Package 'PatientLevelPrediction'

April 6, 2016

Type Package

Title Package for patient level prediction using data in the OMOP Common Data Model

Version 1.1.0

Date 2015-11-4

Author Martijn J. Schuemie [aut, cre],
Marc A. Suchard [aut],
Patrick B. Ryan [aut],
Jenna Reps [aut],
Peter Rijnbeek [aut]

Maintainer Martijn J. Schuemie <schuemie@ohdsi.org>

Description A package for creating patient level prediction models. Given a cohort of interest and an outcome of interest, the package can use data in the Common Data Model to build a large set of features. These features can then be used by the Cyclops package to fit a predictive model. Also included are function for evaluating the predictive models.

License Apache License 2.0

Depends R (>= 3.2.2),
DatabaseConnector (>= 1.3.0),
Cyclops (>= 1.2.0)

Imports ggplot2,
bit,
ff,
ffbase (>= 0.12.1),
plyr,
survAUC,
Rcpp (>= 0.11.2),
RJDBC,
SqlRender (>= 1.1.3),
survival,
h2o,
FeatureExtraction

Suggests testthat,
pROC,
gnm,
knitr,
rmarkdown

LinkingTo Rcpp

NeedsCompilation yes

RoxygenNote 5.0.0

R topics documented:

bySumFf *Compute sum of values binned by a second variable*

Description

Compute sum of values binned by a second variable

Usage

```
bySumFf(values, bins)
```

Arguments

values	An ff object containing the numeric values to be summed
bins	An ff object containing the numeric values to bin by

Examples

```
values <- ff::as.ff(c(1, 1, 2, 2, 1))
bins <- ff::as.ff(c(1, 1, 1, 2, 2))
bySumFf(values, bins)
```

computeAuc *Compute the area under the ROC curve*

Description

Compute the area under the ROC curve

Usage

```
computeAuc(prediction, confidenceInterval = FALSE)
```

Arguments

prediction	A prediction object as generated using the predict functions.
confidenceInterval	Should 95 percebt confidence intervals be computed?

Details

Computes the area under the ROC curve for the predicted probabilities, given the true observed outcomes.

computeAucFromDataFrames *Compute the area under the ROC curve*

Description

Compute the area under the ROC curve

Usage

```
computeAucFromDataFrames(prediction, status, time = NULL,
  confidenceInterval = FALSE, timePoint, modelType = "logistic")
```

Arguments

prediction	A vector with the predicted hazard rate.
status	A vector with the status of 1 (event) or 0 (no event).
time	Only for survival models: a vector with the time to event or censor (which ever comes first).
confidenceInterval	Should 95 percebt confidence intervals be computed?
timePoint	Only for survival models: time point when the AUC should be evaluated
modelType	Type of model. Currently supported are "logistic" and "survival".

Details

Computes the area under the ROC curve for the predicted probabilities, given the true observed outcomes.

computeCovariateMeans *Compute covariate means*

Description

Compute covariate means

Usage

```
computeCovariateMeans(plpData, cohortId = NULL, outcomeId = NULL)
```

Arguments

plpData	An object of type plpData.
cohortId	The ID of the specific cohort for which to compute the means.
outcomeId	The ID of the specific outcome for which to compute the subgroup means.

createPlpSimulationProfile *Create simulation profile*

Description

createplpDataSimulationProfile creates a profile based on the provided plpData object, which can be used to generate simulated data that has similar characteristics.

Usage

```
createPlpSimulationProfile(plpData)
```

Arguments

plpData	An object of type plpData as generated using getDbplpData.

Details

The output of this function is an object that can be used by the simulateplpData function to generate a plpData object.

Value

An object of type plpDataSimulationProfile.

createStudyPopulation *Create a study population*

Description

Create a study population

Usage

```
createStudyPopulation(plpData, population = NULL, outcomeId,
  firstExposureOnly = FALSE, washoutPeriod = 0,
  removeSubjectsWithPriorOutcome = TRUE, priorOutcomeLookback = 99999,
  requireTimeAtRisk = T, minTimeAtRisk = 365, riskWindowStart = 0,
  addExposureDaysToStart = FALSE, riskWindowEnd = 365,
  addExposureDaysToEnd = F)
```

