Package 'CNVreg'

July 21, 2025

Type Package

Title CNV-Profile Regression for Copy Number Variants Association Analysis with Penalized Regression

Version 1.0

Description Performs copy number variants association analysis with Lasso and Weighted Fusion penalized regression.
Creates a ``CNV profile curve'' to represent an individual's CNV events across a genomic region so to capture variations
in CNV length and dosage. When evaluating association, the CNV profile curve is directly used as a predictor in the
regression model, avoiding the need to predefine CNV loci. CNV profile regression estimates CNV effects at each genome
position, making the results comparable across different studies. The penalization encourages sparsity
in variable selection with a Lasso penalty and encourages effect smoothness between consecutive CNV events with a weighted
fusion penalty, where the weight controls the level of smoothing between adjacent CNVs.
For more details, see Si (2024) <doi:10.1101/2024.11.23.624994>.

License GPL-3

Depends R (>= 4.1.0)

VignetteBuilder knitr

Imports stats, Matrix, doParallel, dplyr, foreach, glmnet, tidyr

Encoding UTF-8

NeedsCompilation no

LazyData true

Suggests rmarkdown, knitr, rlang, tidyverse, markdown, kableExtra, patchwork, ggplot2, withr

Author Yaqin Si [aut],
Shannon T. Holloway [ctb, cre],
Jung-Ying Tzeng [ctb]

Maintainer Shannon T. Holloway <shannon.t.holloway@gmail.com>

RoxygenNote 7.3.2

Collate 'utils.R' 'probPred.R' 'linearPred.R' 'logLH.R' 'loss.R'
'helpful_tests.R' 'ctnsSolution.R' 'rwlsSolution.R'
'Wtsmth_Fit.R' 'nfoldSplit.R' 'WTsmth_nFold_CV.R' 'breakCNV.R'
'datadescrip.R' 'wideDataRaw.R' 'wideFrequency.R'
'weightMatrix.R' 'prep.R'

Repository CRAN

Date/Publication 2025-03-10 16:50:21 UTC

Contents

CNVCOVY 2
cvfit_WTSMTH 3
fit_WTSMTH 5
prep 7

Index **9**

`CNVCOVY` *Simulated data with copy number variants (CNV), Covariates (Cov), and outcomes traits (Y_QT for a continuous outcome and Y_BT for a binary outcome)for the illustration of CNV association analysis with penalized regression in CNVreg.*

Description

Simulated data with copy number variants (CNV), Covariates (Cov), and outcomes traits (Y_QT for a continuous outcome and Y_BT for a binary outcome)for the illustration of CNV association analysis with penalized regression in CNVreg.

Usage

```
data("CNVCOVY")
```

Format

CNVCOVY.RData provides 4 datasets: CNV, Cov, Y_QT, and Y_BT.

- CNV. A data frame of 2680 CNV records, it has 5 variables:
  - ID. The sample ID of the CNV records in each row. There are 797 unique IDs.
  - CHR. An integer variable, the chromosome number of CNV records.
  - BP1. A numeric variable, the starting breakpoint of the CNV records.
  - BP2. A numeric variable, the ending breakpoint of the CNV records.
  - TYPE. An integer variable, how many copies of the CNV present.
- Cov. A data frame contains covariats of 900 samples (including 797 samples in the CNV data set). Cov has 3 variables.

  - ID. The sample ID of 900 individuals (900 unique IDs).
  - Sex. An integer covariate, sex of each sample: 0 male, 1 female.
  - Age. A numeric covariate, age of each sample.
- Y_QT and Y_BT are two data frames for outcomes traits. Y_QT contains a continuous trait. Y_BT contains a binary trait. Both have 2 variables
  - ID. The sample ID of 900 individuals (900 unique IDs).
  - Y. Y_QT has a numeric variable range (-4.89 – 16.70) Y_BT has an integer variable with controls 0 and cases 1.

`cvfit_WTSMTH` *Penalized Regression with Lasso and Weighted Fusion Penalties with Cross-Validation*

Description

Uses n-fold cross-validation (CV) to fit a penalized regression model with Lasso penalty and weighted fusion penalty. Return the loss of all pair of tuning parameters, find the best pair of tuning parameters with the lowest loss, and estimate the regression coefficient. The CV process fine-tunes the tuning parameters required for the penalty terms and find the pair of lambda_1 and lambda_2 that minimizes the average validation loss.

Usage

```
cvfit_WTSMTH(
  data,
  lambda1 = seq(-8, 0, 1),
  lambda2 = seq(-8, 8, 1),
  weight = NULL,
  family = c("gaussian", "binomial"),
  cv.control = list(n.fold = 5L, n.core = 1L, stratified = FALSE),
  iter.control = list(max.iter = 8L, tol.beta = 10^(-3), tol.loss = 10^(-6)),
  verbose = TRUE
)
```

Arguments

`data`	An object of class "WTsmth.data" as generated by prep()
`lambda1`	A numeric vector. Lambda_1 values to be considered that controls the Lasso penalty. Provided values will be transformed to 2^(lambda1). The default value is c(-8:0). The user can customize the range and step_size of the candidate tuning parameters In most cases, the user will need to run the function more than one time to adjust the range and step_size of tuning parameters to locate to a reasonable range according to the 'Loss' and 'selected.lambda' from the previous round of model fitting

`lambda2` A numeric vector. Lambda_2 values to be considered that controls the weighted fusion penalty. Provided values will be transformed to 2^(lambda2). The default value is c(-8:8). The user can customize the range and step_size of the candidate tuning parameters In most cases, the user will need to run the function more than one time to adjust the range and step_size of tuning parameters to locate to a reasonable range according to the 'Loss' and 'selected.lambda' from the previous round of model fitting.

