Package 'nimbleMacros'

March 11, 2025

Version 0.1.1

Date 2025-03-10

Title Macros Generating 'nimble' Code

Depends R (>= 3.4.0), nimble

Imports reformulas

Suggests knitr, markdown, testthat

Description Macros to generate 'nimble' code from a concise syntax. Included are macros for generating linear modeling code using a formula-based syntax and for building for() loops. For more details review the 'nimble' manual: <https://r-nimble.org/html_manual/cha-writing-models.html#subsec:macros>.

License BSD_3_clause + file LICENSE | GPL (>= 2)

NeedsCompilation no

URL https://r-nimble.org

BugReports https://github.com/nimble-dev/nimbleMacros/issues

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.3.2

Author Ken Kellner [cre, aut],
Perry de Valpine [aut],
Christopher Paciorek [aut],
Daniel Turek [aut]

Maintainer Ken Kellner <contact@kenkellner.com>

Repository CRAN

Date/Publication 2025-03-11 17:00:07 UTC

Contents

FORLOOP *Macro to build for loop(s) from code with index ranges in brackets*

Description

This macro takes a line of NIMBLE model code with index ranges inside brackets on either the left-hand side of a declaration or both the left- and right-hand sides of a declaration and constructs a corresponding for loop or series of nested for loops.

Arguments

code	The right-hand side of a parameter declaration
modelInfo	Used internally by nimbleMacros; a list of model information such as constants and dimensions
.env	Used internally by nimbleMacros; the environment where the model was created

Value

NIMBLE code for a for loop or series of nested for loops.

Author(s)

Ken Kellner and Perry de Valpine

Examples

```
code <- nimbleCode({
   y[1:n, 1:2, 1] ~ FORLOOP(dnorm(mu[1:n], sigma))
   mu[1:n] <- FORLOOP(beta[1] + beta[2]*x[1:n])
})

mod <- nimbleModel(code, constants = list(n=10))
mod$getCode()
```

LINPRED *Macro to build code for linear predictor from R formula*

Description

Converts an R formula into corresponding code for a linear predictor in NIMBLE model code. Options are available to specify a link function and to also generate code for priors corresponding to the parameters in the linear predictor.

Arguments

formula	An R formula, possibly with the parameters followed by brackets containing indices. If there are no indices, the macro attempts to guess the correct indices from the context. The formula must be right-hand side only (e.g. ~x). This must always be the first argument supplied to LINPRED.
link	A link function that will be applied to the left-hand-side (the response) in the final linear predictor. Default is none.
coefPrefix	All model coefficient names will begin with this prefix. default is "beta_" (so 'x' becomes 'beta_x', etc.)
sdPrefix	All dispersion parameters will begin with this prefix. Default is no prefix.
priors	Prior specifications, generated with setPrior()
modelMatrixNames	Logical indicating if parameters should be named so they match the names one would get from R's model.matrix. Default is FALSE.
noncentered	Logical indicating whether to use noncentered parameterization for random effects. Default is FALSE. Under the noncentered parameterization, random effects have a standard normal prior (beta_x_raw ~ dnorm(0, sd = 1)) and are then scaled by the hyperparameters (mean and SD), i.e., beta_x = mu_beta + sd_beta * beta_x_raw. This parameterization can improve mixing in some cases.
centerVar	Grouping variable (covariate) to 'center' the random effects on. By default (when NULL), random effects come from normal distributions with mean 0 as with lme4. For example, for random intercepts by grouping variable x, the linear predictor would be beta_Intercept + beta_x[x[i]] and the prior for the random effects would be beta_x ~ dnorm(0, sd_x). When centerVar = x, the linear predictor would be beta_x[x[i]] and the random effect prior would be beta_x ~ dnorm(beta_Intercept, sd = sd_x). That is, the mean of the random effects is now beta_Intercept. These two formulations should yield the same results. Note that this option is unrelated to the noncentered argument despite the similar names.
modelInfo	Used internally by nimbleMacros; a list of model information such as constants and dimensions
.env	Used internally by nimbleMacros; the environment where the model was created

Value

NIMBLE code for the linear predictor specified by the formula, and optionally the associated priors.