Arguments

plpData	An object of type plpData as generated using getDbplpData.
population	If specified, this population will be used as the starting point instead of the cohorts in the plpData object.
outcomeId	The ID of the outcome. If not specified, no outcome-specific transformations will be performed.
firstExposureOnly	Should only the first exposure per subject be included? Note that this is typically done in the createStudyPopulation function,
washoutPeriod	The mininum required continuous observation time prior to index date for a person to be included in the cohort.
removeSubjectsWithPriorOutcome	Remove subjects that have the outcome prior to the risk window start?
priorOutcomeLookback	How many days should we look back when identifying prior outcomes?
requireTimeAtRisk	Should subject without time at risk be removed?
minTimeAtRisk	The minimum number of days at risk required to be included
riskWindowStart	The start of the risk window (in days) relative to the index date (+ days of exposure if the addExposureDaysToStart parameter is specified).
addExposureDaysToStart	Add the length of exposure the start of the risk window?
riskWindowEnd	The end of the risk window (in days) relative to the index data (+ days of exposure if the addExposureDaysToEnd parameter is specified).
addExposureDaysToEnd	Add the length of exposure the risk window?

Details

Create a study population by enforcing certain inclusion and exclusion criteria, defining a risk window, and determining which outcomes fall inside the risk window.

Value

A data frame specifying the study population. This data frame will have the following columns:

rowId A unique identifier for an exposure

subjectId The person ID of the subject

cohortStartdate The index date

outcomeCount The number of outcomes observed during the risk window

timeAtRisk The number of days in the risk window

survivalTime The number of days until either the outcome or the end of the risk window

developModel *developModel - Train and evaluate the model*

Description

This provides a general framework for training patient level prediction models. The user can select various default feature selection methods or incorporate their own, The user can also select from a range of default classifiers or incorporate their own. There are three types of evaluations for the model patient (randomly splits people into train/validation sets) or year (randomly splits data into train/validation sets based on index year - older in training, newer in validation) or both (same as year spliting but checks there are no overlaps in patients within training set and validaiton set - any overlaps are removed from validation set)

Usage

```
developModel(population, plpData, featureSettings = NULL, modelSettings,
  testSplit = "time", testFraction = 0.3, nfold = 3, indexes = NULL,
  dirPath = NULL)
```

Arguments

population	The population created using createStudyPopulation() who will be used to develop the model
plpData	An object of type plpData - the patient level prediction data extracted from the CDM.
featureSettings	An object of class featureSettings created using one of the function: • filterCovariates() Filter covariate/concept/analysis ids • GLRMfeature() Non-negative matrix factorisation
modelSettings	An object of class modelSettings created using one of the function: • logisticRegressionModel() A lasso logistic regression model • GBMclassifier() A gradient boosting machine • RFclassifier() A random forest model • GLMclassifier () A generalised linear model • KNNclassifier() A KNN model
testSplit	Either 'person' or 'time' specifying the type of evaluation used. 'time' find the date where testFraction of patients had an index after the date and assigns patients with an index prior to this date into the training set and post the date into the test set 'person' splits the data into test (1-testFraction of the data) and train (validationFraction of the data) sets. The split is stratified by the class label.
testFraction	The fraction of the data to be used as the test set in the patient split evaluation.
nfold	The number of folds used in the cross validation (default 3)
indexes	A dataframe containing a rowId and index column where the index value of -1 means in the test set, and positive integer represents the cross validation fold (default is NULL)
dirFold	The path to the directory where the models will be saved

Details

Users can define a risk period of interest for the prediction of the outcome relative to index or use the cohprt dates. The user can then specify whether they wish to exclude patients who are not observed during the whole risk period, cohort period or experienced the outcome prior to the risk period.

Value

An object containing the model or location where the model is save, the data selection settings, the preprocessing and training settings as well as various performance measures obtained by the model.

