Package 'MammalMethylClock'

July 22, 2024

Type Package

Version 1.0.0

Title Tools for Building and Utilizing Epigenetic Clocks in Mammals

Description Functions for applying published epigenetic clocks, and pipelines for building new epigenetic clocks for mammals. Utility functions for evaluating and visualizing results.

License CC BY-NC-ND 4.0

Encoding UTF-8

ByteCompile true

LazyData true

Depends R (>= 3.5.0),
glmnet,
WGCNA

Imports graphics,
grDevices,
stats,
utils,
dplyr,
tidyr,
tibble,
ggplot2,
RColorBrewer,
snakecase

Suggests knitr,
rmarkdown,
tidyverse,
glmnetUtils,
survival,
tab,
grid,
gridExtra,
gtable

VignetteBuilder knitr

biocViews

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

R topics documented:

`alignDatToInfo` *Align Normalized Methylation Data with Metadata Sheet Using Sample ID Columns*

Description

Align metadata `info` with normalized methylation data `dat`, by matching rows that represent each sample, and the removing sample ID's from `dat`. This is an important step before doing any analyses or building any clocks, because the methylation data needs to be unlabeled for use in any regression models.

Usage

```
alignDatToInfo(info, dat, SAMPVAR.info, SAMPVAR.dat)
```

Arguments

info	Data frame, the metadata for the samples
dat	Data frame, the normalized methylation data for the samples
SAMPVAR.info	String, the name of the column in info containing unique sample ID's
SAMPVAR.dat	String, the name of the column in dat containing matching sample ID's

Details

For the conceptual framework of this package, the file dat should still contain the sample ID's in one of its columns. In addition, info and dat should be formatted such that rows correspond to samples

This function will (if necessary) rearrange and filter the rows of both data sets so that every row in info corresponds to the same row number in dat, and then remove from dat the column containing sample ID's. We use ys and xs to refer to these half-labeled data frames that are returned by this function, and which are ready for use in regression models.

Note that this function removes all columns of dat that do not contain the string 'cg' in the column name, in order to be compatible with genome annotation probe mapping files.

Value

A 2-element list, containing ys and xs:

ys	The aligned version of info, otherwise unmodified. As the name suggests, use this for the 'y' in regression analyses.
xs	The aligned and modified verison of dat, with the column containing sample ID's removed.

---

axis_limits *Calculate Buffered Axis Limits for Plotting*

Description

Finds buffered axis limits for plotting a single data vector along one dimension.

Usage

```
axis_limits(x)
```

Arguments

x	Numeric vector, containing data to be plotted

Details

In order to make plots appealing, take the range (max - min) of a data vector, and add 3% of the range to the max, and subtract 3% from the min.

Value

A numeric 2-element vector, the buffered axis limits for plotting the data.

---

axis_square_limits *Calculate Buffered Shared Axis Limits for Square Plotting*

Description

Finds buffered axis limits for plotting a pair of data vectors, such that it jointly contains the values of both.

Usage

```
axis_square_limits(x, y)
```

Arguments

x, y	Numeric vectors with same length, containing data to be plotted

Details

In order to make plots appealing, take the range (max - min) of a data vector, and add 3% of the range to the max, and subtract 3% from the min. Pools the data first to generate one set of axis limits for both data vectors.

Value

A numeric 2-element vector, the buffered axis limits for plotting the data in both dimensions.

---

fun_identity *Identity Function*

Description

Apply no transformation to age.

Usage

```
fun_identity(x, ...)
```

Arguments

x	Numeric, the age in years
...	A catch-all for extra arguments (which will not be used) to allow pipeline functions to always pass 2 parameters into age transformation functions

Value

A numeric, the identical value as the input.

Examples

```
fun_identity(1)
fun_identity(c(1, 2, 1))
```

`fun_llin.trans` *Log-Linear v1 Transformation and Inverse*

Description

Apply the Log-Linear v1 transformation to age, or undo the Log-Linear v1 transformation to calculate DNAm age. Requires age of sexual maturity and maximum lifespan.

Usage

```
fun_llin.trans(x, maturity, max)

fun_llin.inv(y, maturity, max)
```

Arguments

`x`	Numeric, the age in years
`maturity`	Numeric, the age of sexual maturity in years
`max`	Numeric, the maximum lifespan in years
`y`	Numeric, the transformed value of age (the un-transformed age is in years)

Details

The Log-Linear v1 transformation of age is defined as

$$LLinAge = \begin{cases} \log\left(\frac{Age+k}{maturity+k}\right) & , \ Age \leq maturity \\ \frac{Age-maturity}{maturity+k} & , \ Age \geq maturity \end{cases},$$

where k is the offset parameter.

The inverse of the Log-Linear v1 transformation of age is defined as

$$Age = \begin{cases} (maturity+k) * e^{LLinAge} - k & , \ LLinAge \leq 0 \\ (maturity+k) * LLinAge + maturity & , \ LLinAge \geq 0 \end{cases},$$

where k is the offset parameter.

This transformation is only defined under the condition

$$max > 3 * maturity + 2.$$

k is a function of maturity and max, and is defined as the unique solution to

$$(k + maturity) * \log\left(\frac{k + maturity}{k - 1}\right) = \frac{max - maturity}{2}.$$

Intuitively, k is defined this way so that the linear piece of the function is twice the size of the logarithmic piece in the co-domain.

Value

A numeric, the transformed or inverse-transformed value of the input.

Examples

```
fun_llin.trans(1, 1, 10)
fun_llin.trans(c(1, 2, 1), 1, 10)
fun_llin.trans(c(1, 2, 1), c(1, 10, 10), c(10, 50, 100))
fun_llin.inv(0, 1, 10)
fun_llin.inv(c(0, -1, 1), 1, 10)
fun_llin.inv(c(0, -1, 1), c(1, 10, 10), c(10, 50, 100))
```

---

fun_llin2.trans *Log-Linear v2 Transformation and Inverse*

Description

Apply the Log-Linear v2 transformation to age, or undo the Log-Linear v2 transformation to calculate DNAm age. Requires age of sexual maturity and gestational period.

Usage

```
fun_llin2.trans(x, maturity, gestation)

fun_llin2.inv(y, maturity, gestation)
```

Arguments

x	Numeric, the age in years
maturity	Numeric, the age of sexual maturity in years
gestation	Numeric, the gestational period in years
y	Numeric, the transformed value of age (the un-transformed age is in years)

Details

The Log-Linear v2 transformation of age is defined as

$$LLin2Age = \begin{cases} \log\left(\frac{Age+gestation}{1.5*maturity+gestation}\right) & , Age \leq 1.5 * maturity \\ \frac{Age-1.5*maturity}{1.5*maturity+gestation} & , Age \geq 1.5 * maturity \end{cases} .$$

The inverse of the Log-Linear v2 transformation of age is defined as

$$Age = \begin{cases} (1.5 * maturity + gestation) * e^{LLin2Age} - gestation & , LLin2Age \leq 0 \\ (1.5 * maturity + gestation) * LLin2Age + 1.5 * maturity & , LLin2Age \geq 0 \end{cases} .$$

Value

A numeric, the transformed or inverse-transformed value of the input.

Examples

```
fun_llin2.trans(1, 1, 0.05)
fun_llin2.trans(c(1, 2, 1), 1, 0.05)
fun_llin2.trans(c(1, 2, 1), c(1, 10, 10), c(0.05, 0.25, 0.5))
fun_llin2.inv(0, 1, 0.05)
fun_llin2.inv(c(0, -1, 1), 1, 0.05)
fun_llin2.inv(c(0, -1, 1), c(1, 10, 10), c(0.05, 0.25, 0.5))
```

---

fun_llin3.trans *Log-Linear v3 Transformation and Inverse*

Description

Apply the Log-Linear v3 transformation to age, or undo the Log-Linear v3 transformation to calculate DNAm age. Requires age of sexual maturity.

Usage

```
fun_llin3.trans(x, maturity, ...)

fun_llin3.inv(y, maturity, ...)
```

Arguments

x	Numeric, the age in years
maturity	Numeric, the age of sexual maturity in years
...	A catch-all for extra arguments (which will not be used) to allow pipeline functions to always pass 2 parameters into age transformation functions
y	Numeric, the transformed value of age (the un-transformed age is in years)

Details

The Log-Linear v3 transformation of age is defined as

$$LLin3Age = \begin{cases} \log\left(\frac{Age+1.5}{maturity+1.5}\right) & , Age \leq maturity \\ \frac{Age-maturity}{maturity+1.5} & , Age \geq maturity \end{cases}.$$

The inverse of the Log-Linear v3 transformation of age is defined as

$$Age = \begin{cases} (maturity+1.5) * e^{LLin3Age} - 1.5 & , LLin3Age \leq 0 \\ (maturity+1.5) * LLin3Age + maturity & , LLin3Age \geq 0 \end{cases}.$$

Value

A numeric, the transformed or inverse-transformed value of the input.

Examples

```
fun_llin3.trans(1, 1)
fun_llin3.trans(c(1, 2, 1), 1)
fun_llin3.trans(c(1, 2, 1), c(1, 10, 10))
fun_llin3.inv(0, 1)
fun_llin3.inv(c(0, -1, 1), 1)
fun_llin3.inv(c(0, -1, 1), c(1, 10, 10))
```

`fun_llinmouse.trans` *Log-Linear Mouse-Specific Transformation and Inverse*

Description

Apply the Log-Linear Mouse-Specific transformation to age, or undo the Log-Linear Mouse-Specific transformation to calculate DNAm age.

This is the transformation used to parameterize the all of the Mouse clocks (Mozhui et. al., 2022).

Usage

```
fun_llinmouse.trans(x, ...)

fun_llinmouse.inv(y, ...)
```

Arguments

x	Numeric, the age in years
...	A catch-all for extra arguments (which will not be used) to allow pipeline functions to always pass 2 parameters into age transformation functions
y	Numeric, the transformed value of age (the un-transformed age is in years)

Details

The Log-Linear Mouse-Specific transformation of age is defined as

$$LLinMouseAge = \begin{cases} \log(x + 0.06) & , Age \leq 1.2 \\ \frac{x-1.2}{1.26} + log(1.26) & , Age \geq 1.2 \end{cases}.$$

The inverse of the Log-Linear Mouse-Specific transformation of age is defined as

$$Age = \begin{cases} (0) * e^{LLinMouseAge} - 0 & , LLinMouseAge \leq 0 \\ (0) * LLinMouseAge + 0 & , LLinMouseAge \geq 0 \end{cases}.$$

Value

A numeric, the transformed or inverse-transformed value of the input.

Examples

```
fun_llinmouse.trans(1)
fun_llinmouse.trans(c(1, 2, 1))
fun_llinmouse.inv(1)
fun_llinmouse.inv(c(1, 2, 1))
```

`fun_llinreladult.trans` *Log-Linear Relative Adult Transformation and Inverse*

Description

Apply the Log-Linear Relative Adult transformation to age, or undo the Log-Linear Relative Adult transformation to calculate DNAm age. Requires age of sexual maturity and gestational period.

This is the transformation used to parameterize the Universal Mammalian "Clock 3" (Lu et. al., 2023).

Usage

```
fun_llinreladult.trans(x, maturity, gestation)

fun_llinreladult.inv(y, maturity, gestation)
```

Arguments

`x`	Numeric, the age in years
`maturity`	Numeric, the age of sexual maturity in years
`gestation`	Numeric, the gestational period in years
`y`	Numeric, the transformed value of age (the un-transformed age is in years)

Details

The Log-Linear Relative Adult transformation of age is defined, in two steps, as

$$LLinRelAdultAge = \begin{cases} \log\left(\frac{RelAdultAge}{\hat{m}}\right) & , RelAdultAge \leq \hat{m} \\ \frac{RelAdultAge - \hat{m}}{\hat{m}} & , RelAdultAge \geq \hat{m} \end{cases},$$

where

$$RelAdultAge = \frac{Age + gestation}{maturity + gestation},$$

$$\hat{m} = 5.0 * \left(\frac{gestation}{maturity}\right)^{0.38}.$$

The inverse of the Log-Linear Relative Adult transformation of age is defined, in two steps, as

$$Age = (maturity + gestation) * RelAdultAge - gestation,$$

where

$$RelAdultAge = \begin{cases} \hat{m} * e^{LLinRelAdultAge} & , LLinRelAdultAge \leq 0 \\ \hat{m} * LLinRelAdultAge + \hat{m} & , LLinRelAdultAge \geq 0 \end{cases},$$

$$\hat{m} = 5.0 * \left(\frac{gestation}{maturity}\right)^{0.38}.$$

Value

A numeric, the transformed or inverse-transformed value of the input.