Author(s)

Ken Kellner

Examples

```
constants <- list(x = rnorm(3), x2 = factor(letters[1:3]))
code <- nimbleCode({
  mu[1:3] <- LINPRED(~x + x2)
})

mod <- nimbleModel(code, constants = constants)
mod$getCode()
```

LINPRED_PRIORS *Macro to build code for priors on a linear predictor from an R formula*

Description

Generates appropriate priors for a linear predictor derived from an R formula. As such it makes the most sense to use this macro together with the LINPRED macro, which takes similar arguments.

Arguments

formula	An R formula The formula must be right-hand side only (e.g., ~x). This must always be the first argument supplied to LINPRED_PRIORS.
coefPrefix	All model coefficient names will begin with this prefix. default is "beta_" (so 'x' becomes 'beta_x', etc.)
sdPrefix	All dispersion parameters will begin with this prefix. Default is no prefix.
priors	List of prior specifications, generated using setPriors. setPriors()
modelMatrixNames	Logical indicating if parameters should be named so they match the names one would get from R's model.matrix. Default is FALSE.
noncentered	Logical indicating whether to use noncentered parameterization for random effects. Default is FALSE. Under the noncentered parameterization, random effects have a standard normal prior (beta_x_raw ~ dnorm(0, sd = 1)) and are then scaled by the hyperparameters (mean and SD), i.e., beta_x = mu_beta + sd_beta * beta_x_raw. This parameterization can improve mixing in some cases.

centerVar	Grouping variable (covariate) to 'center' the random effects on. By default (when NULL), random effects come from normal distributions with mean 0 as with lme4. For example, for random intercepts by grouping variable x, the linear predictor would be beta_Intercept + beta_x[x[i]] and the prior for the random effects would be beta_x ~ dnorm(0, sd_x). When centerVar = x, the linear predictor would be beta_x[x[i]] and the random effect prior would be beta_x ~ dnorm(beta_Intercept, sd = sd_x). That is, the mean of the random effects is now beta_Intercept. These two formulations should yield the same results. Note that this option is unrelated to the noncentered argument despite the similar names.
modelInfo	Used internally by nimbleMacros; a list of model information such as constants and dimensions
.env	Used internally by nimbleMacros; the environment where the model was created

Value

NIMBLE code for the priors specified by the formula.

Author(s)

Ken Kellner

Examples

```
constants <- list(x = rnorm(3), x2 = factor(letters[1:3]))
code <- nimbleCode({
  LINPRED_PRIORS(~x + x2)
})

mod <- nimbleModel(code, constants = constants)
mod$getCode()
```

LM *Macro for fitting linear models, GLMs, and GLMMs*

Description

This macro generates code for LMs, GLMs, and GLMMs using formula notation and arguments similar to R functions such as lm(), glm(), and lmer()/glmer(). Currently only normal, Poisson, and binomial models are supported.

Arguments

formula	An R formula, possibly with the parameters followed by brackets containing indices. If there are no indices, the macro attempts to guess the correct indices from the context. Formulas can include random effects via lme4-style notation (e.g., ~ x + (1|group))

family	A description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. See ?family. Supported families are gaussian (default), binomial, and poisson.
coefPrefix	Character. All model coefficient names will begin with this prefix. default is "beta_" (so 'x' becomes 'beta_x', etc.)
sdPrefix	Character. All dispersion parameters will begin with this prefix. default is no prefix.
priors	List of prior specifications, generated using setPriors().
modelMatrixNames	Logical indicating if parameters be named so they match the names one would get from R's model.matrix.
modelInfo	Used internally by nimbleMacros; a list of model information such as constants and dimensions
.env	Used internally by nimbleMacros; the environment where the model was created

Value

NIMBLE code for the linear model, GLM, or GLMM specified by the formula, including priors.

