Package ‘fusedTree’

July 11, 2025

Title Fused Partitioned Regression for Clinical and Omics Data

Version 1.0.1

Description Fit (generalized) linear regression models in each leaf node of a tree. The tree is constructed using clinical variables only. The linear regression models are constructed using (high-dimensional) omics variables only. The leaf-node-specific regression models are estimated using the penalized likelihood including a standard ridge (L2) penalty and a fusion penalty that links the leaf-node-specific regression models to one another. The intercepts of the leaf nodes reflect the effects of the clinical variables and are left unpenalized. The tree, fitted with the clinical variables only, should be constructed outside of the package with the 'rpart' 'R' package. See Goedhart and others (2024) <doi:10.48550/arXiv.2411.02396> for details on the method.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports Matrix, splitTools, survival, treeClust

Suggests rpart

NeedsCompilation no

Author Jeroen M. Goedhart [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-0134-1897>),
Wessel N. van Wieringen [aut] (Author of 'porridge' 'R' package, from which many functionalities have been adopted),
Thomas Klausch [ths],
Mark A. van de Wiel [ths],
Hanarth Fonds [fnd]

Maintainer Jeroen M. Goedhart <jeroengoed@gmail.com>

Repository CRAN

Date/Publication 2025-07-11 12:30:10 UTC

Contents

CVfoldsTree 2
Dat_Tree 3
fusedTree 5
PenOpt 7
predict.fusedTree 10

Index **12**

CVfoldsTree *Create balanced cross-validation folds for hyperparameter tuning*

Description

Constructs repeated K-fold cross-validation folds, balanced with respect to the fitted tree structure and outcome (if applicable). The folds contain only the test sample indices. This function is useful for tuning penalty parameters in the fusedTree model.

Usage

```
CVfoldsTree(Y, Tree, Z, model = NULL, kfold = 5, nrepeat = 3)
```

Arguments

`Y` The response variable. Should be:

- Numeric (for linear regression),
- Binary (encoded as 0 and 1, for logistic regression),
- A survival object created using `Surv()` (for Cox regression).

Only right-censored survival data is currently supported.

`Tree` A fitted decision tree, typically created using `rpart`. Trees fitted with other packages (e.g., `partykit`) may be supported in future versions.

`Z` A `data.frame` of clinical variables used to fit the tree. This is used to determine node membership for balancing folds.

`model` Character. Specifies the type of outcome model. Must be one of: `"linear"`, `"logistic"`, or `"cox"`.

`kfold` Integer. Number of folds K for cross-validation. Defaults to 5.

`nrepeat` Integer. Number of times the K-fold cross-validation is repeated. Defaults to 3.

Details

For binary and survival outcomes, the function ensures that the proportion of cases vs. controls (or events vs. censored observations) remains relatively constant across folds. In addition, samples are balanced across the leaf nodes of the fitted tree to ensure consistency in node composition between folds.

Value

A list of length `kfold` × `nrepeat`, where each element contains the test indices for a specific fold. These indices can be used to systematically split the data during cross-validation.

Examples

```
p = 5 # number of omics variables (low for illustration)
p_Clin = 5 # number of clinical variables
N = 100 # sample size
# simulate from Friedman-like function
g <- function(z) {
  15 * sin(pi * z[,1] * z[,2]) + 10 * (z[,3] - 0.5)^2 + 2 * exp(z[,4]) + 2 * z[,5]
}
Z <- as.data.frame(matrix(runif(N * p_Clin), nrow = N))
X <- matrix(rnorm(N * p), nrow = N)            # omics data
betas <- c(1,-1,3,4,2)                         # omics effects
Y <- g(Z) + X %*% betas + rnorm(N)             # continuous outcome
Y <- as.vector(Y)
dat = cbind.data.frame(Y, Z) #set-up data correctly for rpart
rp <- rpart::rpart(Y ~ ., data = dat,
                   control = rpart::rpart.control(xval = 5, minbucket = 10),
                   model = TRUE)
cp = rp$cptable[,1][which.min(rp$cptable[,4])] # best model according to pruning
Treefit <- rpart::prune(rp, cp = cp)
plot(Treefit)
folds <- CVfoldsTree(Y = Y, Tree = Treefit, Z = Z, model = "linear")
```

`Dat_Tree` *Construct design data used for fitting fusedTree models*

Description

Prepares the full data design used to fit a fusedTree model, including dummy-encoded clinical leaf node indicators, optional continuous clinical variables, and a block-diagonal omics matrix structured per tree node.

