Package 'fspls2'

September 12, 2026

Type Package

Title A Package to Find Minimal Transcriptional Signatures

Version 0.1.0

Date 2026-09-02

Author Lachlan Coin [aut, cre]

Imports R6, jsonlite, Matrix, glmnet, tidyr, MASS, ggplot2, pROC, binom, confintr, DBI, RSQLite, methods,tibble, ggrepel, RColorBrewer, data.table

Depends R (>= 4.1.0)

Maintainer Lachlan Coin <l.coin@imb.uq.edu.au>

Description Identifies minimal biomarker signatures for predicting phenotypes from multiomic data. Integrates multiple omics layers, supports internal cross-validation. Supports missing values in predictors and outcomes. Enables model based imputation of uncertain values. Supports continuous, binary, multi-class and ordinal outcomes.

License GPL-2

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2026-09-12 12:30:02 UTC

Config/roxygen2/version 8.0.0

Encoding UTF-8

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

Contents

analysisBase 2
analysisEnv 3
dataH 5

fspls.iterative 11
fspls.select 11
getFamily 12
getFolds 13
getTransform 13
plotEval 14

Index **16**

`analysisBase` *Analysis Base Class*

Description

A class that encapsulates a dataset holder and/or analysis of datasets

Details

Available methods:

- `new()` - Create a new instance
- `name()` - Get dataset name
- `clone()` - Clone the object

Methods

Public methods:

- `analysisBase$new()`
- `analysisBase$name()`
- `analysisBase$integrate()`
- `analysisBase$clone()`

`analysisBase$new()`: Create a new instance of base class

Usage:

```
analysisBase$new(nme, dims = NULL, flags = list(), dbDir = tempdir())
```

Arguments:

`nme` name

`dims` dims

`flags` a list object specifying options

`dbDir` location for database to be written, default to tempdir

`analysisBase$name()`: get dataset name;

Usage:

```
analysisBase$name()
```

Returns: assigned name of dataset

`analysisBase$integrate()`: integrates variables from different CV runs

Usage:

```
analysisBase$integrate(variables)
```

Arguments:

`variables` a variables object

`analysisBase$clone()`: The objects of this class are cloneable with this method.

Usage:

```
analysisBase$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`analysisEnv` *Analysis Class*

Description

A class that encapsulates analysis of one or more datasets. Only expert users wlil need access to this

Details

Available methods:

- `new()` - Create a new instance
- `name()` - Get dataset name
- `clone()` - Clone the object

@export

Super class

`analysisBase` -> `analysisEnv`

Methods

Public methods:

- `analysisEnv$new()`
- `analysisEnv$plot_trajectory()`
- `analysisEnv$select_k()`
- `analysisEnv$getTodo()`
- `analysisEnv$clear_db()`
- `analysisEnv$savePvalsAndNextVars()`
- `analysisEnv$select()`

- analysisEnv$clone()

`analysisEnv$new()`: Create a new instance

Usage:

```
analysisEnv$new(flags = list(), dbDir = tempdir())
```

Arguments:

`flags` a list object specifying options

`dbDir` location for database to be written. If NULL then no DB gets the trajectory

`analysisEnv$plot_trajectory()`:

Usage:

```
analysisEnv$plot_trajectory(datasH)
```

Arguments:

`datasH` a list of dataH objects after variable selection selection for a single fold

`analysisEnv$select_k()`:

Usage:

```
analysisEnv$select_k(datasH, k1, vars_l_todo)
```

Arguments:

`datasH` a list of dataH objects

`k1` the fold

`vars_l_todo` an object representing what is left to do

Returns: vars_l_todo object Get todo

`analysisEnv$getTodo()`:

Usage:

```
analysisEnv$getTodo(flags, phens)
```

Arguments:

`flags` list of options

`phens` phenotypes

Returns: object outlining what is left to do for next iteration Clear the databases. Only necessary if using a database and re-running analyses

`analysisEnv$clear_db()`:

Usage:

