Package ‘mermboost’

May 16, 2025

Type Package

Title Gradient Boosting for Generalized Additive Mixed Models

Version 0.1.1

Author Lars Knieper [aut, cre],
Torsten Hothorn [aut],
Elisabeth Bergherr [aut],
Colin Griesbach [aut]

Maintainer Lars Knieper <`lars.knieper@uni-goettingen.de`>

Description Provides a novel framework to estimate mixed models via gradient boosting. The implemented functions are based on the 'mboost' and 'lme4' packages, and the family range is therefore determined by 'lme4'. A correction mechanism for cluster-constant covariates is implemented, as well as estimation of the covariance of random effects. These methods are described in the accompanying publication; see <doi:10.1007/s11222-025-10612-y> for details.

License GPL-2

Depends R (>= 3.5.0), mboost, lme4

Imports stabs, data.table, MASS, Matrix, parallel, stringr, tibble

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Repository CRAN

Date/Publication 2025-05-16 12:40:04 UTC

Contents

`climbers_sub` *Himalayan Climber Data*

Description

A filtered sub-sample of the Himalayan Database distributed through the R for Data Science TidyTuesday project. This dataset includes information on the results and conditions for various Himalayan climbing expeditions. Each row corresponds to a single member of a climbing expedition team.

Usage

```
data("climbers_sub")
```

Format

A filtered version of the one one from bayesrules. A data frame with 2068 observations (1 per climber) and 19 variables:

expedition_id unique expedition identifier

member_id unique climber identifier

peak_id unique identifier of the expedition's destination peak

peak_name name of the expedition's destination peak

year year of expedition

season season of expedition (Autumn, Spring, Summer, Winter)

sex climber gender identity which the database oversimplifies to a binary category

age climber age

citizenship climber citizenship

expedition_role climber's role in the expedition (eg: Co-Leader)

hired whether the climber was a hired member of the expedition

success whether the climber successfully reached the destination

solo whether the climber was on a solo expedition

oxygen_used whether the climber utilized supplemental oxygen

died whether the climber died during the expedition

injured whether the climber was injured on the expedition

count number of climbers in the expedition

height_metres height of the peak in meters

first_ascent_year the year of the first recorded summit of the peak (though not necessarily the actual first summit!)

Source

Original source: `https://www.himalayandatabase.com/`. Complete dataset distributed by: `https://github.com/rfordatascience/tidytuesday/tree/master/data/2020/2020-09-22/`.

`estuaries` *Effect of pollution on marine microinvertebrates in estuaries*

Description

Data from an observational study of whether there is a different in microinvertebrate communities between estuaries that have been heavily modified by human activity and those that have not, across seven estuaries along the coast of New South Wales, Australia (Clark et al. 2015).

Usage

```
data("estuaries")
```

Format

A dataframe containing (among other things):

Mod	A factor describing whether the sample was taken from a 'Modified' or 'Pristine' estuary.
Zone	Whether the sample was taken from Inner (upstream) or Outer (downstream) zone of the estuary.
Estuary	A factor with seven levels identifying which estuary the sample was taken from.
Total	Total abundance of all invertebrates in the sample
Richness	Richness of taxa in the sample – the number of responses (of those in columns 8-94) taking a non-zero value

Other variables in the dataset give invertebrate counts separately for different taxa.

References

Clark, G. F., Kelaher, B. P., Dafforn, K. A., Coleman, M. A., Knott, N. A., Marzinelli, E. M., & Johnston, E. L. (2015). What does impacted look like? high diversity and abundance of epibiota in modified estuaries. Environmental Pollution 196, 12-20.

Examples

```
data("estuaries")
plot(Total~Estuary,data=estuaries,col=c(4,2,2,4,2,4,2))
```

`find_ccc` *Find cluster-constant covariates in a data set.*

Description

This function gives out logical indicators whether a variable is cluster-constant.

Usage

```
find_ccc(df, id_char)
```

Arguments

`df`	some data frame
`id_char`	a character which is the column name for the cluster identifier.