Examples

```
fun_llinreladult.trans(1, 1, 0.05)
fun_llinreladult.trans(c(1, 2, 1), 1, 0.05)
fun_llinreladult.trans(c(1, 2, 1), c(1, 10, 10), c(0.05, 0.25, 0.5))
fun_llinreladult.inv(0, 1, 0.05)
fun_llinreladult.inv(c(0, -1, 1), 1, 0.05)
fun_llinreladult.inv(c(0, -1, 1), c(1, 10, 10), c(0.05, 0.25, 0.5))
```

---

fun_log.trans *Logarithmic Transformation and Inverse*

Description

Apply the logarithmic transformation to age, or undo the logarithmic transformation to calculate DNAm age.

Usage

```
fun_log.trans(x, ...)

fun_log.inv(y, ...)
```

Arguments

x	Numeric, the age in years
...	A catch-all for extra arguments (which will not be used) to allow pipeline functions to always pass 2 parameters into age transformation functions
y	Numeric, the transformed value of age (the un-transformed age is in years)

Details

The logarithmic transformation of age is defined as

$$LogAge = log(Age + 1)$$

The inverse of the logarithmic transformation of age is defined as

$$Age = exp(LogAge) - 1$$

Value

A numeric, the transformed or inverse-transformed value of the input.

Examples

```
fun_log.trans(1)
fun_log.trans(c(1, 2, 1))
fun_log.inv(1)
fun_log.inv(c(1, 2, 1))
```

`fun_log2.trans` *Logarithmic v2 Transformation and Inverse*

Description

Apply the logarithmic transformation to age, or undo the logarithmic transformation to calculate DNAm age.

Usage

```
fun_log2.trans(x, ...)

fun_log2.inv(y, ...)
```

Arguments

x	Numeric, the age in years
...	A catch-all for extra arguments (which will not be used) to allow pipeline functions to always pass 2 parameters into age transformation functions
y	Numeric, the transformed value of age (the un-transformed age is in years)

Details

The logarithmic transformation of age is defined as

$$Log2Age = log(Age + 0.5)$$

The inverse of the logarithmic transformation of age is defined as

$$Age = exp(Log2Age) - 0.5$$

Value

A numeric, the transformed or inverse-transformed value of the input.

Examples

```
fun_log.trans(1)
fun_log.trans(c(1, 2, 1))
fun_log.inv(1)
fun_log.inv(c(1, 2, 1))
```

`fun_relative.trans` *Relative Age Transformation and Inverse*

Description

Convert age into relative age, or convert relative age into age.

Usage

```
fun_relative.trans(x, max)

fun_relative.inv(y, max)
```

Arguments

`x`	Numeric, the age in years
`max`	Numeric, the maximum lifespan in years
`y`	Numeric, the transformed value of age (the un-transformed age is in years)

Details

Relative age is defined as

$$RelAge = \frac{Age}{max}$$

Relative age is converted back to Age via

$$Age = max * RelAge$$

Value

A numeric, the transformed or inverse-transformed value of the input.

Examples

```
fun_relative.trans(1, 10)
fun_relative.trans(c(1, 2, 1), 10)
fun_relative.trans(c(1, 2, 1), c(10, 50, 100))
fun_relative.inv(0.1, 10)
fun_relative.inv(c(0.1, 0.2, 0.1), 10)
fun_relative.inv(c(0.1, 0.2, 0.1), c(10, 50, 100))
```

`fun_sqrt.trans` *Square-Root Transformation and Inverse*

Description

Apply the square root transformation to age, or undo the square root transformation to calculate DNAm age.

Usage

```
fun_sqrt.trans(x, ...)

fun_sqrt.inv(y, ...)
```

Arguments

`x`	Numeric, the age in years
`...`	A catch-all for extra arguments (which will not be used) to allow pipeline functions to always pass 2 parameters into age transformation functions
`y`	Numeric, the transformed value of age (the un-transformed age is in years)

Details

The square root transformation of age is defined as

$$SqrtAge = sqrt(Age + 1)$$

The inverse of the square root transformation of age is defined as

$$Age = (SqrtAge)^2 - 1$$

Value

A numeric, the transformed or inverse-transformed value of the input.

Examples

```
fun_sqrt.trans(1)
fun_sqrt.trans(c(1, 2, 1))
fun_sqrt.inv(1)
fun_sqrt.inv(c(1, 2, 1))
```

loadRData *Load RData File as an Object*

Description

Load an RData file containing one object, and capture said object rather than automatically adding it to the workspace.

Usage

```
loadRData(data.rdata)
```

Arguments

data.rdata	String, the name of the file to load (ending in .rdata)

Details

RData files contain the contents of a workspace when they are created, so loading them does not directly capture objects. When an RData file contains only one object (like a dataframe), it is more convenient to have a function that loads the RData file and captures the single object.

Value

An object, being the single object saved within the RData file.

panel.cor *Add Correlation as Text to a Plot*

Description

Utility function which "plots" a correlation value in large text.

Usage

```
panel.cor(x, y, cex.cor, ...)
```

Arguments

x, y	Numeric vectors with compatible dimensions
cex.cor	Numeric, the character expansion factor for the text that will be plotted (1 for no change)
...	A catch-all for extra arguments (which will not be used)

Details

This utility function is made to help in plotting stratified EWAS results (for data containing multiple tissue types of species), when calling graphics::pairs() on a table stratified z-scores. It is used in either the lower.panel or upper.panel argument of graphics::pairs(), in order to report to the correlation of each pair-wise plot.

The purpose of the catch-all ... for extra arguments is simply to allow this function to be passed through as a valid panel function in the lower.panel or upper.panel arguments of graphics::pairs().

Examples

```
ewas.meta.table <- data.frame(X = c(0), Y = c(0))
pairs(ewas.meta.table[, 2:(ncol(ewas.meta.table) - 2)],
  lower.panel = panel.cor, upper.panel = points, pch = 16,
  cex.cor = 1.2
)
```

`paste.species_tissue` *Concatenate Species and Tissue Names*

Description

Pair-wise concatenate a vector of distinct species names and distinct tissue names, with the option of ignoring certain species. This is useful for plotting a set of species and tissues in a single color scheme, using a single legend.

Usage

```
paste.species_tissue(spec_names, tissue_names, ignore = c("Homo sapiens"))
```

Arguments

`spec_names`	String vector, containing one or more species names
`tissue_names`	String vector, containing one or more tissue names
`ignore`	String vector, a subset of spec_names, the species names to be left unmodified (default `c("Homo sapiens")`)

Details

Uses `base::paste()` with sep='.'.

Value

A string vector of the concatenated values, all of the species-tissue combinations, joined separated by periods ('.').

Examples

```
paste.species_tissue(c("Mus musculus", "Homo sapiens"), c("Blood", "Liver", "Skin"))
```

predictAge *Automatically Apply All Published Clocks to a New Data Set and Generate Predicted Values*

Description

Applies all previously built & published Clock that are potentially applicable (based on tissue type(s) and species) to a given data set (an aligned metadata `ys.new` and an aligned and normalized methylation data `xs.new`). Ensure that the methylation array used to generate `xs.new` is the same array used to build all of the published Clocks (in other words, the Horvath Mammalian Chip).

It is **very important** to note that if the tissue(s) and species name(s) being provided in the arguments match those used to generate the methylation data being provided, because Clocks are **only designed** to be accurate with the set of tissues and set of species on which they were trained.

Files generated and saved by this function:

- Table of metadata appended with all predictions, using all applicable and published Clocks.

Usage

```
predictAge(xs.new, ys.new, tissue.names, species.name)

getClockDatabase()

getAnAgeTable()
```

Arguments

`xs.new`	Data frame, the aligned and normalized methylation data for the samples in the new, independent set (rows are samples). To format `xs.new` correctly, use `alignDatToInfo()`.
`ys.new`	Data frame, the aligned metadata for the samples in the new, independent set NOTE: This can be a data frame with only column, e.g., sample IDs. To format `ys.new` correctly, use `alignDatToInfo()`.
`tissue.names`	String OR String Vector, the tissue(s) from which the samples in `xs.new` were generated. NOTE: These tissue names must match what are encoded in the internal database of existing clocks; see `getClockDatabase()` To ignore tissue type compatibility, use `tissue.names="ALL"`.
`species.name`	String, the species latin name for the species from which the samples in `xs.new` were generated. NOTE: This species latin name must match what is encoded in the internal database of existing clocks; see `getClockDatabase()`. For species not in the database, the universal clock will still be applied, and the species latin name must match what is encoded in the internal anAge table; see `getAnAgeTable()` To ignore species compatibility, use `species.name="ALL"`.

Details

The output of this function is a data frame that copies `ys.new` and appends one or more additional new columns.

The appended columns will be named (automatically) to specify each clock being applied.

For all coefficient files and age transformations for all published Clocks, see the package's GitHub repository.

Value

A data frame containing metadata and all predicted values, created by copying `ys.new` and appending predicted values as new columns. The names of the new columns are generated automatically to describe which clock is being used.

`predictClockSimple` *Apply a Clock to Methylation Data*

Description

Applies a previously built Clock to normalized methylation data `xs.new`. Ensure that the set of probes used to define the Clock are present within the column names of `xs.new` (in order words, that the methylation array used to build the Clock is the same array used to generate `xs.new`)

It is **very important** to note that if the Clock being provided was built using a transformation of the outcome variable (e.g. an age transformation), then you **must** provide the inverse transformation in order to properly generate predicted values.

Usage

```
predictClockSimple(
  in.valbeta,
  xs.new,
  fun_inv = fun_identity,
  fun_VAR1_val = NA,
  fun_VAR2_val = NA
)
```

Arguments

`in.valbeta`	2-column Matrix or Data frame, the table of coefficients for the built Clock (column 1=probe, column 2=coefficient)
`xs.new`	Data frame, the aligned and normalized methylation data for the samples in the new, independent set (rows are samples). To format `xs.new` correctly, use `alignDatToInfo()`.
`fun_inv`	Function, the name of an R function that inverse transforms the weighted sum of methylation values, with up to two additional parameters. This function, if it is present, is part of the definition of the Clock being applied. For more information, see `saveBuildClock()`
`fun_VAR1_val`	Numeric OR Numeric vector, the value of the first parameter of `fun_inv`, if any, for each row of `xs.new`(default is `NA`, meaning no parameter is used)
`fun_VAR2_val`	Numeric OR Numeric vector, the value of the second parameter of `fun_inv`, if any, for each row of `xs.new`(default is `NA`, meaning no parameter is used)

Details

The output of this function is a vector containing the predicted outcome variable, the output of the Clock (e.g., DNAm Age)

For all details related to building Clocks, see Details section in `saveBuildClock()`.

For efficiently applying all published Clocks, and for information on using these Clocks, see Details sections in `predictAge()`.

Value

A vector containing predicted values, where each entry corresponds to each row of `xs.new`.

---

rbind2.complete *Maximally Combine Data Frames by Rows*

Description

Merge two data frames with `base::rbind()`, keeping all column names present in either data frame.

Usage

```
rbind2.complete(x, y, fill = NA)
```

Arguments

`x, y`	Data frames
`fill`	Missing columns in one data frame will be filled with this value (default is `NA`)

Details

Before calling `base::rbind()`, it expands the columns of x and y so that they both share all of their columns. If new columns are created, they are filled with the value of fill (with the default being `NA`)

Value

A data frame, the result of merging x and y by rows.

rbind2.inner *Minimally Combine Data Frames by Rows*

Description

Merge two data frames with `base::rbind()`, only keeping column names present in both data frames.