`model`	A list of class `plpModel` containing the model, training metrics and model metadata
`dataSummary`	A list detailing the size of the train/test sets and outcome prevalence
`indexes`	The dataframe with the rowIds and indexes used to split the data into test/train and cross-validation folds
`type`	The type of evaluation that was performed ('person' or 'time')
`prediction`	A dataframe containing the prediction for each person in the test set
`performance`	A list detailing the performance of the model
`time`	The complete time taken to do the model framework

Examples

```
#******** EXAMPLE 1 *********
#load plpData:
plpData <- loadPlpData(file.path('C:','User','home','data'))

#create study population to develop model on
#require minimum of 365 days observation prior to at risk start
#no prior outcome and person must be observed for 365 after index (minTimeAtRisk)
#with risk window from 0 to 365 days after index
population <- createStudyPopulation(plpData,outcomeId=2042,
                                    firstExposureOnly = FALSE,
                                    washoutPeriod = 365,
                                    removeSubjectsWithPriorOutcome = TRUE,
                                    priorOutcomeLookback = 99999,
                                    requireTimeAtRisk = T,
                                    minTimeAtRisk=365,
                                    riskWindowStart = 0,
                                    addExposureDaysToStart = FALSE,
                                    riskWindowEnd = 365,
                                    addExposureDaysToEnd = F)

#lasso logistic regression predicting outcome 200 in cohorts 10
#using no feature selection with a time split evaluation with 30% in test set
#70% in train set where the model hyper-parameters are selected using 3-fold cross validation:
#and results are saved to file.path('C:','User','home')
model.lr <- logisticRegressionModel()
mod.lr <- developModel(population=population,
                       plpData= plpData,
                       featureSettings = NULL,
                       modelSettings = model.lr ,
                       testSplit = 'time', testFraction=0.3,
                       nfold=3, indexes=NULL,
```

```
                        dirPath=file.path('C:','User','home'))

#******** EXAMPLE 2 *********
# Gradient boosting machine with a grid search to select hyper parameters
# using the test/train/folds created for the lasso logistic regression above
model.gbm <- GBMclassifier(rsampRate=c(0.5,0.9,1),csampRate=1, ntrees=c(10,100), bal=c(F,T),
                           max_depth=c(4,5), learn_rate=c(0.1,0.01))
mod.gbm <- developModel(population=population,
                        plpData= plpData,
                        featureSettings = NULL,
                        modelSettings = model.gbm,
                        testSplit = 'time', testFraction=0.3,
                        nfold=3, indexes=mod.lr$indexes,
                        dirPath=file.path('C:','User','home'))
```

`evaluatePlp` *evaluatePlp*

Description

Evaluates the performance of the patient level prediction model

Usage

```
evaluatePlp(prediction)
```

Arguments

`prediction`	The patient level prediction model's prediction

Details

The function calculates various metrics to measure the performance of the model

Value

A list containing the performance values

`exportPlpDataToCsv` *Export all data in a plpData object to CSV files*

Description

Export all data in a plpData object to CSV files

Usage

```
exportPlpDataToCsv(plpData, outputFolder)
```

Arguments

plpData	An object of type plpData.
outputFolder	The folder on the file system where the CSV files will be created. If the folder does not yet exist it will be created.

Details

Created a set of CSV files in the output folder with all the data in the plplData object. This function is intended to be used for research into prediction methods. The following files will be created:

cohort.csv Listing all persons and their prediction periods. This file will have these fields: row_id (a unique ID per period), person_id, cohort_start_date, cohort_id, time (number of days in the window).

outcomes.csv Listing all outcomes per period. This file will have these fields: row_id, outcome_id, outcome_count, time_to_event.

exclude.csv Either not exported or a file listing per outcome ID which windows had the outcome prior to the window and should therefore be removed prior to fitting the model. This object will have these fields: rowId, outcomeId.

covariates.csv Listing the baseline covariates per person in the cohorts. This is done using a sparse representation: covariates with a value of 0 are omitted to save space. The covariates file will have three columns: rowId, covariateId, and covariateValue.

covariateRef.csv A file describing the covariates that have been extracted.

metaData Some information on how the plpData object was constructed.

Examples

```
## Not run:
exportPlpDataToCsv(plpData, "s:/temp/exportTest")

## End(Not run)
```

---

fitPlp *fitModel*

Description

Train various models using a default parameter gird search or user specified parameters

Usage

```
fitPlp(population, data, index, modelSettings, featureSettings, dirPath)
```

Arguments

population	The population created using createStudyPopulation() who will have their risks predicted
data	An object of type plpData - the patient level prediction data extracted from the CDM.