```
analysisEnv$clear_db(drop = FALSE, exclude = "vars", datasH = NULL)
```

Arguments:

`drop` completely drop tables and start from scratch

`exclude` tables to exclue from clearing

`datasH` list of dataH objects to clear Get the next vars in iteration. Internal function

`analysisEnv$savePvalsAndNextVars()`:

Usage:

```
analysisEnv$savePvalsAndNextVars(vars_l_todo, comb2_new, data_nme, k1)
```

Arguments:

`vars_l_todo` vars_l_todo

`comb2_new` results

`data_nme` name of dataset

`k1` which repetition

Returns: object outlining what is left to do main function for variable selection

`analysisEnv$select()`:

Usage:

```
analysisEnv$select(
  datasH,
  phens,
  transform_x,
  data_types = datasH[[1]]$data_types()
)
```

Arguments:

`datasH` a list of dataH objects

`phens` list of phenotypes

`transform_x` transformation object

`data_types` data_types object

`analysisEnv$clone()`: The objects of this class are cloneable with this method.

Usage:

```
analysisEnv$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`dataH` *Data holder class*

Description

A class that encapsulates a dataset holder

Details

Available methods:

- `new()` - Create a new instance
- `name()` - Get dataset name
- `clone()` - Clone the object

Super class

analysisBase -> dataH

Methods

Public methods:

- dataH$new()
- dataH$multiAnglesAndPv()
- dataH$clear_db()
- dataH$get_trajectory()
- dataH$data_types()
- dataH$split()
- dataH$pheno()
- dataH$dims()
- dataH$nreps()
- dataH$update()
- dataH$select()
- dataH$y()
- dataH$plotPredictions()
- dataH$extractPredictions()
- dataH$plotData()
- dataH$updateWeights()
- dataH$getYNew()
- dataH$getProjectedData()
- dataH$getVariance()
- dataH$makeAllModels()
- dataH$evaluateAllModels()
- dataH$clone()

dataH$new(): Create a new instance

Usage:

```
dataH$new(
  data,
  y,
  certainty = rep(1, nrow(y)),
  nme,
  flags,
  family = getFamily(y),
  dbDir = NULL
)
```

Arguments:

`data` a list of matrices (of different modalities), with columns as variables and rows as samples

`y` phenotype matrix, with columns as outcomes and rows as samples

`certainty` the certainty of each observation, expressed as a probability of being correct. Default is to treat each observation as certain

`nme` The name of the data objetct.

`flags` list of options, described in the vignette

`family` the statistical family of phenotype y, can be calculated by getFamily

`dbDir` dir for database to store results to speed up re-reruns. Can be NULL

`transform_x` a transformation object from the function getTransformation

`dataH$multiAnglesAndPv()`: Calculate the angles and pv across multiple phenotypes. This is an internal function and should not need to be called by user

Usage:

```
dataH$multiAnglesAndPv(comb20, k1, expt_id, vars_l_todo)
```

Arguments:

`comb20` values from previous iteration

`k1` k1 is the current fold

`expt_id` and ID for the experiment (can be 0)

`vars_l_todo` the variables under consideration clear database. This is only used if you want to clear the database. Internal function

`phens1` phenotypes being used

`flags` list of options

`dataH$clear_db()`:

Usage:

```
dataH$clear_db(drop = FALSE, exclude = "vars")
```

Arguments:

`drop` drop the tables instead of clear

`exclude` which tables not to clear

`dataH$get_trajectory()`: returns data on trajectory

Usage:

```
dataH$get_trajectory()
```

Returns: a dataframe with information to plot trajectory get the data types

`dataH$data_types()`:

Usage:

```
dataH$data_types(
  data_nmes = list(all = names(private$data$data)),
 inds = lapply(data_nmes, function(x) list(pos_inds = c(), neg_inds = c(), max = 1000)),
  direction = lapply(inds, private$data$getDirection),
  incl = c(),
  excl = c()
)
```

