Package 'plasma'

March 13, 2023

Title Partial LeAst Squares for Multiomic Analysis

Version 1.0.0

Date 2023-03-10

Description Contains tools for supervised analyses of incomplete, overlapping multiomics datasets. Applies partial least squares in multiple steps to find models that predict survival outcomes. See Yamaguchi et al. (2023) <doi:10.1101/2023.03.10.532096>.

Depends R (>= 3.5.0)

Imports methods, stats, graphics, survival, pls, plsRcox, Polychrome (>= 1.5.0), viridisLite, beanplot, oompaBase

Suggests R.rsp, tidyr, ClassDiscovery

License Apache License (== 2.0)

LazyLoad yes

URL http://oompa.r-forge.r-project.org/

VignetteBuilder R.rsp

NeedsCompilation no

Author Kevin R. Coombes [cre, aut],
Kyoko Yamaguchi [aut],
Salma Abdelbaky [aut]

Maintainer Kevin R. Coombes <krc@silicovore.com>

Repository CRAN

Date/Publication 2023-03-13 14:50:02 UTC

R topics documented:

`CombinedWeights-class` *Class* `"CombinedWeights"`

Description

The `CombinedWeights` object class merges the weight matrices for all data sets in a plasma object.

Usage

```
combineAllWeights(pl)
## S4 method for signature 'CombinedWeights'
summary(object, ...)
## S4 method for signature 'CombinedWeights'
image(x, ...)
stdize(object, type = c("standard", "robust"))
interpret(object, component, alpha = 0.05)
```

Arguments

`pl`	An object of the `plasma` class.
`object`	An object of the `CombinedWeightss` class.
`x`	An object of the `CombinedWeightss` class.
`type`	A single character string indicating how to standardize the object. Legal value are "standard" or "robust".
`component`	A single chaaracter string; which componen should be interpreted.
`alpha`	A single numerical value between 0 and 1; what signfiicance value should be used to select important features.
`...`	Ignored; potentially, extra arguments to the summary or image methods.

Value

The `combineAllWeights` function returns a newly constructed object of the `CombinedWeights` class. The summary method returna list containing four matrices. Each matrix has one row for each omics data set and one column for each model component. Each amtric contains different summary statistics, including the Mean, SD, Median, and MAD.

Objects from the Class

Objects are defined using the `combineAllWeights` functions. Simply supply an object of class `plasma`.

Slots

`combined`: a matrix of the original variables in dataset `N` as rows and the PLS components `M` as columns.

`featureSize`: a numeric (usually integer) vector that stores the number of features in each omics data set.

`dataSource`: a factor indicating which omics data set each feature came from.

Methods

`summary`: outputs summary statistics for the contributions of dataset `N` to components from all datasets in the case of `getAllWeights` or dataset `M` in the case of `getCompositeWeights`.

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Kyoko Yamaguchi <kyoko.yamaguchi@osumc.edu>

Examples

```
fls <- try(loadESCAdata())
if (inherits(fls, "try-error")) {
  stop("Unable to load data from remote server.")
}
# restrict data set size
MO <- with(plasmaEnv, prepareMultiOmics(
    assemble[c("ClinicalBin", "ClinicalCont", "RPPA")], Outcome))

splitVec <- with(plasmaEnv, rbinom(nrow(Outcome), 1, 0.6))
trainD <- MO[, splitVec == 1]
testD <- MO[, splitVec == 0]

firstPass <- fitCoxModels(trainD, "Days", "vital_status", "dead")
pl <- plasma(object = trainD, multi = firstPass)

getCompositeWeights(object = pl, N = "ClinicalBin", M = "RPPA")

cbin <- getAllWeights(object = pl, N = "ClinicalBin")
summary(cbin)
image(cbin)
heat(cbin, cexCol = 0.5)

cbin01 <- pickSignificant(object = cbin, alpha = 0.01)
image(cbin01)
heat(cbin01, cexCol = 0.5)

getTop(object = cbin01, N = 3)
```

Contribution-class *Class* "Contribution"

Description

The Contribution object class contains the weight matrix between variables and the PLS components. The values in the weight matrix are a numeric representation of how much a variable from the omics datasets contributed to defining the final PLS components.

