

# Constrained Linear

## Parametrization

This model is like a “fixed” effect where you can constrained the coefficient of a covariate to be in an interval:

$$\eta_i = \beta x_i$$

where  $\beta$  is in the interval `[low, high]` and  $x$  are the covariates.

## Hyperparameters

The  $\beta$  parameter, since its is constrained in general, is a hyperparamter. The internal transformation depends on the values of `low` and `high`. If `low` is `-Inf` and `high` is `Inf`, then

$$\beta = \theta$$

and the prior is put on  $\theta$ . If `low` is finite and `high` is `Inf`, then

$$\beta = \text{low} + \exp(\theta)$$

and the prior is put on  $\theta$ . If `low` is finite and `high` is finite, then

$$\beta = \text{low} + (\text{high} - \text{low}) \frac{\exp(\theta)}{1 + \exp(\theta)}$$

and the prior is put on  $\theta$ .

## Specification

```
f(x, model="clinear", range = c(low, high), precision = <precision>)
```

where `precision` is the precision for the tiny noise used to implement this as a latent model.

## Hyperparameter spesification and default values

`hyper`

`theta`

`name` beta

`short.name` b

`initial` 1

`fixed` FALSE

`prior` normal

`param` 1 10

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

```

    from.theta function(x, REPLACE.ME.low, REPLACE.ME.high) {
      stopifnot(low < high)
    } else if (all(is.finite(c(low, high)))) {
      stopifnot(low < high)
    } else if (is.finite(low) && is.infinite(high) && hi
      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** clinear

## Example

```

n = 100
x = runif(n)
y = 1 + x + rnorm(n)
r = inla(y ~ f(x, model = "clinear", range = c(0, Inf)),
  data = data.frame(y,x))
summary(r)

```

## Notes

None