`weight` A character. The type of weighting. Must be one of ('eql', 'keql', 'wcs', 'kwcs', 'wif', 'kwif'), which indicates the equal weight, K x equal weight, Cosine similarity, K x cosine similarity, inverse frequency, and K x inverse frequency respectively, where K is the number of individuals in each CNV active region. 'eql' and 'keql' gives equal weight to adjacent CNVs. 'wcs' and 'kwcs' allow similar CNV fragments to have more similar effect size. 'wif' and 'kwif' will encourage CNV with lower frequency to borrow information from nearby more frequent CNV fragments. Considering that CNVs usually present in some CNV-active regions and there are large regions in between with no CNV at all. K will describe the number of individuals having any CNV activities in a CNV-active region, and varying the weight according to the sample size across regions.

`family` A character. The family of the outcome. Must be one of "gaussian" (Y is continuous) or "binomial" (Y is binary).

`cv.control` A list object. Allows user to control cross-validation procedure. Allowed elements are 'n.fold', the number of cross-validation folds with a default value of 5, depends on the sample size, it can be chosen to have other folds (such as 3, 10); 'n.core' is the number of cores to use in procedure, check available computation resource before choosing; and 'stratified', if TRUE and 'family' = "binomial", the folds will be stratified within each category of Y (this option is recommended if either category of the outcome is "rare".)

`iter.control` A list object. Allows the user to control iteratively update procedure. Allowed elements are 'max.iter', the maximum number of iterations, it guarantees the function returns results within reasonable time; 'tol.beta' is the threshold below which the procedure is deemed converged, which controls the absolute difference between consecutive beta updates. 'tol.loss' is the threshold below which the procedure is deemed converged, which controls the difference in consecutive loss updates.

`verbose` A logical object. If 'TRUE', print progression updates.

Value

A list containing 1. 'Loss': The average loss of the validation set for all pairs of candidate tuning parameters, the smaller the loss, the better performance of the corresponding pair of parameters. 2. 'selected.lambda' :The selected tuning parameter values that minimized the loss. 3. 'coef' the model coefficient estimate (coef) at the selected tuning parameters.

Examples

```
# Note we use here a very small example data set and few candidate lambda1
# and lambda2 to expedite examples.
```

```
# load toy dataset
data("CNVCOVY")

# prepare data format for regression analysis

## Continuous outcome Y_QT
frag_data <- prep(CNV = CNV, Y = Y_QT, Z = Cov, rare.out = 0.05)
QT_tune <- cvfit_WTSMTH(frag_data,
                        lambda1 = seq(-4.75, -5.25, -0.25),
                        lambda2 = seq(18, 22, 1),
                        weight = "eql",
                        family = "gaussian")

## Binary outcome Y_BT

# We can directly replace frag_data$Y with Y_BT in the correct format,
# ensuring that the ordering matches that of the prepared object.

rownames(Y_BT) <- Y_BT$ID
frag_data$Y <- Y_BT[names(frag_data$Y), "Y"] |> drop()
names(frag_data$Y) <- rownames(frag_data$Z)

# Or, we can also repeat the prep() call
# frag_data <- prep(CNV = CNV, Y = Y_BT, Z = Cov, rare.out = 0.05)

BT_tune <- cvfit_WTSMTH(frag_data,
                        lambda1 = c(-5.25, -5, -4.75),
                        lambda2 = c(5,  6, 7),
                        weight = "eql",
                        family = "binomial")
```

`fit_WTSMTH` *Penalized Regression with Lasso and Weighted Fusion Penalties with Given Parameters*

Description

Performs penalized regression with Lasso penalty and weighted fusion penalty for a given pair of tuning parameters (lambda1 and lambda2), which is determined by the user based on prior knowledge or use any number just for testing purpose.