Usage

```
getCompositeWeights(object, N, M)
getAllWeights(object, N)
getFinalWeights(object)
getTop(object, N = 1)
pickSignificant(object, alpha)
## S4 method for signature 'Contribution'
summary(object, ...)
## S4 method for signature 'Contribution'
image(x, col = viridis(64), mai = c(1.82, 1.52, 0.32, 0.32), ...)
## S4 method for signature 'Contribution'
heat(object, main = "Contributions", col = viridis(64),
                                          mai = c(1.52, 0.32, 0.82, 1.82), ...)
```

Arguments

object	In the first four functions, an object of the plasma class. In the methods described here, an object of the Contributions class.
N	in the function getCompositeWeights, the name of the dataset being modeled. in the function getTop, the number of significant components you want to print.
M	name of the dataset being modeled pairwise with dataset N in the getCompositeWeights function.
alpha	level of significance used in the pickSignificant function.
...	other graphical parameters.
x	an object of the Contributions class.
main	A character vector of length one; the main plot title.
col	A vector of color descriptors.
mai	A vector of four nonnegative numbers.

Value

The plasma function returns a newly constructed object of the plasma class.

Objects from the Class

Objects are defined using the `getAllWeights`, `getCompositeWeights`, `getTop`, or `pickSignificant` functions. In the simplest scenario, one would enter an object of class `plasma` and any specific parameters associated with the function (see arguments section for more info).

Slots

`contrib`: a matrix of the original variables in dataset `N` as rows and the PLS components `M` as columns.

`datasets`: a character vector that stores the names of the datasets that were specified for the function.

Methods

`summary`: outputs summary statistics for the contributions of dataset `N` to components from all datasets in the case of `getAllWeights` or dataset `M` in the case of `getCompositeWeights`.

`image`: outputs a heatmap of the transposed `contrib` matrix.

`heat`: outputs a clustered heatmap of the `contrib` matrix.

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Kyoko Yamaguchi <kyoko.yamaguchi@osumc.edu>

Examples

```
fls <- try(loadESCAdata())
if (inherits(fls, "try-error")) {
  stop("Unable to load data from remote server.")
}
# restrict data set size
MO <- with(plasmaEnv, prepareMultiOmics(
    assemble[c("ClinicalBin", "ClinicalCont", "RPPA")], Outcome))

splitVec <- with(plasmaEnv, rbinom(nrow(Outcome), 1, 0.6))
trainD <- MO[, splitVec == 1]
testD <- MO[, splitVec == 0]

firstPass <- fitCoxModels(trainD, "Days", "vital_status", "dead")
pl <- plasma(object = trainD, multi = firstPass)

getCompositeWeights(object = pl, N = "ClinicalBin", M = "RPPA")

cbin <- getAllWeights(object = pl, N = "ClinicalBin")
summary(cbin)
image(cbin)
heat(cbin, cexCol = 0.5)

cbin01 <- pickSignificant(object = cbin, alpha = 0.01)
image(cbin01)
heat(cbin01, cexCol = 0.5)
```

```
getTop(object = cbin01, N = 3)
```

---

esca-type-data *ESCA type data*

Description

The CombinedWeights object class merges the weight matrices for all data sets in a plasma object.

Usage

```
data(tfESCA)
data(mirESCA)
```

Format

Both tfData and mirESCA are data frames containng two columns. The first column is and ID column containing the TCGA sample barcode for an esophagela cancer sample. The second column, called Type identifies the sample as either "squamous" (for likely squamous cell carcinomas that cluster near head and neck cancers) or "adeno" (for likley adenocarcinomas that cluster near stomach cancers).

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Kyoko Yamaguchi <kyoko.yamaguchi@osumc.edu>

Source

All data supplied here are based upon esophageal cancer data generated by the TCGA Research Network (https://www.cancer.gov/tcga).

The transcription factor classifications of 196 esophageal cancer into squamous cell carcinoma or adenocarcinoma are taken from work published by Abrams and colleaagues in BMC Genomics.

The microRNA classifications of 195 esophageal cancer samples into squamous cell carcinoma or adenocarcinoma are taken from work published by Asiaee and colleaagues in J Comput Biol.