Author(s)

Ken Kellner

Examples

```
constants <- list(y = rnorm(10),
                  x = rnorm(10),
                  x2 = factor(sample(letters[1:3], 10, replace=TRUE)))

code <- nimbleCode({
   LM(y ~ x + x2)
})

mod <- nimbleModel(code, constants = constants)
mod$getCode()
```

---

matchPrior *Match a prior from a list of prior settings*

Description

Attempts to determine which prior to put on a parameter based on a list of settings, such as the output from setPriors(). The function follows the following search pattern: (1) looks for an exact match to the parameter name including brackets; (2) a match to the parameter name without brackets; (3) goes through each value supplied to ... in order and looks for a match in the names of the settings list. Once a match is found the function returns the corresponding prior value.

Usage

```
matchPrior(parName, ..., priors)
```

Arguments

parName	Parameter to get a prior for, as quoted code/name, possibly including brackets/indices
...	Character strings that categorize the parameter and match the names of elements in priors. The order is important: the first match found is used.
priors	A named list of prior settings, e.g., as generated by setPriors

Value

NIMBLE code for the matching prior.

Author(s)

Ken Kellner

Examples

```
pr <- setPriors(intercept = quote(dunif(-3, 3)), 'alpha' = quote(dunif(0,1)),
                'alpha[2]' = "dnorm(0, 3)")
matchPrior(quote(alpha), priors=pr)
matchPrior(quote(alpha[2]), priors=pr)
matchPrior(quote(intercept), priors=pr)
```

---

setPriors *Set up prior values for different categories of nodes*

Description

Generates a list of custom specifications of priors for parameters in the model. It is possible to set priors for a category of parameters (e.g., intercept, coefficient, sd, factor, continuous) or to set a prior for a specific parameter name (optionally including brackets with indices).

Usage

```
setPriors(
  intercept = quote(dnorm(0, sd = 1000)),
  coefficient = quote(dnorm(0, sd = 1000)),
  sd = quote(dunif(0, 100)),
  factor = NULL,
  continuous = NULL,
  lkjShape = 1,
  ...
)
```

Arguments

intercept	Prior specification for intercepts
coefficient	Prior specfication for slope coefficients
sd	Prior specification for random effects SDs
factor	Prior specifications for slope coefficients corresponding to factor data
continuous	Prior specifications for slope coefficients corresponding to continuous data
lkjShape	Value of shape parameter for LKJ distribution prior, used for correlation matrix in correlated random slope and intercept models
...	Specific parameters, optionally with brackets/indices

Details

Exact name matches including brackets/indices are used first, followed by name matches without indices, followed by data type (factor/continuous) followed by parameter type (intercept/coefficient/sd). Arguments can be supplied as quoted code, a character string, or as a list of prior components. For example, to specify the prior `dnorm(0, sd = 10)` you could specify `quote(dnorm(0, sd = 10))`, or `"dnorm(0, sd = 10)"`, or `list("dnorm", 0, sd = 10)`.

Value

A named list of prior specifications to be passed to the `priors` argument of other macros in the package, such as `LINPRED`.

Author(s)

Ken Kellner

See Also

[nimble::dlkj_corr_cholesky] for more on the LKJ distribution

Examples

```
# Set a prior for intercept terms using quoted code
setPriors(intercept = quote(dunif(-5,5)))
# Instead using a character string
setPriors(intercept = "dunif(-5,5)")
# Set prior for slopes associated with factor covariates
setPriors(factor = quote(dnorm(0, sd = 2.5)))
# Set prior for a specific coefficient
setPriors('alpha[1]' = "dnorm(0, 3)")
```

uppertri_mult_diag *uppertri_mult_diag*

Description

nimbleFunction needed when fitting correlated random effects. Generates upper triangular Cholesky factor of covariance matrix ("U" in code) from upper triangular Cholesky factor of correlation matrix ("Ustar" in code) and vector of standard deviations. Taken from the NIMBLE manual, section 5.2.4.1.2 (LKJ Distribution for Correlation Matrices).

Usage

```
uppertri_mult_diag(mat, vec)
```

Arguments

mat	upper triangular Cholesky factor of correlation matrix ("Ustar")
vec	vector of standard deviations for individual random effects

Value

The upper triangular Cholesky factor of the covariance matrix.

Index