# Autoregressive model of order 1 with covariates (AR1C)

## Parametrization

This is an extention of the common autoregressive model (AR1) to include a set of covariates into the conditional mean

$$\begin{aligned} x_1 &\sim \mathcal{N}(0, (\tau(1-\rho^2))^{-1}) \\ x_t &= \rho\, x_{t-1} + \sum_{j=1}^{m} \beta_j z_{t-1}^{(j)} + \epsilon_t; \qquad \epsilon_i \sim \mathcal{N}(0, \tau^{-1}) \qquad t = 2, \ldots, n \end{aligned}$$

where $|\rho| < 1$. The latent vector has length $n + m$ and is represeted as $(x_1, x_2, \ldots, x_n, \beta_1, \ldots, \beta_m)$.

## Hyperparameters

The precision parameter $\kappa$ is represented as

$$\theta_1 = \log(\kappa)$$

where $\kappa$ is the *marginal* precision (when there is no covariates)

$$\kappa = \tau(1-\rho^2).$$

The parameter $\rho$ is represented as

$$\theta_2 = \log\left(\frac{1+\rho}{1-\rho}\right)$$

and the prior is defined on $\theta = (\theta_1, \theta_2)$.

## Specification

The AR1C model is specified as

```
f(<whatever>, model="ar1c", hyper = <hyper>,
  args.ar1c = list(Z=Z, Q.beta = Q))
```

The covariates are given in the **matrix** $Z$ and **must have** dimension $n \times m$[1]. The prior for $\beta = (\beta_1, \ldots, \beta_m)$ is a zero mean Gaussian with a $m \times m$ precision **matrix** `Q`.

### Hyperparameter spesification and default values

**doc** Auto-regressive model of order 1 w/covariates

**hyper**

**theta1**

- **hyperid** 14101
- **name** log precision
- **short.name** prec
- **prior** pc.prec
- **param** 1 0.01
- **initial** 4
- **fixed** FALSE

[1]Despite the fact that the last row of $Z$ is not used

**to.theta** `function(x) log(x)`

**from.theta** `function(x) exp(x)`

**theta2**

**hyperid** 14102

**name** logit lag one correlation

**short.name** rho

**prior** pc.cor0

**param** 0.5 0.5

**initial** 2

**fixed** FALSE

**to.theta** `function(x) log((1 + x) / (1 - x))`

**from.theta** `function(x) 2 * exp(x) / (1 + exp(x)) - 1`

**constr** FALSE

**nrow.ncol** FALSE

**augmented** FALSE

**aug.factor** 1

**aug.constr**

**n.div.by**

**n.required** FALSE

**set.default.values** TRUE

**status** experimental

**pdf** ar1c

## Example

```
n = 500
N = n+1
phi = 0.98
sd.h = 0.4
prec = (1/sd.h)^2
prec.prime = prec / (1-phi^2)
beta = 0.05
h = numeric(n)
y = numeric(n)
z = numeric(n)
s = 0.01
z[n] = NA ## not used
h[1] = rnorm(1, sd = sqrt(1/prec))
y[1] = rnorm(1, sd = sd.h) + rnorm(1, sd = s)
for(i in 2:n) {
    z[i-1] = rnorm(1)
    h[i] = phi * h[i-1] + beta * z[i-1] +
        rnorm(1, sd = sqrt(1/prec.prime))
```

```
    y[i] = h[i] + rnorm(1, sd = s)
}
idx = 1:n

r = inla(y ~ -1 + f(idx, model="ar1c",
                   args.ar1c = list(Z = cbind(z),
                                    Q.beta = matrix(1, 1, 1))),
         data = data.frame(y, idx),
         family = "gaussian",
         control.family = list(
             hyper = list(prec = list(
                              initial = log(1/s^2),
                              fixed=TRUE))))

par(mfrow=c(2, 2))
plot(idx, y, type="l", main = "data")
plot(inla.tmarginal(function(x) sqrt(1/exp(x)),
                    r$internal.marginals.hyperpar[[1]]),
     type = "l", lwd=2, main = "sd(h)")
abline(v = sd.h)

plot(inla.smarginal(r$marginals.hyperpar[[2]]),
     type = "l", lwd=2, main = "phi")
abline(v = phi)

## the N'th element is 'beta'
plot(inla.smarginal(r$marginals.random$idx[[N]]), type="l", lwd=2,
     main = "beta")
abline(v = beta)
```

## Notes

- If $m = 0$, use `model="ar1"`.