References

Abrams ZB, Zucker M, Wang M, Asiaee Taheri A, Abruzzo LV, Coombes KR.
Thirty biologically interpretable clusters of transcription factors distinguish cancer type.
BMC Genomics. 2018 Oct 11;19(1):738. doi: 10.1186/s12864-018-5093-z.

Asiaee A, Abrams ZB, Nakayiza S, Sampath D, Coombes KR.
Explaining Gene Expression Using Twenty-One MicroRNAs.
J Comput Biol. 2020 Jul;27(7):1157-1170. doi: 10.1089/cmb.2019.0321.

`Imputation` *Imputation*

Description

Functions to impute missing data in omics data sets.

Usage

```
meanModeImputer(X)
samplingImputer(X)
```

Arguments

`X`	A numeric matrix, where the columns represent independent observations (patients or samples) and the columns represent measured features (genes, proteins, clinical variables, etc).

Details

We recommend imputing small amounts of missing data in the input data sets when using the `plasma` package. The underlying issue is that the PLS models we use for individual omics data sets will not be able to make predictions on a sample if even one data point is missing. As a result, if a sample is missing at least one data point in every omics data set, then it will be impossible to use that sample at all.

For a range of available imputation methods and R packages, consult the CRAN Task View on Missing Data. We also recommend the R-miss-tastic web site on missing data. Their simulations suggest that, for purposes of producing predictive models from omics data, the imputation method is not particularly important. Because of the latter finding, we have only implemented two simple imputation methods in the `plasma` package:

1. The `meanModeImputer` function will replace any missing data by the mean value of the observed data if there are more than five distinct values; otherwise, it will replace missing data by the mode. This approach works relatively well for both continuous data and for binary or small categorical data.
2. The `samplingImpute` function replaces missing values by sampling randomly from the observed data distribution.

Value

Both functions return a numeric matrix of the same size and with the same row and column names as the input variable

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Kyoko Yamaguchi <kyoko.yamaguchi@osumc.edu>

Examples

```
loadESCAdata()
imputed <- with(plasmaEnv, lapply(assemble, samplingImputer) )
imputed <- with(plasmaEnv, lapply(assemble, meanModeImputer))
```

MultiOmics-class *Class* "MultiOmics"

Description

The prepareMultiOmics function returns a new object of MultiOmics class for use in fitCoxModel.

Usage

```
prepareMultiOmics(datalist, outcome)
## S4 method for signature 'MultiOmics'
summary(object, ...)
## S4 method for signature 'MultiOmics,missing'
plot(x, y, ...)
```

Arguments

datalist	a list of dataframes formatted to have variables as rows (dimension D) and samples as columns (dimension N).
outcome	a dataframe of clinical outcomes formatted to have sample names as row indexes and variable names as column indexes
object	An object of the MultiOmics class.
x	An object of the MultiOmics class.
y	Nothing; ignored.
...	Extra graphical or other parameters.

Value

The prepareMultiOmics function returns a new object of the MultiOmics class.

Objects from the Class

Objects should be defined using the prepareMultiOmics constructor. In the simplest case, you enter two objects: a list of dataframes and a dataframe of clinical outcomes.

Slots

data: A list of dataframes with variables as rows or varying length and samples as columns of uniform length N, where N is the maximum value of non-missing samples in any given dataset. Note that NAs have been added to "pad" to make the column length uniform across data types.

outcome: A dataframe of clinical outcomes with variables as columns and samples as rows.

Methods

plot: Produces a visual representation of the dimensionalities of each dataframe in datalist. D corresponds to the number of variables in each omics dataframe, and N corresponds to samples (or members) whose variable is not entirely missing. Gray areas correspond to missing samples.

summary: Produces summary tables corresponding to datasets and outcomes.

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Kyoko Yamaguchi <kyoko.yamaguchi@osumc.edu>

Examples

```
fls <- try(loadESCAdata())
if (inherits(fls, "try-error")) {
  stop("Unable to load data from remote server.")
}
MO <- with(plasmaEnv,
           prepareMultiOmics(datalist = assemble, outcome = Outcome))
plot(MO)
summary(MO)
```

MultiplePLSCoxModels-class

Class "MultiplePLSCoxModels"

Description

The MultiplePLSCoxModels object class ... The validMultipleCoxModels function checks if each data set contains the same set of samples. The fitCoxModels function fits many plsRcox-models and returns an S4 object of class MultiplePLSCoxModels. The getSizes function returns a matrix with the list of dataframes of the MultiOmics object as rownames and columns with NT, cNT, and p-values.