Arguments:

`data_nmes` a list of names of the data names to include, optional

`inds` a list of positive and negative inds, optional

`direction` a direction object

`incl` a list of variables to include

`excl` a list of variables to exclude

Returns: an object which is used to specify both directionallity and data types

`dataH$split()`: split dataset into smaller datasets

Usage:

```
dataH$split(proportions = c(0.5, 0.5))
```

Arguments:

`proportions` what proportions to split into

Returns: a list of dataH objects with data partiioned according to proportions

`dataH$pheno()`: get the phenotyeps

Usage:

```
dataH$pheno()
```

Returns: phenotypes

`dataH$dims()`: get the dimensions

Usage:

```
dataH$dims()
```

Returns: dimensions get the nreps

`dataH$nreps()`:

Usage:

```
dataH$nreps()
```

Returns: nreps

`dataH$update()`: update the phenotypes without remaking the entire object. You can provide many phenotypes in the initialisation stage, but only consider a subset in model fitting stage in this way.Must run this before select

Usage:

```
dataH$update(
  phens = self$pheno()$all,
  flags = private$flags,
  transform_x = fromJSON(flags$transform_x),
  data_types = self$data_types()
)
```

Arguments:

`phens` phenotypes

`flags` list of options

`transform_x` transformation object

`data_types` a data_type object

`dataH$select()`: main function for variable selection

Usage:

```
dataH$select(analysis = analysisEnv$new(flags = private$flags, dbDir = NULL))
```

Arguments:

`analysis` an analysisEnv object

`dataH$y()`: provides access to internal storage of phenotype data

Usage:

```
dataH$y(phens = self$pheno()[[1]])
```

Arguments:

`phens` list of phenotypes

Returns: list of matrices

`dataH$plotPredictions()`: ggplot to visualise predictions

Usage:

```
dataH$plotPredictions(all_modelsh, update = FALSE, liab = TRUE)
```

Arguments:

`all_modelsh` fitted models from makeAllModels

`update` whether to update

`liab` return liability score, or probability (for binomial, ordinal multinomial)

Returns: a ggplot

`dataH$extractPredictions()`: extract the predictions for the fitted models

Usage:

```
dataH$extractPredictions(all_modelsh, liab = TRUE)
```

Arguments:

`all_modelsh` fitted models from makeAllModels

`liab` return liability score, or probability (for binomial, ordinal multinomial)

Returns: a table with results

`dataH$plotData()`: plot all data for selected variables

Usage:

```
dataH$plotData(
  variables,
  all_types = FALSE,
  violin = FALSE,
  assoc = FALSE,
  update = FALSE
)
```

Arguments:

`variables` object returned by fspls.select
`all_types` use all types?
`violin` violin plots
`assoc` use association
`update` whether to use attributes from bariables to update this object, default is FALSE

Returns: a table with results

`dataH$updateWeights()`: update the weights based on predictions from the model

Usage:

```
dataH$updateWeights(all_models)
```

Arguments:

`all_models` the output of dataH function getAllModels

Returns: list of auc beta and updated sumdiff of weights used as stopping criteria

`dataH$getYNew()`: return updated y after iterative imputation

Usage:

```
dataH$getYNew()
```

Returns: updated y, updated certainty and indices which were updated

`dataH$getProjectedData()`: get data after projection

Usage:

```
dataH$getProjectedData(varnames)
```

Arguments:

`varnames` varnames

Returns: projected data after projecting out varnames

`dataH$getVariance()`: get variance of data

Usage:

```
dataH$getVariance(varnames)
```

Arguments:

`varnames` varnames

Returns: variance

`dataH$makeAllModels()`: fit models based on variables

Usage:

```
dataH$makeAllModels(variables, update = FALSE)
```

Arguments:

`variables` list of variables selected by select method
`update` whether to automatically update phens, transform_x and flags , default TRUE