Usage

```
rbind2.inner(x, y)
```

Arguments

`x, y`	Data frames

Details

Before calling `base::rbind()`, it filters the columns of x and y to those that they both share

Value

A data frame, the result of merging x and y by rows.

reduct_factor *Character Expansion Multiplier for Multi-Panel Plots*

Description

Returns the multiplier (reduction factor) that is automatically applied to the cex in `base::plot()` when creating multi-panel plots.

Usage

```
reduct_factor(mfrow)
```

Arguments

`mfrow`	2-element Numeric Vector, the dimensions of the multi-panel plot layout, given by `c(nrows, ncols)`

Details

Multi-panel plots can be made using the base `base::plot()` in R, by setting the value of `mfrow`. It is not well-known that `base::plot()` applies an automatic reduction to the cex (character expansion factor), which sets the relative size of all texts.

Specifically, this reduction is 0.83 if 2x2 plotting, and 0.66 if `mfrow[1]` or `mfrow[2]` is 3 or greater.

Value

A numeric, the multiplier (reduction factor) applied to the cex.

Examples

```
reduct_factor(mfrow = c(1, 2))
reduct_factor(mfrow = c(2, 2))
reduct_factor(mfrow = c(1, 3))
reduct_factor(mfrow = c(2, 3))
```

---

relevel.last *Reorder Levels of Factor*

Description

The levels of a factor are re-ordered so that the level specified by ref is first and the others are moved up.

Usage

```
## S3 method for class 'last'
relevel(x, ref, ...)
```

Arguments

x	Factor
ref	String, the reference level of x to be moved to last
...	Arguments to be passed to stats::relevel() method

Value

A factor of the same length as x.

---

saveApplyClock *Apply a Clock to a New Data Set and Generate Predicted Values*

Description

Applies a previously built Clock to a given data set (an aligned metadata ys.new and an aligned and normalized methylation data xs.new). Ensure that the set of probes used to define the Clock are present within the column names of xs.new (in order words, that the methylation array used to build the Clock is the same array used to generate xs.new)

It is **very important** to note that if the Clock being provided was built using a transformation of the outcome variable (e.g. an age transformation), then you **must** provide the inverse transformation in order to properly generate predicted values.

Files generated and saved by this function:

- Table of metadata appended with all Clock predictions,
- Plot of Clock predictions.

Usage

```
saveApplyClock(
  in.valbeta,
  xs.new,
  ys.new,
  OUTVAR,
  output.csv,
  out.png,
  out.png.title,
  PREDVAR,
  RESVAR,
  fun_inv = fun_identity,
  fun_VAR1 = NULL,
  fun_VAR2 = NULL,
  COLVAR = NULL,
  show.legend = T,
  oma.right = 9,
  square.axes = T
)
```

Arguments

in.valbeta	2-column Matrix or Data frame, the table of coefficients for the built Clock (column 1=probe, column 2=coefficient)
xs.new	Data frame, the aligned and normalized methylation data for the samples in the new, independent set (rows are samples). To format xs.new correctly, use alignDatToInfo().
ys.new	Data frame, the aligned metadata for the samples in the new, independent set. To format ys.new correctly, use alignDatToInfo().
OUTVAR	String, the name of the column in ys.new containing the outcome variable (typically Age, in years)
output.csv	File name/File path String (must end in ".csv"), the name and location of the file where the output data frame with predicted values will be saved. For more information, see Details section below To skip this, use output.csv=NULL.
out.png	File name/File path String (must end in ".png"), the name and location of the file where the plot containing Clock predictions vs. true values will be saved To skip this, use out.png=NULL.
out.png.title	String, the title to be used in the plot saved to out.png
PREDVAR	String, the name of the NEW appended column in the output which will contain the predicted outcome variable, the output of the Clock (same units as OUTVAR, and typically DNAm Age)
RESVAR	String, the name of the NEW appended column in the output which will contain the regression residual of the Clock, the difference of PREDVAR and OUTVAR (same units as OUTVAR, and typically Age Acceleration)
fun_inv	Function, the name of an R function that inverse transforms the weighted sum of methylation values, with up to two additional parameters. This function, if it is present, is part of the definition of the Clock being applied. For more information, see saveBuildClock()

fun_VAR1	String, the name of the column in ys.new containing the first parameter of fun_inv, if any (default is NULL, meaning no parameter is required)
fun_VAR2	String, the name of the column in ys.new containing the second parameter of fun_inv, if any (default is NULL, meaning no parameter is required)
COLVAR	String, the name of the column in ys.new containing the coloring variable for the plot (default is NULL, does not color points). NOTE: This variable must be encoded as a base::factor().
show.legend	Logical, Should the plot include a legend for COLVAR? (default is TRUE)
oma.right	Numeric, the size of the right outer margin for the plot, in lines of text (default is 9). Increase this value if the legend is being truncated (the image width will be automatically adjusted).
square.axes	Logical, Should the axis limits for the x-axis and y-axis be forced to be equal? (default is TRUE, otherwise axis limits are determined independently)

Details

While this function only returns a single data frame, it creates and saves an additional plot directly to a new file in the workspace.

The output of this function is a data frame that copies ys.new and appends two additional new columns.

The two appended column will be named based on the values (given by the user) of the arguments PREDVAR and RESVAR.

For all details related to building Clocks, see Details section in saveBuildClock().

For efficiently applying all published Clocks, and for information on using these Clocks, see Details sections in predictAge().

Value

A data frame containing metadata and predicted values, created by copying ys.new and appending predicted values as new columns. The names of the new columns are specified by the arguments PREDVAR and RESVAR.

---

saveBuildClock *Build a New Clock based on a Training Data Set*

Description

Fit an elastic net regression model, trained by using an aligned metadata ys.train and an aligned and normalized methylation data xs.train. Elastic net regression results in a sparse model, meaning that most probes will not be selected in the final fitted model.

Files generated and saved by this function:

- Table of Clock coefficients,
- Table of metadata appended with Clock predictions,
- Panel plot of regression diagnostics and Clock predictions.

WARNING: This function calls glmnet::cv.glmnet(), so the results of this function are subject to randomness. Users should add line in their R scripts calling set.seed() before calling this function, in order for the results to be reproducible and consistent

Usage

```
saveBuildClock(
  xs.train,
  ys.train,
  xs.other,
  ys.other,
  OUTVAR,
  out.csv,
  output.csv,
  out.png,
  out.png.title,
  PREDVAR,
  RESVAR,
  RESinOtherVAR,
  ALPHA = 0.5,
  NFOLD = 10,
  fun_trans = fun_identity,
  fun_inv = fun_identity,
  fun_VAR1 = NULL,
  fun_VAR2 = NULL,
  loglambda.seq = NULL,
  alpha.seq = NULL
)
```

Arguments

xs.train	Data frame, the aligned and normalized methylation data for the samples in the training set (rows are samples). To format xs.train correctly, use alignDatToInfo().
ys.train	Data frame, the aligned metadata for the samples in the training set. To format ys.train correctly, use alignDatToInfo().
xs.other	Data frame, the aligned and normalized methylation data for additional samples, to which the train Clock should generate predictions. For example, samples you are considering adding to the training set, or samples from an experimental or comparison group (rows are samples). NOTE: xs.other should be compatible with xs.train, see Details section below. To skip this, use xs.other=NULL. To format xs.other correctly, use alignDatToInfo().
ys.other	Data frame, the aligned metadata for additional samples, to which the train Clock should generate predictions. For example, samples you are considering adding to the training set, or samples from an experimental or comparison group. NOTE: ys.other should be compatible with ys.train, see Details section below. To skip this, use ys.other=NULL. To format ys.other correctly, use alignDatToInfo().
OUTVAR	String, the name of the column in ys.train and ys.other containing the outcome variable (typically Age, in years)

out.csv	File name/File path String (must end in ".csv"), the name and location of the file where the table of coefficients for the Clock will be saved. For more information, see Details section below
output.csv	File name/File path String (must end in ".csv"), the name and location of the file where the output data frame with predicted values will be saved. For more information, see Details section below To skip this, use output.csv=NULL.
out.png	File name/File path String (must end in ".png"), the name and location of the file where the panel plot containing regression diagnostics and Clock predictions will be saved. For description of the elastic net regression diagnostics, see glmnet::plot.cv.glmnet() and glmnet::plot.glmnet() To skip this, use out.png=NULL.
out.png.title	String, the title to be used in the panel plot saved to out.png
PREDVAR	String, the name of the NEW appended column in the output which will contain the predicted outcome variable, the output of the Clock (same units as OUTVAR, and typically DNAm Age)
RESVAR	String, the name of the NEW appended column in the output which will contain the regression residual of the Clock, the difference of PREDVAR and OUTVAR (same units as OUTVAR, and typically Age Acceleration)
RESinOtherVAR	String, the name of the NEW appended column in the output which will contain the regression residual of the Clock restricted to the samples in ys.other. To skip this, use RESinOtherVAR=NULL.
ALPHA	Numeric, the elasticnet mixing parameter, to be passed into glmnet::glmnet() and glmnet::cv.glmnet() (default is 0.5). WARNING: Do not change this argument unless you understand what it is; the option to change it is mostly a formality.
NFOLD	Numeric, the number of folds used for selecting lambda, to be passed into glmnet::cv.glmnet() (default is 10). WARNING: Do not change this argument unless you understand what it is; it will change computation time, at the cost of model fitting optimality.
fun_trans	Function, the name of an R function that transforms the outcome variable, with up to two additional parameters. For more information, see Details section below
fun_inv	Function, the name of an R function that is the mathematical inverse of fun_trans, with the same parameters
fun_VAR1	String, the name of the column in ys.train and ys.other containing the first parameter of fun_trans/fun_inv, if any (default is NULL, meaning no parameter is required)
fun_VAR2	String, the name of the column in ys.train and ys.other containing the second parameter of fun_trans/fun_inv, if any (default is NULL, meaning no parameter is required)
loglambda.seq	Numeric Vector, the sequence of lambda values, on the natural log scale, to be evaluated in glmnet::cv.glmnet(), instead of the sequence automatically generated (default is NULL, which automatically selects 100 values, evenly spaced on the natural log scale). WARNING: Do not change this argument unless you understand what it is; it should only be modified in the case of highly heterogeneous data, when the optimal value of lambda might be lower than the range of values automatically generated.

`alpha.seq` Numeric Vector, the sequence of `alpha` values to be evaluated in `glmnetUtils::cva.glmnet()`, instead of forcing a pre-specified value of `alpha` (default is `NULL`, which skips this step and simply uses the value of the `ALPHA` argument). A general recommended choice is `alpha.seq = seq(0,1,len=11)^3`.

This feature requires the package glmnetUtils::glmnetUtils to be installed. The function `glmnetUtils::cva.glmnet()` will select the optimal value of alpha from the provided sequence using internal cross-validation, just like how lambda is selected.

WARNING: Do not change this argument unless you understand what it is; it should only be modified in the case of poor prediction when using `alpha = 0.5`. Beware of values of `alpha` close to `0`, as this allows for less sparsity, which can lead to over-fitting to the training data.

Details

While this function only returns a single data frame, it creates and saves an additional table and plot directly to new files in the workspace.

This function builds a Clock using elastic net regression (see `glmnet::glmnet()`. The "optimal value" of the penalty parameter `lamda` is selected using `glmnet::cv.glmnet()`; specifically, it uses the value `lambda.1se` (which is the default). This is often referred to as the "one-standard-error rule" (see Friedman & Hastie, 2010). The value of `alpha` is set to `0.5` by default, to enforce moderate sparsity and avoid over-fitting to the training data.

The output of this function is a data frame that stacks `ys.train` and `ys.other`, and then appends four additional new columns. It is expected that `ys.train` and `ys.other` (as well as `xs.train` and `xs.other`) be compatible, meaning that they share the same columns and column names, in the same order. If `ys.train` and `ys.other` are not compatible, this function will only keeping column names present in both data frames, prior to row-binding them (see `rbind2.inner()`).