index	A data frame containing rowId: a vector of rowids and index: a vector of doubles the same length as the rowIds. If used, only the rowIds with a negative index value are used to calculate the prediction.
modelSettings	An object of class modelSettings created using one of the function: • logisticRegressionModel() A lasso logistic regression model • GBMclassifier() A gradient boosting machine • RFclassifier() A random forest model • GLMclassifier () A generalised linear model • KNNclassifier() A KNN model
featureSettings	An object of class featureSettings created using one of the function: • filterCovariates() Filter covariate/concept/analysis ids • GLRMfeature() Non-negative matrix factorisation
dirPath	The path to the directory where the model will be saved

Details

The user can define the machine learning model to train (regularised logistic regression, random forest, gradient boosting machine, neural network and)

Value

An object of class plpModel containing:

model	The trained prediction model
modelLoc	The path to where the model is saved (if saved)
trainAuc	The AUC obtained on the training set
trainCalibration	The calibration obtained on the training set
modelSettings	A list specifiying the model, preprocessing, outcomeId and cohortId
metaData	The model meta data
trainingTime	The time taken to train the classifier

---

fitPredictiveModel *Fit a predictive model*

Description

Fit a predictive model

Usage

```
fitPredictiveModel(population, plpData, modelType = "logistic",
  excludeCovariateIds = c(), includeCovariateIds = c(),
  prior = createPrior("laplace", useCrossValidation = TRUE),
  control = createControl(cvType = "auto", fold = 3, startingVariance = 0.01,
  tolerance = 2e-07, cvRepetitions = 1, selectorType = "byPid", noiseLevel =
  "quiet"))
```

Arguments

population	A population object generated by createStudyPopulation, potentially filtered by other functions.
plpData	An object of type plpData as generated using getDbPlpData.
modelType	The type of outcome model that will be used. Possible values are "logistic", "poisson", or "cox".
excludeCovariateIds	Exclude these covariates from the outcome model.
includeCovariateIds	Include only these covariates in the outcome model.
prior	The prior used to fit the model. See createPrior for details.
control	The control object used to control the cross-validation used to determine the hyperparameters of the prior (if applicable). See createControl for details.

GBMclassifier *Create setting for gradient boosting machine model*

Description

Create setting for gradient boosting machine model

Usage

```
GBMclassifier(rsampRate = 0.9, csampRate = 1, ntrees = c(10, 100),
  bal = F, nbins = 20, max_depth = 4, min_rows = 2, learn_rate = 0.1)
```

Arguments

rsampRate	The fraction of rows to include in each tree during training
csampRate	The fraction of features to include in each tree during training
ntrees	The number of trees to build
bal	Whether to balance the training set classes
nbins	Number of bins used in continuous variables?
max_depth	Maximum number of interactions - a large value will lead to slow model training
min_rows	The minimum number of rows required at each end node of the tree
learn_rate	The boosting learn rate

Examples

```
model.gbm <- GBMclassifier(rsampRate=c(0.5,0.9,1),csampRate=1, ntrees=c(10,100), bal=c(F,T),
                           max_depth=c(4,5), learn_rate=c(0.1,0.01))
```

getAttritionTable *Get the attrition table for a population*

Description

Get the attrition table for a population

Usage

```
getAttritionTable(object)
```

Arguments

object	Either an object of type plpData, a population object generated by functions like createStudyPopulation, or an object of type outcomeModel.

Value

A data frame specifying the number of people and exposures in the population after specific steps of filtering.

getDbPlpData *Get the patient level prediction data from the server*

Description

This function executes a large set of SQL statements against the database in OMOP CDM format to extract the data needed to perform the analysis.

Usage

```
getDbPlpData(connectionDetails, cdmDatabaseSchema,
  oracleTempSchema = cdmDatabaseSchema, cohortId, outcomeIds,
  studyStartDate = "", studyEndDate = "",
  cohortDatabaseSchema = cdmDatabaseSchema, cohortTable = "cohort",
  outcomeDatabaseSchema = cdmDatabaseSchema, outcomeTable = "cohort",
  cdmVersion = "5", excludeDrugsFromCovariates = F,
  firstExposureOnly = FALSE, washoutPeriod = 0, covariateSettings)
```

Arguments

connectionDetails	An R object of type connectionDetails created using the function createConnectionDetails in the DatabaseConnector package.
cdmDatabaseSchema	The name of the database schema that contains the OMOP CDM instance. Requires read permissions to this database. On SQL Server, this should specifiy both the database and the schema, so for example 'cdm_instance.dbo'.