Returns: fitted models

`dataH$evaluateAllModels()`: evaluate the fit of models

Usage:

```
dataH$evaluateAllModels(all_modelsh, update = TRUE)
```

Arguments:

`all_modelsh` models fitted from makeAllModels

`update` whether to automatically update phens, transform_x and flags , default TRUE

Returns: evaluation of models

`dataH$clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
dataH$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`fspls.iterative` *main function for variable selection with unknown values*

Description

main function for variable selection with unknown values

Usage

```
fspls.iterative(dataset, flags, transform_x)
```

Arguments

`dataset`	a list with y and data value
`flags`	list of options
`transform_x`	transformation object

`fspls.select` *main function for variable selection*

Description

main function for variable selection

Usage

```
fspls.select(
  datasH,
  flags,
  transform_x,
  phens = datasH[[1]]$pheno()$all,
  data_types = datasH[[1]]$data_types(),
  dbDir = NULL
)
```

Arguments

datasH	a list of dataH objects
flags	list of options
transform_x	transformation object
phens	list of phenotypes
data_types	a data type object
dbDir	database dir, can be NULL

---

getFamily *Infer the statistical families of the phenotypes*

Description

Infer the statistical families of the phenotypes

Usage

```
getFamily(y_mat, max_ordinal = getOption("max_ordinal", 10))
```

Arguments

y_mat	a matrix of phenotypes, with columns as variables and rows as samples
max_ordinal	maximum number of values before we consider an ordinal value as a continuous value

Value

a list with the statistical families detected

getFolds *extracts the full model variables from a cross validation run*

Description

extracts the full model variables from a cross validation run

Usage

```
getFolds(leng, nrep = NA, batch = NA, randomize = TRUE)
```

Arguments

leng	length of dataset
nrep	number of reps
batch	batch size (one of batch or nrep is non NA)
randomize	whether to randomize

getTransform *Get object for transforming the x variables*

Description

Get object for transforming the x variables

Usage

```
getTransform(
  pows = c(1),
  offset = 0,
  n_random = 1,
  perm = FALSE,
  norm = 1,
  CHECK = FALSE
)
```

Arguments

pows	power to raise to
offset	offset to subtract
n_random	how many random variables
perm	whether random is permutation
norm	rescaling factor
CHECK	check whether inverse function works

Value

transformation object

`plotEval` *Plot the results from evaluateAll*

Description

Plot the results from evaluateAll

Usage

```
plotEval(
  eval3,
  shape_color = c("pheno", "subpheno"),
  shape = shape_color,
  text = "variable",
  dotsize = "nsamps",
  color = shape_color,
  linetype = shape,
  showranges = TRUE,
  txtsize = 1,
  logy = FALSE,
  legend = FALSE,
  sep_by = "",
  scales = "free",
  point = TRUE,
  line = TRUE,
  labelsize = 2,
  grid0 = c("cohort", "measure"),
  grid1 = "cv_full",
  title = "",
  title1 = ""
)
```

Arguments

`eval3`	a tibble or data frame from evaluateAll method from dataH
`shape_color`	a list of column names to encode as both shape and color
`shape`	defaults to shape_color but can be specified separately
`text`	a list of column names to include as text labels
`dotsize`	a numerical column to encode dotsize, or a number for constant sizes
`color`	defaults to shape_color but can be specified separately
`linetype`	column controlling the type of line

showranges	whether to show 95% CI as ribbon
txtsize	size of text
logy	whether to log y axis
legend	whether to show a legend
sep_by	list of column names to separate into diff plots
scales	used in faceting can be free or fixed
point	whether to show dots
line	whether to include line
labelsize	the size of the text labels
grid0	how to facet on y
grid1	how to facet on x
title	the title
title1	columns to include in the title

Value

ggplot2 plot

Index

analysisBase, 2, *3*, *6*
analysisEnv, 3

dataH, 5

fspls.iterative, 11
fspls.select, 11

getFamily, 12
getFolds, 13
getTransform, 13

plotEval, 14