Usage

```
fitCoxModels(multi, timevar, eventvar, eventvalue, verbose)
## S4 method for signature 'MultiplePLSCoxModels'
summary(object, ...)
## S4 method for signature 'MultiplePLSCoxModels,missing'
plot(x, y, col = c("blue", "red"),
      lwd = 2, xlab = "", ylab = "Fraction Surviving",
       mark.time = TRUE, legloc = "topright", ...)
## S4 method for signature 'MultiplePLSCoxModels'
predict(object, newdata, type = c("components", "risk",
                 "split", "survfit"), ...)
```

Arguments

multi	an object of class MultiOmics for fitting the model.
timevar	a column in the MultiOmics object in the outcome dataframe containing the time-to-event.
eventvar	a column in the MultiOmics object in the outcome dataframe containing the event.
eventvalue	a character string specifying the value of the event in eventvar.
verbose	logical; should the function report progress.
object	an object of class MultiplePLSCoxModels for outputting the summary.
x	an object of class MultiplePLSCoxModels for plotting the Kaplan-Meier curves.
y	An ignored argrument for the plot method.
col	A vector of color specifications. Default is c("blue", "red").
lwd	A vector specifying the line width. Default is "2".
xlab	A character string to label the x-axis. Default is "".
ylab	A character string to label the y-axis. Default is "Fraction Surviving".
mark.time	A logical value; should tickmarks indicate censored data? Default is TRUE.
legloc	A character string indicating where to put the legend. Default is "topright".
...	Other graphical parameters.
newdata	A MultiOmics object with the same structure as the training data.
type	An enumerated character value.

Value

The fitCoxModels function retuns a newly constructed object of the MultiplePLSCoxModels class. The plot method invisibly returns the object on which it was invoked. The summary method returns no value. The predict method returns a list of prediction results, each of which comes from the predict method for the SingleModel-class.

Slots

models: A list of SingleModel objects, one for each assay.

timevar: A character matching the name of the column containing the time-to-event.

eventvar: A character matching the name of the column containing the event.

eventvalue: A character specifying the event in eventvar.

Methods

plot: Plots Kaplan-Meier curves for each omics dataset split into Low Risk and High Risk groups.

summary: Returns a description of the MultiplePLSCoxModels object and the names of the omics datasets used to build the model.

predict: usually returns a list of numeric vectors of predicted risk per data type. When type = "survfit", retuns a list of survfit objects.

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Kyoko Yamaguchi <kyoko.yamaguchi@osumc.edu>

See Also

fitSingleModel

Examples

```
fls <- try(loadESCAdata())
if (inherits(fls, "try-error")) {
  stop("Unable to load data from remote server.")
}
# restrict data set size
MO <- with(plasmaEnv, prepareMultiOmics(
   assemble[c("ClinicalBin", "ClinicalCont", "RPPA")], Outcome))

splitVec <- with(plasmaEnv, rbinom(nrow(Outcome), 1, 0.6))
trainD <- MO[, splitVec == 1]
testD <- MO[, splitVec == 0]

firstPass <- fitCoxModels(trainD, "Days", "vital_status", "dead")

summary(firstPass)
plot(firstPass)
getSizes(firstPass)
pre1 <- predict(firstPass, testD)
```

---

plasma-class *Class* "plasma"

Description

The plasma object class is returned after running the plasma function. The plasma function uses the PLSRCox components from one dataset as the predictor variables and the PLSRCox components of another dataset as the response variables to fit a partial least squares regression (plsr) model. Then, we take the mean of the predictions to create a final matrix of samples versus components.

The matrix of components described earlier is then used to fit a Cox Proportional Hazards (coxph) model with AIC stepwise variable selection to return a final object of class plasma which includes a coxph model with a reduced number of predictors.