`oracleTempSchema`
: For Oracle only: the name of the database schema where you want all temporary tables to be managed. Requires create/insert permissions to this database.

`cohortId`
: A unique identifier to define the at risk cohort. If cohortTable = DRUG_ERA, cohortId is a CONCEPT_ID and all descendant concepts within that CONCEPT_ID will be used to define the cohort. If cohortTable <> DRUG_ERA, cohortId is used to select the cohort_concept_id in the cohort-like table.

`outcomeIds`
: A list of cohort_definition_ids used to define outcomes.

`studyStartDate`
: A calendar date specifying the minimum date that a cohort index date can appear. Date format is 'yyyymmdd'.

`studyEndDate`
: A calendar date specifying the maximum date that a cohort index date can appear. Date format is 'yyyymmdd'. Important: the study end data is also used to truncate risk windows, meaning no outcomes beyond the study end date will be considered.

`cohortDatabaseSchema`
: The name of the database schema that is the location where the cohort data used to define the at risk cohort is available. If cohortTable = DRUG_ERA, cohortDatabaseSchema is not used by assumed to be cdmSchema. Requires read permissions to this database.

`cohortTable`
: The tablename that contains the at risk cohort. If cohortTable <> DRUG_ERA, then expectation is cohortTable has format of COHORT table: cohort_concept_id, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.

`outcomeDatabaseSchema`
: The name of the database schema that is the location where the data used to define the outcome cohorts is available. If cohortTable = CONDITION_ERA, exposureDatabaseSchema is not used by assumed to be cdmSchema. Requires read permissions to this database.

`outcomeTable`
: The tablename that contains the outcome cohorts. If outcomeTable <> CONDITION_OCCURRENCE, then expectation is outcomeTable has format of COHORT table: COHORT_DEFINITION_ID, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.

`cdmVersion`
: Define the OMOP CDM version used: currently support "4" and "5".

`excludeDrugsFromCovariates`
: Should the target and comparator drugs (and their descendant concepts) be excluded from the covariates? Note that this will work if the drugs are actualy drug concept IDs (and not cohort IDs).

`firstExposureOnly`
: Should only the first exposure per subject be included? Note that this is typically done in the `createStudyPopulation` function, but can already be done here for efficiency reasons.

`washoutPeriod`
: The mininum required continuous observation time prior to index date for a person to be included in the at risk cohort. Note that this is typically done in the `createStudyPopulation` function, but can already be done here for efficiency reasons.

`covariateSettings`
: An object of type `covariateSettings` as created using the `createCovariateSettings` function in the `FeatureExtraction` package.

Details

Based on the arguments, the at risk cohort data is retrieved, as well as outcomes occurring in these subjects. The at risk cohort can be identified using the drug_era table, or through user-defined cohorts in a cohort table either inside the CDM instance or in a separate schema. Similarly, outcomes are identified using the condition_era table or through user-defined cohorts in a cohort table either inside the CDM instance or in a separate schema. Covariates are automatically extracted from the appropriate tables within the CDM. Important: The concepts used to define the at risk cohort must not be included in the covariates, including any descendant concepts. If the `cohortId` arguments represent real concept IDs, you can set the `excludeDrugsFromCovariates` argument to TRUE and automatically the drugs and their descendants will be excluded from the covariates. However, if the `cohortId` argument does not represent concept IDs, you will need to manually add the concept_ids and descendants to the `excludedCovariateConceptIds` of the `covariateSettings` argument.

Value

Returns an object of type `plpData`, containing information on the cohorts, their outcomes, and baseline covariates. Information about multiple outcomes can be captured at once for efficiency reasons. This object is a list with the following components:

outcomes A data frame listing the outcomes per person, including the time to event, and the outcome id. Outcomes are not yet filtered based on risk window, since this is done at a later stage.

cohorts A data frame listing the persons in each cohort, listing their exposure status as well as the time to the end of the observation period and time to the end of the cohort (usually the end of the exposure era).

covariates An ffdf object listing the baseline covariates per person in the two cohorts. This is done using a sparse representation: covariates with a value of 0 are omitted to save space.

covariateRef An ffdf object describing the covariates that have been extracted.

metaData A list of objects with information on how the cohortMethodData object was constructed.

The generic `()` and `summary()` functions have been implemented for this object.