Usage

```
Dat_Tree(Tree, X, Z, LinVars = TRUE)
```

Arguments

`Tree`	A fitted tree object, created using `rpart`. Support for trees fitted with other packages (e.g., `partykit`) may be added in the future.
`X`	A numeric omics data matrix with dimensions (sample size × number of omics variables). Must be a `matrix`.
`Z`	A `data.frame` of clinical covariates used in tree fitting. Must be the same data used to construct `Tree`.

`LinVars` Logical. Whether to include continuous clinical variables linearly in the model (in addition to tree clustering). Recommended, as trees may not capture linear effects well. Defaults to `TRUE`.

Details

This function allows users to inspect the exact data structure used in fusedTree model fitting. The `PenOpt()` and `fusedTreeFit()` functions call this function internally so no need to call this function to set-up the right data format. It is just meant for users to check what is going on.

Value

A list with the following components:

Clinical A matrix encoding the clinical structure:

- Dummy variables representing membership to leaf nodes of the tree,
- Continuous clinical covariates (if `LinVars = TRUE`).

Each row corresponds to a sample.

Omics A matrix of omics data per leaf node. This matrix has dimensions: sample size × (number of leaf nodes × number of omics variables). For each observation, only the block of omics variables corresponding to its tree node is populated (other blocks are set to zero).#'

Examples

```
p = 5 # number of omics variables (low for illustration)
p_Clin = 5 # number of clinical variables
N = 100 # sample size
# simulate from Friedman-like function
g <- function(z) {
  15 * sin(pi * z[,1] * z[,2]) + 10 * (z[,3] - 0.5)^2 + 2 * exp(z[,4]) + 2 * z[,5]
}
Z <- as.data.frame(matrix(runif(N * p_Clin), nrow = N))
X <- matrix(rnorm(N * p), nrow = N)            # omics data
betas <- c(1,-1,3,4,2)                         # omics effects
Y <- g(Z) + X %*% betas + rnorm(N)             # continuous outcome
Y <- as.vector(Y)
dat = cbind.data.frame(Y, Z) #set-up data correctly for rpart
library(rpart)
rp <- rpart::rpart(Y ~ ., data = dat,
                   control = rpart::rpart.control(xval = 5, minbucket = 10),
                   model = TRUE)
cp = rp$cptable[,1][which.min(rp$cptable[,4])] # best model according to pruning
Treefit <- rpart::prune(rp, cp = cp)
plot(Treefit)
Dat_fusedTree <- Dat_Tree(Tree = Treefit, X = X, Z = Z, LinVars = FALSE)
Omics <- Dat_fusedTree$Omics
Clinical <- Dat_fusedTree$Clinical
```

fusedTree — *Fit a fusedTree model with or without fusion penalty*

Description

Fits a fusedTree model by solving a penalized regression problem using either a linear, logistic, or Cox model. The model includes both a standard ridge (L2) penalty and an optional fusion penalty to encourage similarity between leaf node-specific omics effects.

Usage

```
fusedTree(
  Tree,
  X,
  Y,
  Z,
  LinVars = TRUE,
  model,
  lambda,
  alpha,
  maxIter = 50,
  minSuccDiff = 10^(-10),
  dat = FALSE,
  verbose = TRUE
)
```

Arguments

`Tree`	The fitted tree object. Should be created using `rpart`. Support for other tree-fitting packages (e.g., `partykit`) may be added in the future.
`X`	A matrix of omics data with dimensions (sample size × number of omics variables).
`Y`	The response variable. Can be: • numeric (for linear regression), • binary (0/1, for logistic regression), • a survival object created by `Surv()` (right-censored data only).
`Z`	A data frame of clinical covariates used to fit the tree. Must be a `data.frame`, not a matrix.
`LinVars`	Logical. Whether to include continuous clinical variables linearly in the model (in addition to the tree structure). Defaults to `TRUE`.
`model`	Character. Specifies the type of outcome model to fit. One of: `"linear"`, `"logistic"`, or `"cox"`.
`lambda`	Numeric. Value for the standard ridge (L2) penalty.
`alpha`	Numeric. Value for the fusion penalty.