Details

For uncorrected boosting of a mixed model the estimates for cluster constant covariates might be biased as part of their effect is held in the random effects. This bias is corrected by the underlying mermboost package.

Value

Gives a logical vector that indicates which variables of the dataframe have the same realisation for all observations of one cluster/individual.

See Also

`glmermboost` and `mermboost`

Examples

```
data(Orthodont)
find_ccc(Orthodont, "Subject")
```

`glmermboost` *Component-wise Gradient Boosting for Generalised Mixed Models*

Description

Gradient boosting for optimizing negative log-likelihoods as loss functions where component-wise linear models are utilized as base-learners and an estimation of random components is guaranteed via a maximum likelihood approach.

Usage

```
glmermboost(formula, data = list(), weights = NULL,
          offset = NULL, family = gaussian,
          na.action = na.omit, contrasts.arg = NULL,
          center = TRUE, control = boost_control(), oobweights = NULL, ...)
```

Arguments

`formula`	a symbolic description of the model to be fit in the lme4-format including random effects.
`data`	a data frame containing the variables in the model.
`weights`	an optional vector of weights to be used in the fitting process.
`offset`	a numeric vector to be used as offset (optional).
`family`	!! This is in contrast to usual mboost - "only" a `family` object is possible - except for `NBinomial()`.
`na.action`	a function which indicates what should happen when the data contain `NA`s.
`contrasts.arg`	a list, whose entries are contrasts suitable for input to the `contrasts` replacement function and whose names are the names of columns of `data` containing factors. See `model.matrix.default`.
`center`	logical indicating of the predictor variables are centered before fitting.
`control`	a list of parameters controlling the algorithm. For more details see `boost_control`.
`oobweights`	an additional vector of out-of-bag weights, which is used for the out-of-bag risk (i.e., if `boost_control(risk = "oobag")`).
`...`	additional arguments passed to `mboost_fit`; currently none.

Details

The warning "model with centered covariates does not contain intercept" is correctly given - the intercept is estimated via the mixed model.

A (generalized) linear mixed model is fitted using a boosting algorithm based on component-wise univariate linear models. Additionally, a mixed model gets estimated in every iteration and added to the current fit. The fit, i.e., the regression coefficients and random effects, can be interpreted in the usual way. This particular methodology is described in Knieper et al. (2025).

Value

The description of `glmboost` holds while some methods are newly implemented like `predict.mermboost`, `plot.mer_cv` and `mstop.mer_cv`. Only the former one requires a further argument. Additionally, methods `VarCorr.mermboost` and `ranef.mermboost`are implemented specifically.

See Also

See `mermboost` for the same approach using additive models.

See `mer_cvrisk` for a cluster-sensitive cross-validation.

Examples

```
data(Orthodont)

# are there cluster-constant covariates?
find_ccc(Orthodont, "Subject")


# fit initial model
mod <- glmermboost(distance ~ age + Sex + (1 |Subject),
                   data = Orthodont, family = gaussian,
                   control = boost_control(mstop = 100))

# let mermboost do the cluster-sensitive cross-validation for you
norm_cv <- mer_cvrisk(mod, no_of_folds = 10)
opt_m <- mstop(norm_cv)

# fit model with optimal stopping iteration
mod_opt <- glmermboost(distance ~ age + Sex + (1 |Subject),
                   data = Orthodont, family = gaussian,
                   control = boost_control(mstop = opt_m))

# use the model as known from mboost
# in additional, there are some methods knwon from lme4
ranef(mod_opt)
VarCorr(mod_opt)


#######################

set.seed(123)

# Parameters
n_groups <- 10        # Number of groups
n_per_group <- 50     # Number of observations per group
beta_fixed <- c(0.5, -0.3, 0.7)  # Fixed effects for intercept, covariate1, covariate2
sigma_random <- 1     # Random effect standard deviation

# Simulate random effects (group-specific)
group_effects <- rnorm(n_groups, mean = 0, sd = sigma_random)
```

```
# Simulate covariates
covariate1 <- rnorm(n_groups * n_per_group)
covariate2 <- rnorm(n_groups * n_per_group)

# Simulate data
group <- rep(1:n_groups, each = n_per_group)
random_effect <- group_effects[group]

# Linear predictor including fixed effects and random effects
linear_predictor <- beta_fixed[1] + beta_fixed[2] * covariate1 +
                  beta_fixed[3] * covariate2 + random_effect
prob <- plogis(linear_predictor)  # Convert to probabilities

# Simulate binomial outcomes
y <- rbinom(n_groups * n_per_group, size = 1, prob = prob)

# Combine into a data frame
sim_data <- data.frame(group = group, y = y,
                       covariate1 = covariate1,
                       covariate2 = covariate2)
sim_data$group <- as.factor(sim_data$group)



mod3 <- glmermboost(y ~ covariate1 + covariate2 + (1 | group),
                    data = sim_data, family = binomial())
bin_cv <- mer_cvrisk(mod3, no_of_folds = 10)
mstop(bin_cv)
```