`getModelDetails` *Get the predictive model details*

Description

`getModelDetails` shows the full model, so showing the betas of all variables included in the model, along with the variable names

Usage

```
getModelDetails(predictiveModel, plpData)
```

Arguments

`predictiveModel`	An object of type `predictiveModel` as generated using he `fitPredictiveModel` function.
`plpData`	An object of type `plpData` as generated using `getDbPlpData`.

Details

Shows the coefficients and names of the covariates with non-zero coefficients.

`GLMclassifier` *Create setting for generalised linear model with elastic-new regularisation*

Description

Create setting for generalised linear model with elastic-new regularisation

Usage

```
GLMclassifier(alpha = 0.5, lambda = 1e-06, lambda_search = T,
  nlambdas = 100, lambda_min_ratio = 1/10000)
```

Arguments

`alpha`	...
`lambda`	...
`lambda_search`	...
`nlambdas`	...
`lambda_min_ratio`	...

Examples

```
model.glm <- GLMclassifier(alpha=c(0.5,0.1,0.9))
```

`grepCovariateNames` *Extract covariate names*

Description

Extracts covariate names using a regular-expression.

Usage

```
grepCovariateNames(pattern, object)
```

Arguments

`pattern`	A regular expression with which to name covariate names
`object`	An R object of type `plpData` or `covariateData`.

Details

This function extracts covariate names that match a regular-expression for a `plpData` or `covariateData` object.

Value

Returns a data.frame containing information about covariates that match a regular expression. This data.frame has the following columns:

covariateId Numerical identifier for use in model fitting using these covariates

covariateName Text identifier

analysisId Analysis identifier

conceptId OMOP common data model concept identifier, or 0

---

insertDbPopulation *Insert a population into a database*

Description

Insert a population into a database

Usage

```
insertDbPopulation(population, cohortIds = 1, connectionDetails,
  cohortDatabaseSchema, cohortTable = "cohort", createTable = FALSE,
  dropTableIfExists = TRUE, cdmVersion = "5")
```

Arguments

population	Either an object of type plpData or a population object generated by functions like createStudyPopulation.
cohortIds	The IDs to be used for the treated and comparator cohort, respectively.
connectionDetails	An R object of type connectionDetails created using the function createConnectionDetails in the DatabaseConnector package.
cohortDatabaseSchema	The name of the database schema where the data will be written. Requires write permissions to this database. On SQL Server, this should specifiy both the database and the schema, so for example 'cdm_instance.dbo'.
cohortTable	The name of the table in the database schema where the data will be written.
createTable	Should a new table be created? If not, the data will be inserted into an existing table.
dropTableIfExists	If createTable = TRUE and the table already exists it will be overwritten.
cdmVersion	Define the OMOP CDM version used: currently support "4" and "5".

Details

Inserts a population table into a database. The table in the database will have the same structure as the 'cohort' table in the Common Data Model.

KNNclassifier *Create setting for knn model*

Description

Create setting for knn model

Usage

```
KNNclassifier(k = 1000, indexFolder = getwd())
```

Arguments

k	The number of neighbors to consider
indexFolder	The directory where the results and intermediate steps are output

Examples

```
model.knn <- KNNclassifier(k=c(3,100,1000))
```

loadPlpData *Load the cohort data from a folder*

Description

loadPlpData loads an object of type plpData from a folder in the file system.

Usage

```
loadPlpData(file, readOnly = TRUE)
```

Arguments

file	The name of the folder containing the data.
readOnly	If true, the data is opened read only.

Details

The data will be written to a set of files in the folder specified by the user.

Value

An object of class plpData.

Examples

```
# todo
```

`logisticRegressionModel` *Create setting for lasso logistic regression*

Description

Create setting for lasso logistic regression

Usage

```
logisticRegressionModel(variance = 0.01)
```

Arguments

`variance`	a single value or vector of values to be used to train multiple models and the model with the best performance on the cross validation set is choosen

Examples

```
model.lr <- logisticRegressionModel()
```

`makeRandomString` *makeRandomString*

Description

A function for making a random string

Usage

```
makeRandomString(n = 1, lenght = 12)
```

Arguments

`n`	An integer - the number of random string to generate
`length`	An integer - the number of characters for each string

Details

The function creates n random strings of size length

Value

A list containing n random strings with the number of characters specified by the use input length