Usage

```
plasma(object, multi)
## S4 method for signature 'plasma,missing'
plot(x, y, ...)
## S4 method for signature 'plasma'
barplot(height, source, n, direction = c("both", "up","down"),
```

```
                lhcol = c("cyan", "red"), ...)
## S4 method for signature 'plasma'
predict(object, newdata = NULL, type = c("components", "risk",
                "split"), ...)
```

Arguments

multi	an object of the MultiplePLSCoxModels class.
object	an object of the plasma class.
height	an object of the plasma class for the barplot method.
x	an object of class plasma for plotting the Kaplan-Meier curves.
y	An ignored argrument for the plot method.
source	A length-one character vector; the name of a data set in a plasma object.
n	A length-one integer vector; the number of high-weight features to display.
direction	A length-one character vector; show features with positive weights (up), negative (down), or both.
lhcol	A chaacter vector of length 2, indicating the preferred colors for low (negative) or high (positive) weights.
newdata	A MultiOmics object with the same structure as the training data.
type	An enumerated character value.
...	Additional graphical parameters.

Value

The plasma function returns a newly constructed object of the plasma class. The plot method invisibly returns the object on which it was invoked. The predict method returns an object of the plasmaPredictions class.

Objects from the Class

Objects should be defined using the plasma function.

Slots

traindata: An object of class MultiOmics used for training the model.

compModels: A list containing objects in the form of plsr.

fullModel: A coxph object with variables (components) selected via AIC stepwise selection.

Methods

plot: Plots a Kaplan-Meier curve of the final coxph model that has been categorized into "low risk" and "high risk" based whether it is higher or lower, respectively, than the median value of risk.

predict: creates an object of class plasmaPredictions.

barplot: Produces a barplot of the n largest weights assigned to features from the appropriate data source.

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Kyoko Yamaguchi <kyoko.yamaguchi@osumc.edu>

See Also

plasmaPredictions, plsr

Examples

```
fls <- try(loadESCAdata())
if (inherits(fls, "try-error")) {
  stop("Unable to load data from remote server.")
}
# restrict data set size
MO <- with(plasmaEnv, prepareMultiOmics(
    assemble[c("ClinicalBin", "ClinicalCont", "RPPA")], Outcome) )

splitVec <- with(plasmaEnv, rbinom(nrow(Outcome), 1, 0.6))
trainD <- MO[, splitVec == 1]
testD <- MO[, splitVec == 0]

firstPass <- fitCoxModels(trainD, "Days", "vital_status", "dead")
pl <- plasma(object = trainD, multi = firstPass)

plot(pl, legloc = "topright", main = "Training Data")
barplot(pl, "RPPA", 6)
barplot(pl, "RPPA", 10, "up")
```

---

plasmaPredictions-class

Class "plasmaPredictions"

Description

The plasmaPredictions object class is returned when running the predict method on an object of class plasma.

Usage

```
## S4 method for signature 'plasmaPredictions,missing'
plot(x, y, col = c("blue", "red"),
                 lwd = 2, xlab = "", ylab = "Fraction Surviving",
                 mark.time = TRUE, legloc = "topright", ...)
```

Arguments

x	An object of the plasmaPredictions class for plotting the Kaplan-Meier curves.
y	An ignored argument for the plot method.
col	A vector of color specifications. Default is c("blue", "red").
lwd	A vactor specifying the line width. Default is "2".
xlab	A character string to label the x-axis. Default is "".
ylab	A character string to label the y-axis. Default is "Fraction Surviving".
mark.time	A logical value; should tickmarks indicate censored data? Default is TRUE.
legloc	A character string indicating where to put the legend. Default is "topright".
...	Other graphical parameters.

Value

The predict method on an object of the plasma class returns an object of the plasmaPredictions class. The plot method invisibly returns the value on which it was invoked.

Objects from the Class

Users shold not create objects of this class directly. They will be automatically created when you apply the predict method to a fully worked out plasma model.