`mermboost` *Gradient Boosting for Additive Mixed Models*

Description

Gradient boosting for optimizing negative log-likelihoods as loss functions, where component-wise arbitrary base-learners, e.g., smoothing procedures, are utilized as additive base-learners. In addition, every iteration estimates random component via a maximum likelihood approach using the current fit.

Usage

```
mermboost(formula, data = list(), na.action = na.omit, weights = NULL,
       offset = NULL, family = gaussian, control = boost_control(),
       oobweights = NULL, baselearner = c("bbs", "bols", "btree", "bss", "bns"),
       ...)
```

Arguments

`formula`	a symbolic description of the model to be fit in the lme4-format including random effects.
`data`	a data frame containing the variables in the model.
`na.action`	a function which indicates what should happen when the data contain `NA`s.
`weights`	(optional) a numeric vector of weights to be used in the fitting process.
`offset`	a numeric vector to be used as offset (optional).
`family`	!! This is in contrast to usual mboost - "only" a `family` object is possible - except for `NBinomial()`.
`control`	a list of parameters controlling the algorithm. For more details see `boost_control`.
`oobweights`	an additional vector of out-of-bag weights, which is used for the out-of-bag risk (i.e., if `boost_control(risk = "oobag")`). This argument is also used internally by `cvrisk`.
`baselearner`	a character specifying the component-wise base learner to be used: `bbs` means P-splines with a B-spline basis (see Schmid and Hothorn 2008), `bols` linear models and `btree` boosts stumps. `bss` and `bns` are deprecated. Component-wise smoothing splines have been considered in Buehlmann and Yu (2003) and Schmid and Hothorn (2008) investigate P-splines with a B-spline basis. Kneib, Hothorn and Tutz (2009) also utilize P-splines with a B-spline basis, supplement them with their bivariate tensor product version to estimate interaction surfaces and spatial effects and also consider random effects base learners.
`...`	additional arguments passed to `mboost_fit`; currently none.

Details

A (generalized) additive mixed model is fitted using a boosting algorithm based on component-wise base-learners. Additionally, a mixed model gets estimated in every iteration and added to the current fit.

The base-learners can either be specified via the `formula` object or via the `baselearner` argument. The latter argument is the default base-learner which is used for all variables in the formula, without explicit base-learner specification (i.e., if the base-learners are explicitly specified in `formula`, the `baselearner` argument will be ignored for this variable).

Of note, `"bss"` and `"bns"` are deprecated and only in the list for backward compatibility.

Note that more base-learners (i.e., in addition to the ones provided via `baselearner`) can be specified in `formula`. See `baselearners` for details.

Value

The description of `mboost` holds while some methods are newly implemented like `predict.mermboost`, `plot.mer_cv` and `mstop.mer_cv`. Only the former one requires an further argument. Additionally, methods `VarCorr.mermboost` and `ranef.mermboost` are implemented specifically.

See Also

See `glmermboost` for the same approach using additive models.

See `mer_cvrisk` for a cluster-sensitive cross-validation.