PatientLevelPrediction *PatientLevelPrediction*

Description

PatientLevelPrediction

personSplitter *Split data into random subsets stratified by class*

Description

Split data into random subsets stratified by class

Usage

```
personSplitter(population, test = 0.3, nfold = 3, silent = F)
```

Arguments

population	An object created using createStudyPopulation().
test	A real number between 0 and 1 indicating the test set fraction of the data
nfold	An integer >= 1 specifying the number of folds used in cross validation
silent	Whether to turn off the progress reporting

Details

Returns a dataframe of rowIds and indexes with a -1 index indicating the rowId belongs to the test set and a positive integer index value indicating the rowId's cross valiation fold within the train set.

Value

A dataframe containing the columns: rowId and index

plotCalibration *Plot the calibration*

Description

Plot the calibration

Usage

```
plotCalibration(prediction, numberOfStrata = 5, truncateFraction = 0.01,
  fileName = NULL)
```

Arguments

prediction	A prediction object as generated using the predict functions.
numberOfStrata	The number of strata in the plot.
truncateFraction	This fraction of probability values will be ignored when plotting, to avoid the x-axis scale being dominated by a few outliers.
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a plot showing the predicted probabilities and the observed fractions. Predictions are stratified into equally sized bins of predicted probabilities.

Value

A ggplot object. Use the ggsave function to save to file in a different format.

plotCovariateDifferenceOfTopVariables *Plot variables with largest standardized difference*

Description

Create a plot showing those variables having the largest standardized difference between the group having the outcome and the group that doesn't have the outcome. Requires running computeCovariateMeans first.

Usage

```
plotCovariateDifferenceOfTopVariables(means, n = 20, maxNameWidth = 100,
  fileName = NULL)
```

Arguments

means	A data frame created by the computeCovariateMeans funcion.
n	Count of variates to plot.
maxNameWidth	Covariate names longer than this number of characters are truncated to create a nicer plot.
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Value

A ggplot object. Use the ggsave function to save to file in a different format.

---

plotRoc *Plot the ROC curve*

Description

Plot the ROC curve

Usage

```
plotRoc(prediction, fileName = NULL)
```

Arguments

prediction	A prediction object as generated using the predictProbabilities function.
fileName	Name of the file where the plot should be saved, for example 'plot.png'. See the function ggsave in the ggplot2 package for supported file formats.

Details

Create a plot showing the Receiver Operator Characteristics (ROC) curve.

Value

A ggplot object. Use the ggsave function to save to file in a different format.

---

plpDataSimulationProfile *A simulation profile*

Description

A simulation profile

Usage

```
data(plpDataSimulationProfile)
```

predictFfdf *Generated predictions from a regression model*

Description

Generated predictions from a regression model

Usage

```
predictFfdf(coefficients, population, covariates, modelType = "logistic")
```

Arguments

coefficients	A names numeric vector where the names are the covariateIds, except for the first value which is expected to be the intercept.
covariates	A data frame or ffdf object containing the covariates with predefined columns (see below).
modelType	Current supported types are "logistic", "poisson", or "survival".
outcomes	A data frame or ffdf object containing the outcomes with predefined columns (see below).

Details

These columns are expected in the outcome object:

rowId	(integer)	Row ID is used to link multiple covariates (x) to a single outcome (y)
time	(real)	For models that use time (e.g. Poisson or Cox regression) this contains time (e.g. number of days)

These columns are expected in the covariates object:

rowId	(integer)	Row ID is used to link multiple covariates (x) to a single outcome (y)
covariateId	(integer)	A numeric identifier of a covariate
covariateValue	(real)	The value of the specified covariate

predictPlp *predictPlp*

Description

Predict the risk of the outcome using the input plpModel for the input plpData

Usage

```
predictPlp(plpModel, population, plpData, dirPath, index = NULL)
```

Arguments

plpModel	An object of type plpModel - a patient level prediction model
population	The population created using createStudyPopulation() who will have their risks predicted
plpData	An object of type plpData - the patient level prediction data extracted from the CDM.
dirPath	The location of the output directory. If using a h2o model, the libSvm will be saved here
index	A data frame containing rowId: a vector of rowids and index: a vector of doubles the same length as the rowIds. If used, only the rowIds with a negative index value are used to calculate the prediction.