The first appended column is `"train"`, which indicates whether or not a sample was part of the training set (`1` means used in training, `0` means not used in training). The additional three appended column will be named based on the values (given by the user) of the arguments `PREDVAR`, `RESVAR`, and `RESinOtherVAR`. If `RESinOtherVAR=NULL`, then this column will not be appended.

The selected set of probes and the coefficients for the Clock are not returned, but are instead saved in the file given to the argument `out.csv`. This table of coefficients contains two columns, named `"var"` and `"beta"`. The first column, `"var"`, contains the names of the probes selected by the Clock (using the column names of `xs.train`). The second column, `"beta"`, contains the regression coefficients of the Clock, which define the fitted regression model. The regression model is defined by:

$$Age = x \cdot \beta,$$

where β is the vector of true coefficients (or weights) and x is the vector of methylation values of a given sample, including an intercept term. The fitted regression model is used to calculate DNAm Age for a given sample, as follows:

$$DNAmAge = x \cdot \hat{\beta},$$

where $\hat{\beta}$ is the sparse vector of estimated coefficients.

If you are using a transformation of the outcome variable, the regression model for the Clock fundamentally changes:

$$F(Age) = x \cdot \beta,$$

where $F()$ is the transformation of the outcome variable (possibly with species-specific parameters, which are unwritten). It is *very important* to note that this changes the definition of DNAm Age as

well:

$$DNAmAge = F^{-1}(x \cdot \hat{\beta}),$$

and that the inverse transformation is now tied to the definition of the Clock.

If you choose not to use one of the transformation functions provided in this package, then it is up to you, the user, to ensure that `fun_trans` and `fun_inv` are mathematical inverse of one another.

Value

A data frame containing metadata and predicted values, created by stacking `ys.train` and `ys.other`, and then appending predicted values as new columns. The names of the new columns are specified by the arguments `PREDVAR`, `RESVAR`, and `RESinOtherVAR`.

`saveBuildClockStrat` *Build a Set of Independent New Clocks based on a Training Data Set, Stratified by Species*

Description

Fit a set of independent elastic net regression models, trained by stratifying and using an aligned metadata `ys.train` and an aligned and normalized methylation data `xs.train`. Elastic net regression results in a sparse model, meaning that most probes will not be selected in a final fitted model.

This function should be used if you have a significant number of samples from multiple different species, and would like to a set of build single-species Clocks simultaneously.

Files generated and saved by this function:

- Table of all Clock coefficients,
- Table of metadata appended with all Clock predictions,
- Panel plot of all Clock predictions (each panel is one stratum).

WARNING: This function calls `glmnet::cv.glmnet()`, so the results of this function are subject to randomness. Users should add line in their R scripts calling `set.seed()` before calling this function, in order for the results to be reproducible and consistent

Usage

```
saveBuildClockStrat(
  xs.train,
  ys.train,
  xs.other,
  ys.other,
  OUTVAR,
  STRATVAR,
  out.csv,
  output.csv,
  out.png,
  out.png.title,
  PREDVAR,
  RESVAR,
  RESinOtherVAR,
  ALPHA = 0.5,
```

```
  NFOLD = 10,
  fun_trans = fun_identity,
  fun_inv = fun_identity,
  fun_VAR1 = NULL,
  fun_VAR2 = NULL,
  COLVAR = NULL,
  show.original = T,
  loglambda.seq = NULL,
  alpha.seq = NULL
)
```

Arguments

xs.train	Data frame, the aligned and normalized methylation data for the samples in the training set (rows are samples). To format xs.train correctly, use alignDatToInfo().
ys.train	Data frame, the aligned metadata for the samples in the training set. To format ys.train correctly, use alignDatToInfo().
xs.other	Data frame, the aligned and normalized methylation data for additional samples, to which the train Clock should generate predictions. For example, samples you are considering adding to the training set, or samples from an experimental or comparison group (rows are samples). NOTE: xs.other should be compatible with xs.train, see Details section below. To skip this, use xs.other=NULL. To format xs.other correctly, use alignDatToInfo().
ys.other	Data frame, the aligned metadata for additional samples, to which the train Clock should generate predictions. For example, samples you are considering adding to the training set, or samples from an experimental or comparison group. NOTE: ys.other should be compatible with ys.train, see Details section below. To skip this, use ys.other=NULL. To format ys.other correctly, use alignDatToInfo().
OUTVAR	String, the name of the column in ys.train and ys.other containing the outcome variable (typically Age, in years)
STRATVAR	String, the name of the column in ys.train and ys.other containing the stratifying variable (typically Species Name), by which to stratify ys.train and build individual Clocks. NOTE: This variable must be encoded as a base::factor().
out.csv	File name/File path String (must end in ".csv"), the name and location of the file where the table of coefficients for the Clock will be saved. For more information, see Details section below
output.csv	File name/File path String (must end in ".csv"), the name and location of the file where the output data frame with predicted values will be saved. For more information, see Details section below To skip this, use output.csv=NULL.
out.png	File name/File path String (must end in ".png"), the name and location of the file where the panel plot containing all Clock predictions vs. true values (each

	panel is one stratum) will be saved. Please note that this plot will contain all the predictions, from both `ys.train` and `ys.other`. Unlike with `saveBuildClock()`, no diagnostic plots are generated. To skip this, use `out.png=NULL`.
`out.png.title`	String, the title to be used in the panel plot saved to `out.png`
`PREDVAR`	String, the name of the NEW appended column in the output which will contain the predicted outcome variable, the output of the corresponding Clock for each stratum (same units as `OUTVAR`, and typically DNAm Age)
`RESVAR`	String, the name of the NEW appended column in the output which will contain the regression residual of the corresponding Clock for each stratum, the difference of `PREDVAR` and `OUTVAR` (same units as `OUTVAR`, and typically Age Acceleration)
`RESinOtherVAR`	String, the name of the NEW appended column in the output which will contain the regression residual of the corresponding Clock for each stratum, restricted to the samples in `ys.other`. To skip this, use `RESinOtherVAR=NULL`.
`ALPHA`	Numeric, the elasticnet mixing parameter, to be passed into `glmnet::glmnet()` and `glmnet::cv.glmnet()` (default is `0.5`). WARNING: Do not change this argument unless you understand what it is; the option to change it is mostly a formality.
`NFOLD`	Numeric, the number of folds used for selecting `lambda`, to be passed into `glmnet::cv.glmnet()` (default is `10`). WARNING: Do not change this argument unless you understand what it is; it will change computation time, at the cost of model fitting optimality.
`fun_trans`	Function, the name of an R function that transforms the outcome variable, with up to two additional parameters. For more information, see `saveBuildClock()`
`fun_inv`	Function, the name of an R function that is the mathematical inverse of `fun_trans`, with the same parameters
`fun_VAR1`	String, the name of the column in `ys.train` and `ys.other` containing the first parameter of `fun_trans`/`fun_inv`, if any (default is `NULL`, meaning no parameter is required)
`fun_VAR2`	String, the name of the column in `ys.train` and `ys.other` containing the second parameter of `fun_trans`/`fun_inv`, if any (default is `NULL`, meaning no parameter is required)
`COLVAR`	String, the name of the column in `ys.train` and `ys.other` containing the coloring variable for the panel plot (default is `NULL`, uses `COLVAR=STRATVAR`). NOTE: This variable must be encoded as a `base::factor()`.
`show.original`	Logical, Should the original full-data plot be shown as the first panel? (default is `TRUE`)
`loglambda.seq`	Numeric Vector, the sequence of `lambda` values, on the natural log scale, to be used in `glmnet::cv.glmnet()`, instead of the ones automatically generated (default is `NULL`, which automatically selects 100 values, evenly spaced on the natural log scale, and independently for each stratum). WARNING: Do not change this argument unless you understand what it is; it should only be modified in the case of highly heterogeneous data, when the optimal value of lambda might be lower than the range of values automatically generated.

alpha.seq — Numeric Vector, the sequence of alpha values to be evaluated in glmnetUtils::cva.glmnet(), instead of forcing a pre-specified value of alpha (default is NULL, which skips this step and simply uses the value of the ALPHA argument). A general recommended choice is alpha.seq = seq(0,1,len=11)^3.

This feature requires the package glmnetUtils::glmnetUtils to be installed. The function glmnetUtils::cva.glmnet() will select the optimal value of alpha from the provided sequence using internal cross-validation, just like how lambda is selected.

WARNING: Do not change this argument unless you understand what it is; it should only be modified in the case of poor prediction when using alpha = 0.5. Beware of values of alpha close to 0, as this allows for less sparsity, which can lead to over-fitting to the training data.

Details

While this function only returns a single data frame, it creates and saves an additional table and plot directly to new files in the workspace.

The output of this function is a data frame that stacks ys.train and ys.other, and then appends four additional new columns. It is expected that ys.train and ys.other (as well as xs.train and xs.other) be compatible, meaning that they share the same columns and column names, in the same order. If ys.train and ys.other are not compatible, this function will only keeping column names present in both data frames, prior to row-binding them (see rbind2.inner()).

The first appended column is "train", which indicates whether or not a sample was part of the training set (1 means used in training, 0 means not used in training). The additional three appended column will be named based on the values (given by the user) of the arguments PREDVAR, RESVAR, and RESinOtherVAR. If RESinOtherVAR=NULL, then this column will not be appended.

The selected set of probes and the coefficients for all of the Clocks are not returned, but are instead saved together in the file given to the argument out.csv. This table of coefficients contains $S + 1$ columns, named "var" and "beta.<level>", where S refers to S == length(levels(ys.train[,SPECVAR])). The first column, "var", contains the names of the probes selected by at least one of the Clocks (using the column names of xs.train). The remaining S columns, "beta.<level>", each contain the regression coefficients of the Clock (corresponding to that stratum), which define the fitted regression model.

For all other details related to building Clocks, see Details section in saveBuildClock().

Value

A data frame containing metadata and predicted values, created by stacking ys.train and ys.other, and then appending predicted values as new columns. The names of the new columns are specified by the arguments PREDVAR, RESVAR, and RESinOtherVAR.

---

saveEWAS *Perform Epigenome-wide Association Study on Outcome Variable*

Description

Performs an epigenome-wide association study (EWAS) on the outcome variable.

Files generated and saved by this function:

- Table of EWAS results.

Usage

```
saveEWAS(
  xs,
  ys,
  OUTVAR,
  ewas.csv,
  fun_trans = fun_identity,
  fun_VAR1 = NULL,
  fun_VAR2 = NULL
)
```

Arguments

xs	Data frame, the aligned and normalized methylation data for the samples used to build a Clock (rows are samples). To format xs correctly, use alignDatToInfo().
ys	Data frame, the aligned metadata for the samples used to build a Clock. To format ys correctly, use alignDatToInfo().
OUTVAR	String, the name of the column in ys containing the outcome variable (typically Age, in years, but can be binary as well)
ewas.csv	File name/File path String (must end in ".csv"), the name and location of the file where the output data frame with the EWAS scores and significant values will be saved. For more information, see Details section below To skip this, use ewas.csv=NULL.
fun_trans	Function, the name of an R function that transforms the outcome variable prior to calculating correlations with every probe, with up to two additional parameters. If a transformation function was used to build a corresponding Clock, then the same function should be provided here. For more information, see saveBuildClock()
fun_VAR1	String, the name of the column in ys containing the first parameter of fun_trans, if any (default is NULL, meaning no parameter is required)
fun_VAR2	String, the name of the column in ys containing the second parameter of fun_trans, if any (default is NULL, meaning no parameter is required)

Details

In this function, the outcome variable is first transformed, using fun_trans, if such a function is provided. Then, with these transformed outcome values as 'traits', and along with the methylation data, the EWAS is performed. For details about the computations, see WGCNA::standardScreeningNumericTrait() (or WGCNA::standardScreeningBinaryTrait()).

Value

A data frame containing the results of the EWAS. For an explanation of each column in the output, see WGCNA::standardScreeningNumericTrait() (or WGCNA::standardScreeningBinaryTrait()).