`maxIter` Integer. Maximum number of iterations for the IRLS (iterative reweighted least squares) algorithm. Used only when `model = "logistic"` or `"cox"`. Defaults to 50.

`minSuccDiff` Numeric. The minimum difference in log-likelihood between successive iterations of IRLS to declare convergence. Only used when `model = "logistic"` or `"cox"`.

`dat` Logical. Whether to return the data used in model fitting (i.e., omics, clinical, and response). Defaults to `FALSE`.

`verbose` Logical. Whether to print progress updates from the IRLS algorithm. Only applies to `model = "logistic"` or `"cox"`.

Details

Linear model: Estimated using a closed-form analytic solution.

Logistic and Cox models: Estimated using IRLS (iterative reweighted least squares), equivalent to the Newton-Raphson algorithm.

Cox model: The full likelihood approach is used, following van Houwelingen et al. (2005). See also van de Wiel et al. (2021) for additional details on penalized regression for survival outcomes.

Value

A list with the following components:

Tree The fitted tree object from 'rpart'.

Effects A named numeric vector of estimated effect sizes, including: intercepts (tree leaf nodes), omics effects (per node), and linear clinical effects (if `LinVars = TRUE`).

Breslow (Optional) The breslow estimates of the baseline hazard `ht` and the cumulative baseline hazard `Ht` for each time point. Only returned for `model = "cox"`.

Parameters A list of model parameters used in fitting (e.g., `lambda`, `alpha`, `model`, etc.).

Clinical (Optional) The clinical design matrix used in fitting, if `dat = TRUE`.

Omics (Optional) The omics design matrix used in fitting, if `dat = TRUE`.

Response (Optional) The response vector used in fitting, if `dat = TRUE`.

The returned list object is of class S3 for which predict() is available

References

porridge

van Houwelingen, H. C., et al.. (2005). Cross-validated Cox regression on microarray gene expression data. *Stad Med*

van de Wiel, M. A., et al. (2021). Fast Cross-validation for Multi-penalty High-dimensional Ridge Regression. *J Comput Graph Stat*

Examples

```
p = 5 # number of omics variables (low for illustration)
p_Clin = 5 # number of clinical variables
N = 100 # sample size
# simulate from Friedman-like function
g <- function(z) {
  15 * sin(pi * z[,1] * z[,2]) + 10 * (z[,3] - 0.5)^2 + 2 * exp(z[,4]) + 2 * z[,5]
}
set.seed(11)
Z <- as.data.frame(matrix(runif(N * p_Clin), nrow = N))
X <- matrix(rnorm(N * p), nrow = N)            # omics data
betas <- c(1,-1,3,4,2)                         # omics effects
Y <- g(Z) + X %*% betas + rnorm(N)             # continuous outcome
Y <- as.vector(Y)
dat = cbind.data.frame(Y, Z) #set-up data correctly for rpart
rp <- rpart::rpart(Y ~ ., data = dat,
                   control = rpart::rpart.control(xval = 5, minbucket = 10),
                   model = TRUE)
cp = rp$cptable[,1][which.min(rp$cptable[,4])] # best model according to pruning
Treefit <- rpart::prune(rp, cp = cp)
plot(Treefit)
folds <- CVfoldsTree(Y = Y, Tree = Treefit, Z = Z, model = "linear")
optPenalties <- PenOpt(Tree = Treefit, X = X, Y = Y, Z = Z,
                       model = "linear", lambdaInit = 10, alphaInit = 10,
                       loss = "loglik",
                       LinVars = FALSE,
                       folds = folds, multistart = FALSE)
optPenalties

# with fusion
fit <- fusedTree(Tree = Treefit, X = X, Y = Y, Z = Z,
                    LinVars = FALSE, model = "linear",
                    lambda = optPenalties[1],
                    alpha = optPenalties[2])
# without fusion
fit1 <- fusedTree(Tree = Treefit, X = X, Y = Y, Z = Z,
                     LinVars = FALSE, model = "linear",
                     lambda = optPenalties[1],
                     alpha = 0)
#compare effect estimates
fit$Effects
fit1$Effects
```

