line(data = as.data.frame(tdx), aes(x = x, y = y, colour = "Deterministic"))+
  labs(title= '"Marginal Results for exp(x:1)"', x='x', y='Density') +
  scale_colour_manual("",
                      breaks = c("True", "Deterministic"),
                      values = c("black", "red"))
```




```
ggplot(data = data.frame(y = txx[1, ]), aes(y, after_stat(density), colour = "True")) +
  stat_density(alpha = .1) +
  geom_line(data = as.data.frame(tdxx), aes(x = x, y = y, colour = "Deterministic"))+
  labs(title= '"Marginal Results for exp(xx:1)"', x='x', y='Density') +
  scale_colour_manual("",
                      breaks = c("True", "Deterministic"),
                      values = c("black", "red"))
```




```
ggplot(data = data.frame(y = tx.lin[1, ]), aes(y, after_stat(density), colour = "True")) +
  stat_density(alpha = .1) +
  geom_line(data = as.data.frame(tdx.lin), aes(x = x, y = y, colour = "Deterministic"))+
  labs(title= '"Marginal Results for exp(x:1+xx:1)"', x='x', y='Density') +
  scale_colour_manual("",
                      breaks = c("True", "Deterministic"),
                      values = c("black", "red"))
```



Summaries for all marginal transformations can be obtained through `inla.zmarginal`

```
expx = inla.zmarginal(marginal = tdx, silent = TRUE)
expxx = inla.zmarginal(marginal = tdxx, silent = TRUE)
expx.lin = inla.zmarginal(marginal = tdx.lin, silent = TRUE)

exp.summaries = rbind(expx, expxx, expx.lin)
exp.summaries
```

```
         mean     sd       quant0.025 quant0.25 quant0.5 quant0.75 quant0.975
expx     6.56273  4.800712 1.416449   3.345773  5.256127 8.252426  19.39269
expxx    8.279957 4.98502  2.301932   4.813478  7.071554 10.36362  21.27653
expx.lin 59.8275  71.62368 5.179072   18.93143  37.00414 72.05395  253.5333
```

References

Blangiardo, M., M. Cameletti, G. Baio, and H. Rue. 2013. "Spatial and Spatio-Temporal Models with R-INLA." *Spatial and Spatio-Temporal Epidemiology* 3 (December): 39–55.

Ferkingstad, E., and H. Rue. 2015. "Improving the INLA Approach for Approximate Bayesian Inference for Latent Gaussian Models." *Electronic Journal of Statistics* 9: 2706–31. https://doi.org/10.1214/15-EJS1092.

Martino, Sara, and Andrea Riebler. 2019. "Integrated Nested Laplace Approximations (INLA)." https://arxiv.org/abs/1907.01248.

Martins, T. G., D. Simpson, F. Lindgren, and H. Rue. 2013. "Bayesian Computing with INLA: New Features." *Csda* 67: 68–83.

Rue, Håvard, Sara Martino, and Nicolas Chopin. 2009. "Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations." *Journal of the Royal Statistical Society: Series b (Statistical Methodology)* 71 (2): 319–92.