`saveLOOEstimation` *Generate Unbiased Predictions for a New Clock based on a Training Data Set*

Description

Approximate the out-of-sample behavior of a Clock model by generating unbiased predictions using Leave-One-Out cross-validation. Generates an unbiased version of the Clock predictions using the entire training set of a corresponding Clock (an aligned metadata `ys` and an aligned and normalized methylation data `xs`).

Files generated and saved by this function:

- List of Tables of Clock coefficients of every Leave-One-Out Clock,
- Table of metadata appended with unbiased Clock predictions,
- Plot of unbiased Clock predictions.

WARNING: This function calls `glmnet::cv.glmnet()`, so the results of this function are subject to randomness. Users should add line in their R scripts calling `set.seed()` before calling this function, in order for the results to be reproducible and consistent

Usage

```
saveLOOEstimation(
  xs,
  ys,
  OUTVAR,
  out.rdata,
  output.csv,
  out.png,
  out.png.title,
  PREDVAR,
  RESVAR,
  ALPHA = 0.5,
  NFOLD = 10,
  fun_trans = fun_identity,
  fun_inv = fun_identity,
  fun_VAR1 = NULL,
  fun_VAR2 = NULL,
  COLVAR = NULL,
  NUMVAR = NULL,
  show.legend = T,
  oma.right = 9,
  xs.add_train = NULL,
  ys.add_train = NULL,
  loglambda.seq = NULL,
  alpha.seq = NULL
)
```

Arguments

xs	Data frame, the aligned and normalized methylation data for the samples used to build a Clock (rows are samples). To format xs correctly, use alignDatToInfo().
ys	Data frame, the aligned metadata for the samples used to build a Clock. To format ys correctly, use alignDatToInfo().
OUTVAR	String, the name of the column in ys containing the outcome variable (typically Age, in years)
out.rdata	File name/File path String (must end in ".RData"), the name and location of the file where the list of every table of coefficients for every Leave-One-Out Clock will be saved. For more information, see Details section below To skip this, use out.rdata=NULL.
output.csv	File name/File path String (must end in ".csv"), the name and location of the file where the output data frame with unbiased predicted values will be saved. For more information, see Details section below To skip this, use output.csv=NULL.
out.png	File name/File path String (must end in ".png"), the name and location of the file where the plot containing Leave-One-Out predictions vs. true values will be saved To skip this, use out.png=NULL.
out.png.title	String, the title to be used in the plot saved to out.png
PREDVAR	String, the name of the NEW appended column in the output which will contain the unbiased predicted outcome variable, the output of the corresponding Leave-One-Out Clock (same units as OUTVAR, and typically DNAm Age)
RESVAR	String, the name of the NEW appended column in the output which will contain the residual of the Leave-One-Out predictions against the true values, roughly the difference of PREDVAR and OUTVAR (same units as OUTVAR, and typically Age Acceleration)
ALPHA	Numeric, the elasticnet mixing parameter, to be passed into glmnet::glmnet() and glmnet::cv.glmnet() (default is 0.5). WARNING: Do not change this argument unless you understand what it is; the option to change it is mostly a formality.
NFOLD	Numeric, the number of folds used for selecting lambda, to be passed into glmnet::cv.glmnet() (default is 10). WARNING: Do not change this argument unless you understand what it is; it will change computation time, at the cost of model fitting optimality.
fun_trans	Function, the name of an R function that transforms the outcome variable, with up to two additional parameters. For more information, see saveBuildClock()
fun_inv	Function, the name of an R function that is the mathematical inverse of fun_trans, with the same parameters
fun_VAR1	String, the name of the column in ys containing the first parameter of fun_trans/fun_inv, if any (default is NULL, meaning no parameter is required)
fun_VAR2	String, the name of the column in ys containing the second parameter of fun_trans/fun_inv, if any (default is NULL, meaning no parameter is required)
COLVAR	String, the name of the column in ys containing the coloring variable for the plot (default is NULL, does not color points). NOTE: This variable must be encoded as a base::factor().

NUMVAR — String, the name of the column in ys containing the numbering variable for the plot (default is NULL, does not convert points into number symbols).

NOTE: This variable must be encoded as a base::factor().

show.legend — Logical, Should the plot include a legend for COLVAR? (default is TRUE)

oma.right — Numeric, the size of the right outer margin for the plot, in lines of text (default is 9). Increase this value if the legend is being truncated (the image width will be automatically adjusted).

xs.add_train — Data frame, the aligned and normalized methylation data for additional samples, which will be included in the training set at every iteration of the Leave-One-Out analysis (rows are samples).

To format xs.add_train correctly, use alignDatToInfo().

ys.add_train — Data frame, the aligned metadata for additional samples, which will be included in the training set at every iteration of the Leave-One-Out analysis.

To format ys.add_train correctly, use alignDatToInfo().

loglambda.seq — Numeric Vector, the sequence of lambda values, on the natural log scale, to be evaluated in glmnet::cv.glmnet(), instead of the sequence automatically generated (default is NULL, which automatically selects 100 values, evenly spaced on the natural log scale).

WARNING: Do not change this argument unless you understand what it is; it should only be modified in the case of highly heterogeneous data, when the optimal value of lambda might be lower than the range of values automatically generated.

alpha.seq — Numeric Vector, the sequence of alpha values to be evaluated in glmnetUtils::cva.glmnet(), instead of forcing a pre-specified value of alpha (default is NULL, which skips this step and simply uses the value of the ALPHA argument). A general recommended choice is alpha.seq = seq(0,1,len=11)^3.

This feature requires the package glmnetUtils::glmnetUtils to be installed. The function glmnetUtils::cva.glmnet() will select the optimal value of alpha from the provided sequence using internal cross-validation, just like how lambda is selected.

WARNING: Do not change this argument unless you understand what it is; it should only be modified in the case of poor prediction when using alpha = 0.5. Beware of values of alpha close to 0, as this allows for less sparsity, which can lead to over-fitting to the training data.

Details

While this function only returns a single data frame, it creates and saves an additional plot, and a list of tables, directly to new files in the workspace.

This function performs a Leave-One-Out (LOO) analysis of a specified Clock model. For every sample (every row of ys), it does the following:

- Creates a temporary copy of ys and xs with the selected sample removed,
- Fits a Clock model based on this temporary data set copy, with the same specifications as outlined in saveBuildClock(),
- Saves the table of coefficients of this LOO Clock to a list of matrices,
- Applies this LOO Clock to the sample that was removed, and saves the predicted value (usually called DNAm Age LOO) to the output of this function.

Please note that, in order to be computationally efficient, this function does NOT perform an internal cross-validation at every iteration, to select a potentially different optimal value of the penalty parameter `lambda` for every LOO Clock. Because all of these optimal lambda values will be very close to each other in practice, we instead choose to select ONE optimal lambda value based on the full data set, and use this value of lambda to fit every LOO Clock. The same is true for `alpha`, if `alpha.seq` is specified.

The output of this function is a data frame that copies `ys` and appends three additional new columns.

The first two appended column will be named based on the values (given by the user) of the arguments `PREDVAR` and `RESVAR`. The final appended column is `"count_probes_selected"`, which counts how many probes were selected for in the LOO Clock corresponding to that sample. These values are typically close to each other, and close to the number of probes selected in the corresponding final Clock trained on the full data.

The selected sets of probes and the coefficients for every LOO Clock are not returned, but are instead saved in the file given to the argument `out.rdata`. This file contains a list of tables (matrices) of coefficients, and every table of coefficients within this list contains two columns.

For all details related to building Clocks and using coefficients files, see Details section in `saveBuildClock()`.

Value

A data frame containing metadata and predicted values, created by copying `ys` and unibased appending predicted values as new columns. The names of the new columns are specified by the arguments `PREDVAR` and `RESVAR`.

`saveLOOEstimationStrat` *Generate Unbiased Predictions for a Set of New Clocks based on a Training Data Set, Stratified by Species*

Description

Approximate the out-of-sample behavior of a set of independent Clock models by generating unbiased predictions using stratified Leave-One-Out cross-validation. Generates an unbiased version of the Clock predictions within each stratum, using the entire training set of a corresponding Clock (an aligned metadata `ys` and an aligned and normalized methylation data `xs`).

This function should be used if you have a significant number of samples from multiple different species, and would like to a set of build single-species Clocks simultaneously.

Files generated and saved by this function:

- Multiple Lists of Tables (one for each stratum) of Clock coefficients of every Leave-One-Out Clock,
- Table of metadata appended with all unbiased Clock predictions,
- Plot of all unbiased Clock predictions.

WARNING: This function calls `glmnet::cv.glmnet()`, so the results of this function are subject to randomness. Users should add line in their R scripts calling `set.seed()` before calling this function, in order for the results to be reproducible and consistent

Usage

```
saveLOOEstimationStrat(
  xs,
  ys,
  OUTVAR,
  STRATVAR,
  out.rdata.PREFIX,
  output.csv,
  out.png,
  out.png.title,
  PREDVAR,
  RESVAR,
  ALPHA = 0.5,
  NFOLD = 10,
  fun_trans = fun_identity,
  fun_inv = fun_identity,
  fun_VAR1 = NULL,
  fun_VAR2 = NULL,
  COLVAR = NULL,
  NUMVAR = NULL,
  show.legend = T,
  oma.right = 9,
  xs.add_train = NULL,
  ys.add_train = NULL,
  alpha.seq = NULL
)
```

Arguments

xs	Data frame, the aligned and normalized methylation data for the samples used to build a set of independent Clocks (rows are samples). To format xs correctly, use alignDatToInfo().
ys	Data frame, the aligned metadata for the samples used to build a set of independent Clocks. To format ys correctly, use alignDatToInfo().
OUTVAR	String, the name of the column in ys containing the outcome variable (typically Age, in years)
STRATVAR	String, the name of the column in ys containing the stratifying variable (typically Species Name), by which to stratify ys and perform separate Leave-One-Out analyses. NOTE: This variable must be encoded as a base::factor().
out.rdata.PREFIX	File name/File path String PREFIX (meaning a file name or file path without the '.RData' at the end), the name and location of the files where the lists (one list for each stratum) of every table of coefficients for every Leave-One-Out Clock will be saved. To skip this, use out.rdata.PREFIX=NULL.
output.csv	File name/File path String (must end in ".csv"), the name and location of the file where the output data frame with unbiased predicted values will be saved. For more information, see Details section below To skip this, use output.csv=NULL.

out.png	File name/File path String (must end in ".png"), the name and location of the file where the plot containing Leave-One-Out predictions vs. true values will be saved To skip this, use out.png=NULL.
out.png.title	String, the title to be used in the plot saved to out.png
PREDVAR	String, the name of the NEW appended column in the output which will contain the unbiased predicted outcome variable, the output of the corresponding stratifed Leave-One-Out Clock (same units as OUTVAR, and typically DNAm Age)
RESVAR	String, the name of the NEW appended column in the output which will contain the residual of the Leave-One-Out predictions against the true values, roughly the difference of PREDVAR and OUTVAR (same units as OUTVAR, and typically Age Acceleration)
ALPHA	Numeric, the elasticnet mixing parameter, to be passed into glmnet::glmnet() and glmnet::cv.glmnet() (default is 0.5). WARNING: Do not change this argument unless you understand what it is; the option to change it is mostly a formality.
NFOLD	Numeric, the number of folds used for selecting lambda, to be passed into glmnet::cv.glmnet() (default is 10). WARNING: Do not change this argument unless you understand what it is; it will change computation time, at the cost of model fitting optimality.
fun_trans	Function, the name of an R function that transforms the outcome variable, with up to two additional parameters. For more information, see saveBuildClock()
fun_inv	Function, the name of an R function that is the mathematical inverse of fun_trans, with the same parameters
fun_VAR1	String, the name of the column in ys containing the first parameter of fun_trans/fun_inv, if any (default is NULL, meaning no parameter is required)
fun_VAR2	String, the name of the column in ys containing the second parameter of fun_trans/fun_inv, if any (default is NULL, meaning no parameter is required)
COLVAR	String, the name of the column in ys containing the coloring variable for the plot (default is NULL, does not color points). NOTE: This variable must be encoded as a base::factor().
NUMVAR	String, the name of the column in ys containing the numbering variable for the plot (default is NULL, does not convert points into number symbols). NOTE: This variable must be encoded as a base::factor().
show.legend	Logical, Should the plot include a legend for COLVAR? (default is TRUE)
oma.right	Numeric, the size of the right outer margin for the plot, in lines of text (default is 9). Increase this value if the legend is being truncated (the image width will be automatically adjusted).
xs.add_train	Data frame, the aligned and normalized methylation data for additional samples, which will be included (after also being stratified) in the training set at every iteration of the appropriate Leave-One-Out analysis (rows are samples). To format xs.add_train correctly, use alignDatToInfo().
ys.add_train	Data frame, the aligned metadata for additional samples, which will be included (after also being stratified) in the training set at every iteration of the appropriate Leave-One-Out analysis. To format ys.add_train correctly, use alignDatToInfo().