Examples

```
data("Orthodont")

# are there cluster-constant covariates?
find_ccc(Orthodont, "Subject")


mod <- mermboost(distance ~ bbs(age, knots = 4) + bols(Sex) + (1 |Subject),
                 data = Orthodont, family = gaussian,
                 control = boost_control(mstop = 100))

# let mermboost do the cluster-sensitive cross-validation for you
norm_cv <- mer_cvrisk(mod, no_of_folds = 10)
opt_m <- mstop(norm_cv)

# fit model with optimal stopping iteration
mod_opt <- mermboost(distance ~ bbs(age, knots = 4) + bols(Sex) + (1 |Subject),
                 data = Orthodont, family = gaussian,
                 control = boost_control(mstop = opt_m))

# use the model as known from mboost
# in additional, there are some methods knwon from lme4
ranef(mod_opt)
VarCorr(mod_opt)
```

`mer_cvrisk` *Cluster-sensitive Cross-Validation*

Description

Cross-validated estimation of the empirical risk for hyper-parameter selection. Folds are created cluster-sensitive, hence splitting data into train and tests sets considers the cluster-structure.

Usage

```
mer_cvrisk(object, folds, no_of_folds, cores = 1)
```

Arguments

`object`	an object of class `mermboost`.
`folds`	a weight matrix with number of rows equal to the number of observations. The number of columns corresponds to the number of cross-validation runs. Can be computed using function `cv`.
`no_of_folds`	creates the folds itself by taking the cluster structure into account.
`cores`	is passed on to `mclapply` for parallel computing.

Details

The number of boosting iterations is a hyper-parameter of the boosting algorithms implemented in this package. Honest, i.e., cross-validated, estimates of the empirical risk for different stopping parameters `mstop` are computed by this function which can be utilized to choose an appropriate number of boosting iterations to be applied.

This function uses the cluster-identifier held in the `mermboost` object to split the data into cluster-sensitive folds if the corresponding argument `no_of_folds` is given. As this might lead to imbalanced splits the 1/0 matrix of folds can be given manually via the folds argument.

Value

An object of class `mer_cv`, containing the k-folds as a matrix, the corresponding estimates of the empirical risks, their average and the results optimal stopping iteration. `plot` and `mstop` methods are available.

Examples

```
data(Orthodont)

mod <- mermboost(distance ~ bbs(age, knots = 4) + bols(Sex) + (1 |Subject),
                 data = Orthodont, family = gaussian,
                 control = boost_control(mstop = 100))

# let mermboost do the cluster-sensitive cross-validation for you
norm_cv <- mer_cvrisk(mod, no_of_folds = 10)
opt_m <- mstop(norm_cv)
plot(norm_cv)
```

`methods` *Methods for Gradient Boosting for Mixed Models Objects*

Description

Methods for models fitted by mixed model boosting algorithms.

Usage

```
## S3 method for class 'mermboost'
predict(object, newdata = NULL, RE = TRUE,
        type = c("link", "response", "class"), which = NULL,
        aggregate = c("sum", "cumsum", "none"), ...)
## S3 method for class 'glmermboost'
predict(object, newdata = NULL, RE = TRUE,
        type = c("link", "response", "class"), which = NULL,
        aggregate = c("sum", "cumsum", "none"), ...)
```

```
## S3 method for class 'mermboost'
 ranef(object, iteration = mstop(object), ...)
## S3 method for class 'glmermboost'
 ranef(object, iteration = mstop(object), ...)

## S3 method for class 'mermboost'
VarCorr(x, sigma=1, iteration = mstop(x), ...)
## S3 method for class 'glmermboost'
VarCorr(x, sigma=1, iteration = mstop(x), ...)

## S3 method for class 'mer_cv'
mstop(object, ...)
## S3 method for class 'mer_cv'
plot(x, ...)
```