Slots

meanPredictions: A matrix with samples as rows and factors as columns that is a result of taking the mean of the PLS component predictions from each dataset.

riskDF: Object of type data.frame containing the original outcome dataframe and additional columns for "Risk", and "Split", corresponding to the risk of the event calculated by the model, and patient assignment to low versus high-risk groups, respectively.

riskModel: Object of type coxph that uses predicted Risk (continuous) as the predictor variable and survival as the response variable. See documentation for link{coxph}.

splitModel: Object of type coxph that uses predicted Split (predicted Risk categorized into "high" and "low" risk by the median predicted Risk) as the predictor variable and survival as the response variable. See documentation for link{coxph}.

SF: Object of type survfit which is used by the plot method to plot Kaplan-Meier curves grouped by predicted Split. See documentation for link{survfit}.

Methods

plot: Produces Kaplan-Meier curves for the low risk and high risk groups.

Note

An object of plasmaPredictions class contains many models that are similar to an object of MultiplePLSCoxModels class.

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Kyoko Yamaguchi <kyoko.yamaguchi@osumc.edu>

See Also

plasma

Examples

```
fls <- try(loadESCAdata())
if (inherits(fls, "try-error")) {
  stop("Unable to load data from remote server.")
}
# restrict data set size
MO <- with(plasmaEnv, prepareMultiOmics(
    assemble[c("ClinicalBin", "ClinicalCont", "RPPA")], Outcome))

splitVec <- with(plasmaEnv, rbinom(nrow(Outcome), 1, 0.6))
trainD <- MO[, splitVec == 1]
testD <- MO[, splitVec == 0]

firstPass <- fitCoxModels(trainD, "Days", "vital_status", "dead")
pl <- plasma(object = trainD, multi = firstPass)

testpred <- predict(pl, testD)
plot(testpred, main = "Testing", xlab = "Time (Days)")
```

SingleModel-class *Class* "SingleModel"

Description

The `fitSingleModel` function takes in an object of `MultiOmics` class and returns a new object of `SingleModel` class.

Usage

```
fitSingleModel(multi, N, timevar, eventvar, eventvalue)
## S4 method for signature 'SingleModel'
summary(object, ...)
## S4 method for signature 'SingleModel,missing'
plot(x, y, col = c("blue", "red"),
      lwd = 2, xlab = "", ylab = "Fraction Surviving",
       mark.time = TRUE, legloc = "topright", ...)
## S4 method for signature 'SingleModel'
predict(object, newdata, type = c("components", "risk",
                 "split", "survfit"), ...)
```

Arguments

multi	an object of class MultiOmics for fitting the model.
N	A character string identifying the data set being modeled.
timevar	a column in the MultiOmics object in the outcome dataframe containing the time-to-event.
eventvar	a column in the MultiOmics object in the outcome dataframe containing the event.
eventvalue	a character string specifying the value of the event.
x	an object of class plsRcoxmodel for plotting the Kaplan-Meier curves.
y	An ignored argrument for the plot method.
col	A vector of color specifications.
lwd	A vactor specifying the line width.
xlab	A character string to label the x-axis.
ylab	A character string to label the y-axis.
mark.time	A logical value; should tickmarks indicate censored data?
legloc	A character string indicating where to put the legend.
object	an object of class SingleModel.
newdata	A MultiOmics object with the same structure as the training data.
type	An enumerated character value.
...	other parameters used in graphing or prediction.

Value

The fitSingleModel function returns a newly constructed object of the SingleModel class. The plot method invisibly returns the value on which it was invoked. The summary method returns an object summarizing the final model produced by PLS R cox regression. The predict method returns either a vector or matrix depending on the type of predictions requested.

Slots

plsmod: Object of class plsRcoxmodel containing the fitted model.

Xout: Object of type data.frame containing the original outcome dataframe and additional columns for "Risk", and "Split", corresponding to the risk of the event calculated by the model, and patient assignment to low versus high-risk groups, respectively.

dsname: A character vector of length one; the name of the data set being modeled from a MultiOmics object.

SF: Object of type survfit which is used by the plot method to plot Kaplan-Meier curves grouped by predicted Split. See documentation for link{survfit}.

riskModel: Object of type coxph that uses predicted Risk (continuous) as the predictor variable and survival as the response variable. See documentation for link{coxph}.

splitModel: Object of type coxph that uses predicted Split (predicted Risk categorized into "high" and "low" risk by the median predicted Risk) as the predictor variable and survival as the response variable. See documentation for link{coxph}.