PenOpt *Tuning of the penalty parameters of fusedTree using cross-validation*

Description

Tuning is conducted by optimizing the cross-validated likelihood. Users can either include the fusion penalty (by specifying `alphaInit` > `0`), or omit the fusion penalty (by specifying `alphaInit`

= 0). If alphaInit = 0, only the standard ridge penalty lambda is tuned. Note that Dat_Tree() is called internally so please provide the original data as input arguments.

Usage

```
PenOpt(
  Tree,
  X,
  Y,
  Z,
  model = NULL,
  lambdaInit = 10,
  alphaInit = 10,
  folds = CVfoldsTree(Y = Y, Tree = Tree, Z = Z, model = model),
  loss = "loglik",
  multistart = FALSE,
  maxIter = 30,
  LinVars = FALSE
)
```

Arguments

Tree	The fitted tree. Currently this should be a tree fitted using rpart. Trees fitted by other R packages (e.g., partykit) may be allowed in the future.
X	The original omics data matrix. Has dimensions (sample size × number of omics variables). Should be a matrix.
Y	The response; should be either numeric, binary (encoded by 0 and 1), or a survival object created by Surv() from the survival package. Only right-censored survival data is allowed.
Z	The original clinical data matrix, which was used to fit the tree. Should be a data.frame.
model	Character. Specifies the outcome model. One of "linear", "logistic", or "cox".
lambdaInit	Numeric. Initial value for the standard ridge (L2) penalty lambda. Must be greater than zero. Defaults to 10.
alphaInit	Numeric. Initial value for the fusion penalty alpha. If set to 0, fusion is omitted and only lambda is tuned. Must be zero or greater. Defaults to 10.
folds	List. Each element contains the indices of the test samples for a fold. It is advisable to balance the samples with respect to the outcome (for binary and survival models) and the tree structure. If not provided, folds are generated internally.
loss	Character. The loss function to optimize in cross-validation. For binary and survival outcomes, only "loglik" (cross-validated likelihood) is supported. For continuous outcomes, an alternative is "sos" (sum of squares loss). Defaults to "loglik".

`multistart` Logical. Whether to initialize with different starting values when optimizing the cross-validated likelihood. Can help with stability when both `lambda` and `alpha` are tuned, at the cost of longer runtime. Defaults to `FALSE`.

`maxIter` Integer. Maximum number of iterations for the IRLS (iterative reweighted least squares) algorithm. Used only for logistic and Cox models. Defaults to 30.

`LinVars` Logical. Whether to include continuous clinical variables linearly in the model (in addition to the tree structure). Can be helpful since trees may not capture linear effects well. Defaults to `TRUE`.

Details

The cross-validated likelihood is optimized using the `Nelder-Mead` method from `stats::optim()`. When tuning both `lambda` and `alpha`, the objective function can be noisy. Setting `multistart` = `TRUE` performs optimization from several starting values to improve robustness. This is only applicable when `alphaInit` > `0`.

Value

A numeric vector with the tuned values of the penalties:

- `lambda`: standard ridge (L2) penalty.
- `alpha`: fusion penalty (only if `alphaInit` > `0`).

If `alphaInit` = `0`, only the tuned `lambda` is returned.