Arguments

`object`	objects of class `glmermboost` or `mermboost`. If you are using `mstop.mer_cv` it refers to an object resulting from `mer_cvrisk`.
`newdata`	optionally, a data frame in which to look for variables with which to predict. In case the model was fitted using the `matrix` interface to `glmermboost`, `newdata` must be a `matrix` as well (an error is given otherwise). If `RE = TRUE` but not the same cluster-identifier is found in the newdata object, it gets set to FALSE, `RE = FALSE`.
`RE`	a logical values (`TRUE`/`FALSE`) indicating whether to include random effects.
`which`	a subset of base-learners to take into account for computing predictions or coefficients. If `which` is given (as an integer vector or characters corresponding to base-learners) a list or matrix is returned. This ignores the random effects.
`type`	the type of prediction required. The default is on the scale of the predictors; the alternative `"response"` is on the scale of the response variable. Thus for a binomial model the default predictions are of log-odds (probabilities on logit scale) and `type = "response"` gives the predicted probabilities. The `"class"` option returns predicted classes for binomial data.
`aggregate`	a character specifying how to aggregate predictions or coefficients of single base-learners. The default returns the prediction or coefficient for the final number of boosting iterations. `"cumsum"` returns a matrix (one row per base-learner) with the cumulative coefficients for all iterations simultaneously (in columns). `"none"` returns a list of matrices where the jth columns of the respective matrix contains the predictions of the base-learner of the jth boosting iteration (and zero if the base-learner is not selected in this iteration). Therefore, no random effects are considered.
`iteration`	an integer input that specifies from which iteration the random component is to be drawn.
`sigma`	an argument used in `lme4`. Exists for technical reasons but finds no application here.
`x`	a cross-validation object for `plot.mer_cv` or an `mermboost` object for `VarCorr.mermboost`.
`...`	additional arguments passed to callies.

Details

The methods should correspond to equivalent `mboost` and `lme4` functions. However, additional arguments about random effects handling might be of interest.

Value

The `predict.mermboost`-methods give a vector, matrix or a list depending on the arguments.

A matrix with cluster-identifier as rownames and random effects as element results from `ranef.mermboost`.

A `VarrCorr.merMod` is the result of applying `VarCorr.mermboost` to a `mermboost` model.

To deal with cross validtion objects, class `mer_cv`, `mstop.mer_cv` gives a numeric value of the optimal stopping iteration while `plot.mer_cv` plots cross-validation risk-paths.

See Also

`mstop.mer_cv` and `plot.mer_cv`

Examples

```
data(Orthodont)

mod <- glmermboost(distance ~ age + Sex + (1 |Subject),
                   data = Orthodont, family = gaussian,
                   control = boost_control(mstop = 50))

any(predict(mod, RE = FALSE) == predict(mod, RE = TRUE))
all(predict(mod, RE = FALSE) ==
    predict.glmboost(mod) + mod$nuisance()[[mstop(mod)]]$ff
    )
ranef(mod)
VarCorr(mod, iteration = 10)
```

`Orthodont` *Growth curve data on an orthdontic measurement*

Description

The `Orthodont` data frame has 108 rows and 4 columns of the change in an orthdontic measurement over time for several young subjects.

Usage

```
data("Orthodont")
```

Format

This data frame contains the following columns:

- **distance** a numeric vector of distances from the pituitary to the pterygomaxillary fissure (mm). These distances are measured on x-ray images of the skull.
- **age** a numeric vector of ages of the subject (yr).
- **Subject** an ordered factor indicating the subject on which the measurement was made. The levels are labeled `M01` to `M16` for the males and `F01` to `F13` for the females. The ordering is by increasing average distance within sex.
- **Sex** a factor with levels `Male` and `Female`

Details

Investigators at the University of North Carolina Dental School followed the growth of 27 children (16 males, 11 females) from age 8 until age 14. Every two years they measured the distance between the pituitary and the pterygomaxillary fissure, two points that are easily identified on x-ray exposures of the side of the head.

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.17)

Potthoff, R. F. and Roy, S. N. (1964), “A generalized multivariate analysis of variance model useful especially for growth curve problems”, *Biometrika*, **51**, 313–326.

References

Potthoff, R. F. and Roy, S. N. (1964), “A generalized multivariate analysis of variance model useful especially for growth curve problems”, *Biometrika*, **51**, 313–326.

Index