Methods

plot: Plots Kaplan-Meier curves for each omics dataset split into Low Risk and High Risk groups.

summary: Returns a description of the `MultiplePLSCoxModels` object and the names of the omics datasets used to build the model.

predict: Usually, a numeric vector containing the predicted risk values. However, when using `type = "survfit"`, tghe return value is a `survfit` object from the`survival` package.

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Kyoko Yamaguchi <kyoko.yamaguchi@osumc.edu>

See Also

`getSizes`

Examples

```
fls <- try(loadESCAdata())
if (inherits(fls, "try-error")) {
  stop("Unable to load data from remote server.")
}
MO <- with(plasmaEnv, prepareMultiOmics(assemble, Outcome) )
MO <- MO[c("ClinicalBin", "ClinicalCont", "RPPA"),]
set.seed(98765)
splitVec <- with(plasmaEnv, rbinom(nrow(Outcome), 1, 0.6))
trainD <- MO[, splitVec == 1]
testD <- MO[, splitVec == 0]

zerothPass <- fitSingleModel(trainD, N = "RPPA",
                             timevar = "Days", eventvar = "vital_status",
                             eventvalue = "dead")
summary(zerothPass)
plot(zerothPass)
pre0 <- predict(zerothPass, testD)
```

TCGA-ESCA *Esophageal carcinoma (ESCA) data or lung squamous cell carcinoma (LUSC) data from The Cancer Genome Atlas (TCGA).*

Description

The `TCGA-ESCA` dataset contains the objects `assemble`, `Outcome`, and `m450info` for building the `MultiOmics` object. Because its size exceeds the CRAN limits, the data is stored on a remote server and must be loaded using the function `loadESCAdata`.

The `TCGA-LUSC1`dataset is a parallel object for lung squamous cell carcinoma (LUSC) data, whihc must be loaded using the `loadLUSCdata` function.

Usage

```
loadESCAdata(env = plasmaEnv)
loadLUSCdata(env = plasmaEnv)
```

Arguments

env	an environment in which to load the data. The default value is a private environment in the package, accessible as plasmaEnv. To make access easier, you can use globalenv() or .GlobalEnv.

Format

The "TCGA-ESCA" dataset contains the following:

assemble A list of 7 different omics dataframes with varying numbers of features as rows (D) and varying number of patients as columns (N). Note that some of these omics dataframes had been manipulated to contain NAs, where these may be complete on the GDC Dat Portal from which these data originally came. This was done to illustrate the capability of the plasma package on working with missing data.

1. ClinicalBina dataframe (53x185) of clinical binary values.
2. ClinicalConta dataframe (6x185) of clinical continuous values.
3. MAFa dataframe (566x184) of minor allele frequencies (MAF) that have been converted to binary based on whether they had a MAF greater than 0.03 (1) or not (0).
4. Meth450a dataframe (1454x185) of continuous beta values from the Illumina Infinium HumanMethylation450 arrays. The features in this dataframe have been filtered on mean greater than 0.15 and a standard deviation greater than 0.3.
5. miRSeqa dataframe (926x166) of continuous counts values from microRNA (miRNA) sequencing. The features in this dataframe have been filtered on a standard deviation of 0.05.
6. mRNASeqa dataframe (2520x157) of continuous counts values from mRNA sequencing data. The features in this dataframe have been filtered on a mean greater than 4 and a standard deviation greater than 0.7.
7. RPPAa dataframe (192x126) of continuous protein expression values from reverse phase protein array (RPPA) assays.

Outcome a dataframe (185x5) containing the survival outcomes for the patients in assemble.

m450info a dataframe (1454x3) containing gene symbol, chromosome number, and genomic coordinate IDs corresponding to the features (or "probes") in Meth450.

Author(s)

Kevin R. Coombes <krc@silicovore.com>, Kyoko Yamaguchi <kyoko.yamaguchi@osumc.edu>

Source

https://portal.gdc.cancer.gov/projects/TCGA-ESCA

Examples

```
fls <- try(loadESCAdata())
if (inherits(fls, "try-error")) {
  stop("Unable to load data from remote server.")
}
```

Index