`alpha.seq` Numeric Vector, the sequence of `alpha` values to be evaluated in `glmnetUtils::cva.glmnet()`, instead of forcing a pre-specified value of `alpha` (default is `NULL`, which skips this step and simply uses the value of the `ALPHA` argument). A general recommended choice is `alpha.seq = seq(0,1,len=11)^3`.

This feature requires the package glmnetUtils::glmnetUtils to be installed. The function `glmnetUtils::cva.glmnet()` will select the optimal value of alpha from the provided sequence using internal cross-validation, just like how lambda is selected.

WARNING: Do not change this argument unless you understand what it is; it should only be modified in the case of poor prediction when using `alpha = 0.5`. Beware of values of `alpha` close to `0`, as this allows for less sparsity, which can lead to over-fitting to the training data.

Details

While this function only returns a single data frame, it creates and saves an additional plot, and multiple lists of tables, directly to new files in the workspace.

This function performs a set of simultaneous and independent Leave-One-Out (LOO) analyses of a specified Clock model on a stratified data set. In other words, for every stratum, it performs a LOO analysis using the same Clock model. For all details related to LOO Analysis, see Details section of `saveLOOEstimation()`.

The output of this function is a data frame that copies `ys` and appends three additional new columns.

The first two appended column will be named based on the values (given by the user) of the arguments `PREDVAR` and `RESVAR`. The final appended column is `"count_probes_selected"`, which counts how many probes were selected for in the LOO Clock corresponding to that sample and that strata. These values are typically close to each other, and close to the number of probes selected in the corresponding final stratified Clock trained on the full data.

The selected sets of probes and the coefficients for every LOO Clock are not returned, but are instead saved in multiple files, named based on the file prefix given to the argument `out.rdata.PREFIX`. There is one RData file corresponding to each strata within the data. Each of these files contains a single list of tables (matrices) of coefficients, and every table of coefficients within this list contains two columns.

For all details related to building Clocks and using coefficients files, see Details section in `saveBuildClock()`.

Value

A data frame containing metadata and predicted values, created by copying `ys` and unibased appending predicted values as new columns. The names of the new columns are specified by the arguments `PREDVAR` and `RESVAR`.

---

saveLOSOEstimation *Generate Species-Level Unbiased Predictions for a New Multi-Species Clock based on a Training Data Set*

Description

Approximate the out-of-sample behavior of a multi-species Clock model on new species by generating unbiased predictions using Leave-One-Species-Out cross-validation. Generates an unbiased

version of the Clock predictions using the entire training set of a corresponding Clock (an aligned metadata ys and an aligned and normalized methylation data xs).

Files generated and saved by this function:

- List of Tables of Clock coefficients of every Leave-One-Species-Out Clock,
- Table of metadata appended with unbiased Clock predictions,
- Plot of unbiased Clock predictions.

WARNING: This function calls glmnet::cv.glmnet(), so the results of this function are subject to randomness. Users should add line in their R scripts calling set.seed() before calling this function, in order for the results to be reproducible and consistent

Usage

```
saveLOSOEstimation(
  xs,
  ys,
  OUTVAR,
  SPECVAR,
  out.rdata,
  output.csv,
  out.png,
  out.png.title,
  PREDVAR,
  RESVAR,
  ALPHA = 0.5,
  NFOLD = 10,
  fun_trans = fun_identity,
  fun_inv = fun_identity,
  fun_VAR1 = NULL,
  fun_VAR2 = NULL,
  COLVAR = NULL,
  NUMVAR = NULL,
  show.legend = T,
  oma.right = 9,
  cpg_names.list = NULL,
  xs.add_train = NULL,
  ys.add_train = NULL,
  loglambda.seq = NULL,
  alpha.seq = NULL
)
```

Arguments

xs	Data frame, the aligned and normalized methylation data for the samples used to build a Clock (rows are samples). To format xs correctly, use alignDatToInfo().
ys	Data frame, the aligned metadata for the samples used to build a Clock. To format ys correctly, use alignDatToInfo().
OUTVAR	String, the name of the column in ys containing the outcome variable (typically Age, in years)

SPECVAR	String, the name of the column in ys containing the stratifying variable (typically Species Name). NOTE: This variable must be encoded as a base::factor().
out.rdata	File name/File path String (must end in ".RData"), the name and location of the file where the list of every table of coefficients for every Leave-One-Species-Out Clock will be saved. For more information, see Details section below To skip this, use out.rdata=NULL.
output.csv	File name/File path String (must end in ".csv"), the name and location of the file where the output data frame with unbiased predicted values will be saved. For more information, see Details section below To skip this, use output.csv=NULL.
out.png	File name/File path String (must end in ".png"), the name and location of the file where the plot containing Leave-One-Species-Out predictions vs. true values will be saved To skip this, use out.png=NULL.
out.png.title	String, the title to be used in the plot saved to out.png
PREDVAR	String, the name of the NEW appended column in the output which will contain the unbiased predicted outcome variable, the output of the corresponding Leave-One-Species-Out Clock (same units as OUTVAR, and typically DNAm Age)
RESVAR	String, the name of the NEW appended column in the output which will contain the residual of the Leave-One-Species-Out predictions against the true values, roughly the difference of PREDVAR and OUTVAR (same units as OUTVAR, and typically Age Acceleration)
ALPHA	Numeric, the elasticnet mixing parameter, to be passed into glmnet::glmnet() and glmnet::cv.glmnet() (default is 0.5). WARNING: Do not change this argument unless you understand what it is; the option to change it is mostly a formality.
NFOLD	Numeric, the number of folds used for selecting lambda, to be passed into glmnet::cv.glmnet() (default is 10). WARNING: Do not change this argument unless you understand what it is; it will change computation time, at the cost of model fitting optimality.
fun_trans	Function, the name of an R function that transforms the outcome variable, with up to two additional parameters. For more information, see saveBuildClock()
fun_inv	Function, the name of an R function that is the mathematical inverse of fun_trans, with the same parameters
fun_VAR1	String, the name of the column in ys containing the first parameter of fun_trans/fun_inv, if any (default is NULL, meaning no parameter is required)
fun_VAR2	String, the name of the column in ys containing the second parameter of fun_trans/fun_inv, if any (default is NULL, meaning no parameter is required)
COLVAR	String, the name of the column in ys containing the coloring variable for the plot (default is NULL, does not color points). NOTE: This variable must be encoded as a base::factor().
NUMVAR	String, the name of the column in ys containing the numbering variable for the plot (default is NULL, does not convert points into number symbols). NOTE: This variable must be encoded as a base::factor().
show.legend	Logical, Should the plot include a legend for COLVAR? (default is TRUE)

`oma.right` Numeric, the size of the right outer margin for the plot, in lines of text (default is 9). Increase this value if the legend is being truncated (the image width will be automatically adjusted).

`cpg_names.list` List of String Vectors, the probe names used for pre-filtering for each iteration of the Leave-One-Species-Out analysis. The elements of `cpg_names.list` should match with the `levels(ys[,SPECVAR])` in order

`xs.add_train` Data frame, the aligned and normalized methylation data for additional samples, which will be included in the training set at every iteration of the Leave-One-Species-Out analysis, regardless of species (rows are samples).

To format `xs.add_train` correctly, use `alignDatToInfo()`.

`ys.add_train` Data frame, the aligned metadata for additional samples, which will be included in the training set at every iteration of the Leave-One-Species-Out analysis, regardless of species.

To format `ys.add_train` correctly, use `alignDatToInfo()`.

`loglambda.seq` Numeric Vector, the sequence of `lambda` values, on the natural log scale, to be evaluated in `glmnet::cv.glmnet()` every time, instead of the sequences automatically generated (default is `NULL`, which automatically selects 100 values, evenly spaced on the natural log scale).

WARNING: Do not change this argument unless you understand what it is; it should only be modified in the case of highly heterogeneous data, when the optimal value of lambda might be lower than the range of values automatically generated.

`alpha.seq` Numeric Vector, the sequence of `alpha` values to be evaluated in `glmnetUtils::cva.glmnet()` every time, instead of forcing a pre-specified value of `alpha` (default is `NULL`, which skips this step and simply uses the value of the `ALPHA` argument). A general recommended choice is `alpha.seq = seq(0,1,len=11)^3`.

This feature requires the package glmnetUtils::glmnetUtils to be installed. The function `glmnetUtils::cva.glmnet()` will select the optimal value of alpha from the provided sequence using internal cross-validation, just like how lambda is selected.

WARNING: Do not change this argument unless you understand what it is; it should only be modified in the case of poor prediction when using `alpha = 0.5`. Beware of values of `alpha` close to `0`, as this allows for less sparsity, which can lead to over-fitting to the training data.

Details

While this function only returns a single data frame, it creates and saves an additional plot, and a list of tables, directly to new files in the workspace.

This function performs a Leave-One-Species-Out (LOSO) analysis of a specified Clock model. For every species (level) in `ys`, it does the following:

- Creates a temporary copy of `ys` and `xs` with all samples from the selected species removed,
- Fits a Clock model based on this temporary data set copy, with the same specifications as outlined in `saveBuildClock()`,
- Saves the table of coefficients of this LOSO Clock to a list of matrices,
- Applies this LOSO Clock to all of the samples that were removed, and saves the predicted values (usually called DNAm Age LOSO) to the output of this function.

Please note that, unlike in saveLOOEstimation(), a new optimal lambda is selected (via internal cross-validation) during every iteration of LOSO. This is done because all of these optimal lambda values will likely be significantly different from each other in practice. The same is true of alpha, if alpha.seq is specified.

The output of this function is a data frame that copies ys and appends four additional new columns.

The first two appended column will be named based on the values (given by the user) of the arguments PREDVAR and RESVAR. The third appended column is "count_probes_selected", which counts how many probes were selected for in the LOSO Clock corresponding to that sample's species. These values may be quite different, depending on how different the species in the data are.

The final appended column is "log_lambda_hat", which is the natural log of the optimal lambda that is selected based on glmnet::cv.glmnet(). If you manually specify loglambda.seq and observe that many of these values are at one end of the sequence range, it is recommended that you re-specify loglambda.seq.

If alpha.seq is specified, then an extra appended column is "alpha_hat", which is the optimal alpha that is selected based on glmnetUtils::cva.glmnet(). Beware of values of alpha close to 0, as this allows for less sparsity, which can lead to over-fitting to the training data.

The selected sets of probes and the coefficients for every LOSO Clock are not returned, but are instead saved in the file given to the argument out.rdata. This file contains a list of tables (matrices) of coefficients, and every table of coefficients within this list contains two columns.

For all details related to building Clocks and using coefficients files, see Details section in saveBuildClock().

Value

A data frame containing metadata and predicted values, created by copying ys and unibased appending predicted values as new columns. The names of the new columns are specified by the arguments PREDVAR and RESVAR.