Usage

```
fit_WTSMTH(
  data,
  lambda1,
  lambda2,
  weight = NULL,
  family = c("gaussian", "binomial"),
```

```
  iter.control = list(max.iter = 8L, tol.beta = 10^(-3), tol.loss = 10^(-6)),
  ...
)
```

Arguments

data	An object of class "WTsmth.data" as generated by prep()
lambda1	A scalar numeric. Lambda_1 value to be considered. Provided value will be transformed to 2^(lambda1).
lambda2	A scalar numeric Lambda_2 value to be considered. Provided value will be transformed to 2^(lambda2).
weight	A character. The type of weighting. Must be one of eql, keql, wcs, kwcs, wif, kwif indicating equal weight, K x equal weight, Cosine similarity, K x cosine similarity, inverse frequency, and K x inverse frequency, where K is the number of individuals in each CNV-active region. ‘eql‘ and ‘keql‘ gives equal weight to adjacent CNVs. ‘wcs‘ and ‘kwcs‘ allow similar CNV fragments to have more similar effect size. ‘wif‘ and ‘kwif‘ will encourage CNV with lower frequency to borrow information from nearby more frequent CNV fragments. Considering that CNVs usually present in some CNV-active regions and there are large regions in between with no CNV at all. K will describe the number of individuals having any CNV activities in a CNV-active region, and varying the weight according to the sample size across regions.
family	A character. The family of the outcome. Must be one of "gaussian" (Y is continuous) or "binomial" (Y is binary).
iter.control	A list object. Allows user to control iterative update procedure. Allowed elements are "max.iter", the maximum number of iterations; "tol.beta", the difference between consecutive beta updates below which the procedure is deemed converged; and "tol.loss", the difference in consecutive loss updates below which the procedure is deemed converged.
...	Ignored.

Value

A numeric vector. The estimated model parameters

Examples

```
# Note we use here a very small example data set to expedite examples.

# load toy dataset
data("CNVCOVY")

# prepare data format for regression analysis

## Continuous outcome Y_QT
frag_data <- prep(CNV = CNV, Y = Y_QT, Z = Cov, rare.out = 0.05)
QT_fit <- fit_WTSMTH(frag_data,
                     lambda1 = -5,
```

```
                      lambda2 = 21,
                      weight = "eql",
                      family = "gaussian")

## Binary outcome Y_BT

# We can directly replace frag_data$Y with Y_BT in the correct format,
# ensuring that the ordering matches that of the prepared object.

rownames(Y_BT) <- Y_BT$ID
frag_data$Y <- Y_BT[names(frag_data$Y), "Y"] |> drop()
names(frag_data$Y) <- rownames(frag_data$Z)

# Or, we can also repeat the prep() call
# frag_data <- prep(CNV = CNV, Y = Y_BT, Z = Cov, rare.out = 0.05)

BT_fit <- fit_WTSMTH(frag_data,
                        lambda1 = -5,
                        lambda2 = 6,
                        weight = "eql",
                        family = "binomial")
```

`prep` *Prepare Data for Analysis*

Description

Required preprocessing of analysis data. Function converts an individual's CNV events within a genomic region (from one chromosome) to a CNV profile curve, further processes it as CNV fragments, and filter out rare fragments. In addition, the adjacency relationship between CNV fragments is analyzed and weight matrices are generated. The resulting 'WTsmth.data' object, is provided as input to the regression analysis.

Usage

```
prep(CNV, Y, Z = NULL, rare.out = 0.05)
```

Arguments

`CNV` A data.frame in PLINK format. Specifically, must contain columns:

- "ID": character, unique identity for each sample
- "CHR": integer, allowed range 1-22 NOTE: only 1 CHR can be present, which means this function processes one chromosome at a time.
- "BP1": integer, CNV event starting position,
- "BP2": integer, CNV event ending position, each record must have BP1 <= BP2, CNV at least 1bp (or other unit length)
- "TYPE": integer, range 0, 1, 3, 4, and larger allowed, i.e., 2 is not allowed.

Y	A data.frame. Must include column "ID". Must have 2 columns. For binary, values must be 0 (control) or 1 (case). For continuous, values must be real number. Y$ID must contain all unique CNV$ID. Y and Z have the same IDs.
Z	A data.frame. Must include column "ID". All other columns are covariates, which can be continuous, binary, or categorical variables. At a minimum, Z must contain all unique CNV$ID values.
rare.out	A scalar numeric in the range [0, 0.5); event rates below this value are filtered out of the data.

Value

An S3 object of class "WTsmth.data" extending a list object containing

- design CNV data converted to design matrix.
- Z The processed covariate matrix.
- Y The processed response vector.
- weight.structure A Matrix object. The structure of the weight matrix.
- weight.options A matrix object. Each row is the multiplicative vector to obtain each available weight. Specifically, the A matrix is obtained as weight_option[i,] * weight.structure where i = 1-6 with 1="eql", 2="keql", 3="wcs", 4="kwcs", 5="wif", and 6="kwif".
- CNVR.info A data.frame containing details about the fragment structure.

Examples

```
# Note we use here a very small example data set to expedite examples.

# load toy dataset
data("CNVCOVY")

## Continuous outcome Y_QT
frag_data <- prep(CNV = CNV, Y = Y_QT, Z = Cov, rare.out = 0.05)
```

Index

∗ **datasets**
CNVCOVY, 2

CNV *(*CNVCOVY*)*, 2
CNVCOVY, 2
Cov *(*CNVCOVY*)*, 2
cvfit_WTSMTH, 3

fit_WTSMTH, 5

prep, 7

Y_BT *(*CNVCOVY*)*, 2
Y_QT *(*CNVCOVY*)*, 2