References

porridge

Examples

```
p = 5 # number of omics variables (low for illustration)
p_Clin = 5 # number of clinical variables
N = 100 # sample size
# simulate from Friedman-like function
g <- function(z) {
  15 * sin(pi * z[,1] * z[,2]) + 10 * (z[,3] - 0.5)^2 + 2 * exp(z[,4]) + 2 * z[,5]
}
set.seed(11)
Z <- as.data.frame(matrix(runif(N * p_Clin), nrow = N))
X <- matrix(rnorm(N * p), nrow = N)            # omics data
betas <- c(1,-1,3,4,2)                         # omics effects
Y <- g(Z) + X %*% betas + rnorm(N)             # continuous outcome
Y <- as.vector(Y)
dat = cbind.data.frame(Y, Z) #set-up data correctly for rpart
rp <- rpart::rpart(Y ~ ., data = dat,
                   control = rpart::rpart.control(xval = 5, minbucket = 10),
                   model = TRUE)
cp = rp$cptable[,1][which.min(rp$cptable[,4])] # best model according to pruning
Treefit <- rpart::prune(rp, cp = cp)
plot(Treefit)
```

```
folds <- CVfoldsTree(Y = Y, Tree = Treefit, Z = Z, model = "linear")
optPenalties <- PenOpt(Tree = Treefit, X = X, Y = Y, Z = Z,
                       model = "linear", lambdaInit = 10, alphaInit = 10,
                       loss = "loglik",
                       LinVars = FALSE,
                       folds = folds, multistart = FALSE)
optPenalties
```

`predict.fusedTree` *Predict Method for Fused Tree Models*

Description

Generates predictions from a fitted `fusedTree` object using new clinical and omics data.

Usage

```
## S3 method for class 'fusedTree'
predict(object, newX, newZ, newY, ...)
```

Arguments

`object` An object of class `"fusedTree"` returned by the `fusedTree` function.

`newX` A matrix of new omics covariates for prediction. Must have the same number of columns (variables) as used in the model fitting.

`newZ` A data frame of new clinical covariates for prediction.

`newY` A numeric response vector or a `Surv` object. For linear regression, and logistic regression, newY is only used to store predictions and response values together. For cox regression, newY is used to interpolate the baseline hazard to the event times of the test data.

`...` Currently not used. Included for S3 method consistency.

Value

A model-specific prediction object:

- For `"linear"` models: a `data.frame` with columns `Resp` (observed responses) and `Ypred` (predicted values).
- For `"logistic"` models: a `data.frame` with columns `Resp` (observed labels), `Ypred` (predicted probabilities), and `LinPred` (linear predictor).
- For `"cox"` models: a `list` with two elements:

  data A `data.frame` with columns `Resp` (a `Surv` object) and `LinPred`.

  Survival A matrix of predicted survival probabilities. Rows = test subjects, columns = unique times from `newY`.

See Also

fusedTree for model fitting.

Examples

```
p = 5 # number of omics variables (low for illustration)
p_Clin = 5 # number of clinical variables
N = 100 # sample size
# simulate from Friedman-like function
g <- function(z) {
  15 * sin(pi * z[,1] * z[,2]) + 10 * (z[,3] - 0.5)^2 + 2 * exp(z[,4]) + 2 * z[,5]
}
set.seed(11)
Z <- as.data.frame(matrix(runif(N * p_Clin), nrow = N))
X <- matrix(rnorm(N * p), nrow = N)            # omics data
betas <- c(1,-1,3,4,2)                         # omics effects
Y <- g(Z) + X %*% betas + rnorm(N)             # continuous outcome
Y <- as.vector(Y)
dat = cbind.data.frame(Y, Z) #set-up data correctly for rpart
rp <- rpart::rpart(Y ~ ., data = dat,
                   control = rpart::rpart.control(xval = 5, minbucket = 10),
                   model = TRUE)
cp = rp$cptable[,1][which.min(rp$cptable[,4])] # best model according to pruning
Treefit <- rpart::prune(rp, cp = cp)
fit <- fusedTree(Tree = Treefit, X = X, Y = Y, Z = Z,
                    LinVars = FALSE, model = ”linear”,
                    lambda = 10,
                    alpha = 1000)
Preds <- predict(fit, newX = X, newZ = Z, newY = Y)
```

Index

CVfoldsTree, 2

Dat_Tree, 3

fusedTree, 5, *10, 11*

PenOpt, 7
predict.fusedTree, 10

Surv, *10*