`saveMetaEWAS` *Perform Meta Epigenome-wide Association Study on Outcome Variable, by Secondary Variable*

Description

Performs a meta epigenome-wide association study (meta-EWAS) on the outcome variable, by first stratifying the data by secondary categorical variable and performing an EWAS within each stratum. The results are then aggregated via Stouffer's method to calculate a single index.

Files generated and saved by this function:

- Aggregate Table of meta-EWAS results,
- Multiple Tables of stratified EWAS results (one for each stratum),
- Plot of Clock predictions.

Usage

```
saveMetaEWAS(
  xs,
  ys,
  OUTVAR,
  STRATVAR,
  ewas.csv.PREFIX,
  ewas.png,
  fun_trans = fun_identity,
  fun_VAR1 = NULL,
  fun_VAR2 = NULL,
  ewas.png.size = NULL
)
```

Arguments

xs	Data frame, the aligned and normalized methylation data for the samples used to build a Clock (rows are samples). To format xs correctly, use alignDatToInfo().
ys	Data frame, the aligned metadata for the samples used to build a Clock. To format ys correctly, use alignDatToInfo().
OUTVAR	String, the name of the column in ys containing the outcome variable (typically Age, in years, but can be binary as well)
STRATVAR	String, the name of the column in ys containing the stratifying variable (typically Tissue Type), by which to stratify ys and calculate independent z-scores. NOTE: This variable must be encoded as a base::factor().
ewas.csv.PREFIX	File name/File path String PREFIX (meaning a file name or file path without the '.csv' at the end), the name and location of the files where the meta-EWAS output table, and each of the strata EWAS output tables, will be saved To skip this, use ewas.csv.PREFIX=NULL.
ewas.png	File name/File path String (must end in ".png"), the name and location of the file where the panel plot containing pairwise strata z-scores of the probes will be saved To skip this, use ewas.png=NULL.
fun_trans	Function, the name of an R function that transforms the outcome variable prior to calculating correlations with every probe, with up to two additional parameters. If a transformation function was used to build a corresponding Clock, then the same function should be provided here. For more information, see saveBuildClock()
fun_VAR1	String, the name of the column in ys containing the first parameter of fun_trans, if any (default is NULL, meaning no parameter is required)
fun_VAR2	String, the name of the column in ys containing the second parameter of fun_trans, if any (default is NULL, meaning no parameter is required)
ewas.png.size	Numeric, the width and height of the plot, in inches (at 600 ppi) (default is NULL, automatically calculates based on number of strata)

Details

While this function only returns a single data frame, it creates and saves an additional plot, and multiple tables, directly to a new file in the workspace.

The output of this function is a data frame that contains the results of the meta-EWAS. The set of tables that are saved as files contain the results of each individual EWAS, with a suffix added with the name of the stratum value.

The panel plot that is generated is a pairs plot of all of the strata z-score columns, where the upper-right plots are pairwise scatter plots, the lower-left "plots" display the pairwise correlations of each scatter plot, and the diagonal contains the name of the stratum value corresponding to every row and column.

For details about performing regular EWAS, see `saveEWAS()`.

Value

A data frame containing the following components:

- IDThe ID/name of the probe
- Z.(Multiple columns) The score of the probe within each stratum, calculated in `saveEWAS()`
- metaZThe aggregate score of the probe across all strata, calculated via Stouffer's method
- metaPvalueThe unadjusted p-value coresponding to `metaZ`

---

saveSurvivalAnalysis *Estimate Hazard Ratios for Age Acceleration and Other Factors*

Description

Estimate the level of association between Age Acceleration (or other covariates) on survival time, after building a new Clock or applying an built Clock. Estimates coefficients using GEE and fits a survival curve to the data using the metadata of the entire training set of a corresponding Clock (an aligned metadata `ys`).

Files generated and saved by this function:

- Table of GEE coefficients,
- Image of table of GEE coefficients,
- Plot of fitted survival curve.
- Plot of fitted survival curve combined with table of GEE coefficients.

Usage

```
saveSurvivalAnalysis(
  ys.output,
  TIMEVAR,
  STATUSVAR,
  COVARS,
  coxphtable.tsv,
  coxphtable.png,
  coxph.png,
  coxphcombined.png,
```

```
  coxph.png.title,
  STRATVAR = NULL,
  CLUSTVAR = NULL
)
```

Arguments

ys.output	Data frame, the aligned metadata for the samples, after a Clock been applied and predictions (typically DNAm Age) and residuals (typically Age Acceleration) have been generated.
TIMEVAR	String, the name of the column in ys.output containing the survival time variable (the amount of time that the individual survived after the study began)
STATUSVAR	String, the name of the column in ys.output containing the survival status indicator variable (TRUE indicates death) NOTE: This variable must be encoded as a base::logical().
COVARS	String or String Vector, the name(s) of the column(s) in ys.output containing the covariate variable(s) of interest for the survival analysis, such as Age Acceleration from a built Clock, or a treatment/control variable. To include no covariates (and use the intercept-only model), use COVARS=NULL.
coxphtable.tsv	File name/File path String (must end in ".tsv"), the name and location of the file where the GEE summary table will be saved. For more information, see tab::tabcoxph()
coxphtable.png	File name/File path String (must end in ".png"), the name and location of the file where an image of the GEE summary table (useful for presenting) will be saved To skip this, use coxphtable.png=NULL.
coxph.png	File name/File path String (must end in ".png"), the name and location of the file where the fitted survival curve plot will be saved To skip this, use coxph.png=NULL.
coxphcombined.png	File name/File path String (must end in ".png"), the name and location of the file where the fitted survival curve plot, combined with an image of the GEE summary table, will be saved To skip this, use coxphcombined.png=NULL.
coxph.png.title	String, the title to be used in all three of the plots saved
STRATVAR	String or String Vector, the name(s) of the column(s) in ys.output containing the stratifying variable(s) (default is NULL, does not stratify the survival model) NOTE: This variable must be encoded as a base::factor().
CLUSTVAR	String, the name of the column in ys.output containing the clustering variable (default is NULL, does not cluster the data) NOTE: This variable must be encoded as a base::factor().

Details

While this function returns nothing, it creates and saves a table and three plots directly to new files in the workspace.

This function performs a survival analysis of a specified covariate (or covariates) on survival time. This function is included in this package with the intended use of performing a survival analysis on the residual variable (typically Age Acceleration) generated by saveBuildClock() or saveApplyClock().

This function provides the option to stratify the survival analysis by one or more variables, or to cluster the samples according to a given variable. For more information, see `survival::coxph()`.

Please note that this function will always include the intercept term in the model.

For all details related to performing a survival analysis and fitting the regression model, see Details section in `survival::coxph()`.

Value

Nothing.

`saveValidationPanelPlot` *Save Multi-Panel Plot Image of Predicted vs. True Outcome Values with Legend*

Description

Creates and saves an image of a single multi-panel plot, showing the true values of the outcome variable (on the x-axis) against the predicted values (on the y-axis) generated by a Clock; each panel is automatically labeled and titled, and shows the samples within one group of a categorical variable (typically Tissue Type or Experimental Treatment).

Includes the correlation, the MAE (median absolute error), and the sample size in the header of every panel. Provides the option to color and/or number points according to additional variables, and include a legend for the colors.

Usage

```
saveValidationPanelPlot(
  ys.output,
  OUTVAR,
  PREDVAR,
  PANELVAR,
  out.png,
  TITLE_str,
  mfrow = NULL,
  width = NULL,
  height = NULL,
  COLVAR = PANELVAR,
  NUMVAR = NULL,
  show.original = T,
  y.axis.labs = PREDVAR,
  x.axis.labs = OUTVAR,
  panel.labs = LETTERS,
  panel.mains = NULL,
  pointsize = 12,
  show.legend = F,
  oma.right = 9,
  consistent.axes = T,
  ys.colors = NULL,
  ys.numbers = NULL
)
```

Arguments

ys.output	Data frame, the post-analysis aligned metadata for the samples, including predicted values from a Clock or LOO Analysis
OUTVAR	String, the name of the column in ys.output containing the outcome variable (typically Age, in years)
PREDVAR	String, the name of the column in ys.output containing the predicted outcome variable, the output of the Clock (same units as OUTVAR, and typically DNAm Age)
PANELVAR	String, the name of the column in ys.output containing the partitioning variable for the plot. NOTE: This variable must be encoded as a base::factor().
out.png	File name/File path String (must end in ".png"), the name and location of the file where the validation multi-panel plot containing predictions vs. true values will be saved
TITLE_str	String, the main title to be used in the plot
mfrow	2-element Numeric Vector, the dimensions of the multi-panel plot layout, given by c(nrows, ncols) (default is NULL, automatically calculates a roughly square layout, favoring rectangular wide over rectangular tall)
width	Numeric, the width of the plot, in inches (at 600 ppi) (default is NULL, automatically calculates based on number of strata). If height is NULL, this argument is treated as NULL as well.
height	Numeric, the height of the plot, in inches (at 600 ppi) (default is NULL, automatically calculates based on number of strata). If width is NULL, this argument is treated as NULL as well.
COLVAR	String OR String Vector, the name(s) of the column(s) in ys.output containing the coloring variable(s) for every panel (default is NULL, does not color points). NOTE: This variable(s) must be encoded as a base::factor().
NUMVAR	String OR String Vector, the name(s) of the column(s) in ys.output containing the numbering variable(s) for every panel (default is NULL, does not convert points into number symbols). NOTE: This variable(s) must be encoded as a base::factor().
show.original	Logical, Should the plot include the original, unpartitioned data as the first panel? (default is TRUE)
y.axis.labs	String OR String Vector, the label(s) to put on the y-axis of every panel (default is PREDVAR)
x.axis.labs	String OR String Vector, the label(s) to put on the x-axis of every panel (default is OUTVAR)
panel.labs	String Vector, the labels to put in the upper-left corner of every panel, in order to enumerate them (default is LETTERS)
panel.mains	String OR String Vector, the title(s) to put on every panel (default is NULL, uses levels(ys.output[,PANELVAR]))
pointsize	Numeric, the font point size for all points (default is 12, the same default used in plot())
show.legend	Logical, Should the plot include a legend for COLVAR? (default is FALSE)
oma.right	Numeric, the size of the right outer margin for the plot, in lines of text (default is 9). Increase this value if the legend is being truncated (the image width will be automatically adjusted).

consistent.axes
: Logical, Should the function force all the panels to have the exact same x- and y-axis limits? (default is TRUE)

ys.colors
: String Vector, the color names (or hex codes) for the colors used to color the points, according to the order in levels(ys.output[,COLVAR]) (default is NULL, automatically generates colors using Dark2() or rainbow())

ys.numbers
: String Vector, the numbers/symbols (as strings) used to replace the points, according to the order in levels(ys.output[,NUMVAR]) (default is NULL, automatically generates integer numbers using as.numeric(levels(ys.output[,NUMVAR])))

Value

Nothing.

---

saveValidationPanelPlotSeries

Save Sequential Multi-Panel Plot Images of Predicted vs. True Outcome Values without Legend

---

Description

Creates and saves multiple images of multi-panel plots in a series, showing the true values of the outcome variable (on the x-axis) against the predicted values (on the y-axis) generated by a Clock; each panel is automatically labeled and titled, and shows the samples within one group of a categorical variable (typically Tissue Type or Species). Every multi-panel plot is exactly 3x3, and the function generates as many images as are necessary to plot every group.

Includes the correlation, the MAE (median absolute error), and the sample size in the header of every panel. Provides the option to color and/or number points according to additional variables. However, it does not provide the option to include a legend for the colors.