Details

The function applied the trained model on the plpData to make predictions

Value

A dataframe containing the prediction for each person in the population with an attribute metaData containing prediction details.

---

predictProbabilities *Create predictive probabilities*

Description

Create predictive probabilities

Usage

```
predictProbabilities(predictiveModel, population, covariates)
```

Arguments

predictiveModel	An object of type predictiveModel as generated using fitPredictiveModel.
plpData	An object of type plpData as generated using getDbPlpData.

Details

Generates predictions for the population specified in plpData given the model.

Value

The value column in the result data.frame is: logistic: probabilities of the outcome, poisson: Poisson rate (per day) of the outome, survival: hazard rate (per day) of the outcome.

RFclassifier *Create setting for random forest model*

Description

Create setting for random forest model

Usage

```
RFclassifier(mtries = -1, ntrees = c(10, 500), rsampRate = 0.9,
  csampRate = 1, bal = F, nbins = 20, max_depth = 17, min_rows = 2)
```

Arguments

mtries	The number of features to include in each tree (-1 defaults to square root of total features)
ntrees	The number of trees to build
rsampRate	The fraction of rows to include in each tree during training
csampRate	The fraction of features to include in each tree during training
bal	Whether to balance the training set classes
nbins	Number of bins used in continuous variables?
max_depth	Maximum number of interactions - a large value will lead to slow model training
min_rows	The minimum number of rows required at each end node of the tree

Examples

```
model.rf <- RFclassifier(mtries=c(-1,5,20), rsampRate=c(0.5,0.9,1),csampRate=1, ntrees=c(10,100), bal=c(F,T)
                         max_depth=c(5,20))
```

saveLibSVM *Save the plpData in the sparse libSVM format*

Description

Converts the plpData to libSVM format and saves the data in the user specified directory

Usage

```
saveLibSVM(population, plpData, filePath, mapping = NULL, silent = F)
```

Arguments

population	The population created using createStudyPopulation() who will be used to develop the model
plpData	An object of type plpData - the patient level prediction data extracted from the CDM.
filePath	The path to the directory to output the files
silent	Whether to turn off progress reporting
mappings	An ffdf containing originalCovariateId (old) and covariateId (new) columns specifying the old to new mapping

Details

Given the plpData and a directory the libSVM format data will be saved into the directory. The file plpData.txt contains the plpData in libSVM format, file covRef.txt is the covariate reference dataframe, the file rowId.txt is a vector of the rowIds of each row in the plpData file. The plpData.txt can be loaded into h2o, oython or spark for running efficient machine learning techniques

Examples

```
# To convert plpData into libSVM and save results to C:\plpData
```

---

savePlpData *Save the cohort data to folder*

Description

savePlpData saves an object of type plpData to folder.

Usage

```
savePlpData(plpData, file)
```

Arguments

plpData	An object of type plpData as generated using getDbPlpData.
file	The name of the folder where the data will be written. The folder should not yet exist.

Details

The data will be written to a set of files in the folder specified by the user.

Examples

```
# todo
```

---

simulatePlpData *Generate simulated data*

Description

simulateplpData creates a plpData object with simulated data.

Usage

```
simulatePlpData(plpDataSimulationProfile, n = 10000)
```

Arguments

plpDataSimulationProfile	An object of type plpDataSimulationProfile as generated using the createplpDataSimulationProfile function.
n	The size of the population to be generated.

Details

This function generates simulated data that is in many ways similar to the original data on which the simulation profile is based. The contains same outcome, comparator, and outcome concept IDs, and the covariates and their 1st order statistics should be comparable.

Value

An object of type plpData.

---

timeSplitter *Split test/train data by time and then partitions training set into random folds stratified by class*

Description

Split test/train data by time and then partitions training set into random folds stratified by class

Usage

```
timeSplitter(population, test = 0.3, nfold = 3, silent = F)
```

Arguments

population	An object created using createStudyPopulation().
test	A real number between 0 and 1 indicating the test set fraction of the data
nfold	An integer >= 1 specifying the number of folds used in cross validation
silent	Whether to turn off the progress reporting

Details

Returns a dataframe of rowIds and indexes with a -1 index indicating the rowId belongs to the test set and a positive integer index value indicating the rowId's cross valiation fold within the train set.

Value

A dataframe containing the columns: rowId and index

Index