Usage

```
saveValidationPanelPlotSeries(
  ys.output,
  OUTVAR,
  PREDVAR,
  PANELVAR,
  out.png.PREFIX,
  TITLE_str.PREFIX,
  COLVAR = NULL,
  NUMVAR = NULL,
  sort.by_sample_size = T,
  pointsize = 12,
  ys.colors = NULL,
  ys.numbers = NULL
)
```

Arguments

ys.output	Data frame, the post-analysis aligned metadata for the samples, including predicted values from a Clock or LOO Analysis
OUTVAR	String, the name of the column in ys.output containing the outcome variable (typically Age, in years)
PREDVAR	String, the name of the column in ys.output containing the predicted outcome variable, the output of the Clock (same units as OUTVAR, and typically DNAm Age)
PANELVAR	String, the name of the column in ys.output containing the partitioning variable for the plots. NOTE: This variable must be encoded as a base::factor().
out.png.PREFIX	File name/File path String PREFIX (meaning a file name or file path without the '.png' at the end), the name and location of the files where all of the validation multi-panel plots containing predictions vs. true values will be saved
TITLE_str.PREFIX	String, the main title to be used in the plot ("(part #)" will be added to the title in every plot)
COLVAR	String, the name of the column in ys.output containing the coloring variable for the plots (default is NULL, does not color points). NOTE: This variable must be encoded as a base::factor().
NUMVAR	String, the name of the column in ys.output containing the numbering variable for the plots (default is NULL, does not convert points into number symbols). NOTE: This variable must be encoded as a base::factor().
sort.by_sample_size	Logical, Should the function sort the plots and panels based on number of samples in each panel, from largest to smallest? (default is TRUE)
pointsize	Numeric, the font point size for all points (default is 12, the same default used in plot())
ys.colors	String Vector, the color names (or hex codes) for the colors used to color the points, according to the order in levels(ys.output[,COLVAR]) (default is NULL, automatically generates colors using Dark2() or rainbow())
ys.numbers	String Vector, the numbers/symbols (as strings) used to replace the points, according to the order in levels(ys.output[,NUMVAR]) (default is NULL, automatically generates integer numbers using as.numeric(levels(ys.output[,NUMVAR])))

Value

Nothing.

---

saveValidationPlot *Save Plot Image of Predicted vs. True Outcome Values with Legend*

Description

Creates and saves an image of a single plot, showing the true values of the outcome variable (on the x-axis) against the predicted values (on the y-axis) generated by a Clock.

Includes the correlation, the MAE (median absolute error), and the sample size in the header of the plot. Provides the option to color and/or number points according to additional variables, and include a legend for the colors.

Usage

```
saveValidationPlot(
  ys.output,
  OUTVAR,
  PREDVAR,
  COLVAR,
  out.png,
  TITLE_str,
  width = NULL,
  height = NULL,
  NUMVAR = NULL,
  show.legend = T,
  oma.right = 9,
  square.axes = T,
  ys.colors = NULL,
  ys.numbers = NULL
)
```

Arguments

ys.output	Data frame, the post-analysis aligned metadata for the samples, including predicted values from a Clock or LOO Analysis
OUTVAR	String, the name of the column in ys.output containing the outcome variable (typically Age, in years)
PREDVAR	String, the name of the column in ys.output containing the predicted outcome variable, the output of the Clock (same units as OUTVAR, and typically DNAm Age)
COLVAR	String, the name of the column in ys.output containing the coloring variable for the plot. NOTE: This variable must be encoded as a base::factor(). To not color points, use COLVAR=NULL.
out.png	File name/File path String (must end in ".png"), the name and location of the file where the validation plot containing predictions vs. true values will be saved
TITLE_str	String, the main title to be used in the plot
width	Numeric, the width of the plot, in inches (at 600 ppi) (default is 5). If height is NULL, this argument is treated as NULL as well.
height	Numeric, the height of the plot, in inches (at 600 ppi) (default is 6). If width is NULL, this argument is treated as NULL as well.
NUMVAR	String, the name of the column in ys.output containing the numbering variable for the plot (default is NULL, does not convert points into number symbols). NOTE: This variable must be encoded as a base::factor().
show.legend	Logical, Should the plot include a legend for COLVAR? (default is TRUE)
oma.right	Numeric, the size of the right outer margin for the plot, in lines of text (default is 9). Increase this value if the legend is being truncated (the image width will be automatically adjusted).
square.axes	Logical, Should the function force the x- and y-axis limits to be the same? (default is TRUE)

`ys.colors` String Vector, the color names (or hex codes) for the colors used to color the points, according to the order in `levels(ys.output[,COLVAR])` (default is `NULL`, automatically generates colors using `Dark2()` or `rainbow()`)

`ys.numbers` String Vector, the numbers/symbols (as strings) used to replace the points, according to the order in `levels(ys.output[,NUMVAR])` (default is `NULL`, automatically generates integer numbers using `as.numeric(levels(ys.output[,NUMVAR]))`)

Value

Nothing.

`saveValidationPlot.manyLevels` *Save Plot Image of Predicted vs. True Outcome Values with Truncated Legend*

Description

Creates and saves an image of a single plot, showing the true values of the outcome variable (on the x-axis) against the predicted values (on the y-axis) generated by a Clock.

This function is an extension of `saveValidationPlot()`, and assumes that there is a coloring variable with a large number of levels, such that the legend becomes to overbearing. In this case, it limits the legend to the top 9 largest factor levels.

Includes the median and mean within-group correlations, the MAE (median absolute error), and the sample size in the header of the plot. Provides the option to number points according to an additional variable.

Usage

```
saveValidationPlot.manyLevels(
  ys.output,
  OUTVAR,
  PREDVAR,
  COLVAR,
  out.png,
  TITLE_str,
  width = NULL,
  height = NULL,
  NUMVAR = NULL,
  show.legend = T,
  oma.right = 9,
  square.axes = T,
  ys.colors = NULL,
  ys.numbers = NULL
)
```

Arguments

ys.output	Data frame, the post-analysis aligned metadata for the samples, including predicted values from a Clock or LOO Analysis
OUTVAR	String, the name of the column in ys.output containing the outcome variable (typically Age, in years)
PREDVAR	String, the name of the column in ys.output containing the predicted outcome variable, the output of the Clock (same units as OUTVAR, and typically DNAm Age)
COLVAR	String, the name of the column in ys.output containing the coloring variable for the plot. NOTE: This variable must be encoded as a base::factor().
out.png	File name/File path String (must end in ".png"), the name and location of the file where the validation plot containing predictions vs. true values will be saved
TITLE_str	String, the main title to be used in the plot
width	Numeric, the width of the plot, in inches (at 600 ppi) (default is 5). If height is NULL, this argument is treated as NULL as well.
height	Numeric, the height of the plot, in inches (at 600 ppi) (default is 6). If width is NULL, this argument is treated as NULL as well.
NUMVAR	String, the name of the column in ys.output containing the numbering variable for the plot (default is NULL, does not convert points into number symbols). NOTE: This variable must be encoded as a base::factor().
show.legend	Logical, Should the plot include a legend for COLVAR? (default is TRUE)
oma.right	Numeric, the size of the right outer margin for the plot, in lines of text (default is 9). Increase this value if the legend is being truncated (the image width will be automatically adjusted).
square.axes	Logical, Should the function force the x- and y-axis limits to be the same? (default is TRUE)
ys.colors	String Vector, the color names (or hex codes) for the colors used to color the points, according to the order in levels(ys.output[,COLVAR]) (default is NULL, automatically generates colors using Dark2() or rainbow())
ys.numbers	String Vector, the numbers/symbols (as strings) used to replace the points, according to the order in levels(ys.output[,NUMVAR]) (default is NULL, automatically generates integer numbers using as.numeric(levels(ys.output[,NUMVAR])))

Value

Nothing.

---

selectProbes.middleFilter

Select Probes within a Methylation Dataset Based on Mean Methylation Value

---

Description

Find the CpG probes that are detectable within a methylation data set, as determined by the mean value of methylation across all samples. The "middle" filter finds all probes that have a mean value sufficient different from 0.5 (fully ambigious), specified by some threshold.

Usage

```
selectProbes.middleFilter(xs, windowsize = 0.1)
```

Arguments

xs	Data frame, the aligned and normalized methylation data for the samples (rows are samples). To format xs correctly, use alignDatToInfo().
windowsize	Numeric, the minimum difference between mean methylation and 0.5 such that a probe is not removed by the filter (default is 0.10)

Details

The mean methylation across all samples is calculated for every probe. Then, a probe is NOT remove by the filter if the mean methylation value is greater than `0.5 + windowsize` or less than `0.5 - windowsize`.

Value

Rturns a string vector containing the names of the CpG probes remaining, after applying the filtering.

---

selectProbes.rankPvalue

Select Probes within a Methylation Dataset Based on Rank P-Values

---

Description

selectProbes.rankPvalue() finds the CpG probes that are most significantly correlated (both positively and negatively) with the outcome variable in a given data set, using rank p-values across a stratifying variable.

selectProbesLOSO.rankPvalue() instead performs a LOSO variation of this process, by iterating through levels of the stratifying variable. For each iteration, it ignores one stratum of the data and then selects the same number of probes. Then, it returns a list of multiple sets of probes.

Usage

```
selectProbes.rankPvalue(
  xs,
  ys,
  OUTVAR,
  SPECVAR,
  halfsize = 2000,
  fun_trans = fun_identity,
  fun_VAR1 = NULL,
  fun_VAR2 = NULL
)

selectProbesLOSO.rankPvalue(
  xs,
```

```
  ys,
  OUTVAR,
  SPECVAR,
  halfsize = 2000,
  fun_trans = fun_identity,
  fun_VAR1 = NULL,
  fun_VAR2 = NULL
)
```

Arguments

xs	Data frame, the aligned and normalized methylation data for the samples (rows are samples). To format xs correctly, use alignDatToInfo().
ys	Data frame, the aligned metadata for the samples. To format ys correctly, use alignDatToInfo().
OUTVAR	String, the name of the column in ys containing the outcome variable (typically Age, in years)
SPECVAR	String, the name of the column in ys containing the stratifying variable (typically Species Name). NOTE: This variable must be encoded as a base::factor().
halfsize	Integer, the number of probes to be selected with most significant positive and negative correlations with the outcome variable (default is 1000). The function will return sets of probes twice the size of halfsize
fun_trans	Function, the name of an R function that transforms the outcome variable, with up to two additional parameters. For more information, see saveBuildClock()
fun_VAR1	String, the name of the column in ys containing the first parameter of fun_trans, if any (default is NULL, meaning no parameter is required)
fun_VAR2	String, the name of the column in ys containing the second parameter of fun_trans, if any (default is NULL, meaning no parameter is required)

Details

Within this documentation, S refers to S == length(levels(ys[,SPECVAR])).

In the function selectProbes.rankPvalue(), the data is divided into strata by each level in SPECVAR, and within each strata, a vector of single-probe correlations are calculated (between the transformed outcome variable and the beta values of a given probe). This results in a correlation table of each probe across S strata, which is then used to calculate rank p-values for each probe, using WGCNA::rankPvalue().

In the function selectProbesLOSO.rankPvalue(), it instead performs a LOSO variation of this process, which results in S different sets of rank p-values instead of 1. The purpose of this function is for properly doing a LOSO evaluations of a multi-species clock, if the clock was built by pre-filtering probes based on rank p-value. In this setting, instead of passing the correlation table directly into WGCNA::rankPvalue(), it generates S sub-tables, each with one column removed, and passes each of them into WGCNA::rankPvalue(). This allows the LOSO evaluation to emulate the pre-filtering process for each of the LOSO iterations independently.

Value

`selectProbes.rankPvalue()` returns a string vector of length `2*halfsize`, containing the names of the most significant CpG probes.

`selectProbesLOSO.rankPvalue()` returns a list of string vectors of length S (each string vector is of length `2*halfsize`), containing the names of the most significant CpG probes for each LOSO iteration.

---

transpose_dat *Data Frame Transpose when Column Names and First Column contain labels*

Description

Given a data frame where the first column contains labels for the rows, return the modified transpose.

Usage

```
transpose_dat(dat_tp, VAR.label)
```

Arguments

`dat_tp`	Data frame to be transposed
`VAR.label`	String, the name of the first column of the modified transpose, that will contain the colnames(dat_tp)

Details

This is a speciality function for handling unique data frame formatting patterns, where the row labels are in the first column rather than the row names.

The first column is pushed into the row names, then the data frame is transposed. Finally, the new row names are pulled out into the new first column. The new first column will be of type character.

Value

A data frame, the modified transpose of dat_tp.

Index