Package ‘PatientLevelPrediction’

February 29, 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,
Peter Rijnbeek

Maintainer Jenna Reps <notareal@email.com>

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,
reshape2,
gridExtra,
genalg,
httr,
RJSONIO,
caret,
h2o,

curl,
rJava,
pbkrtest

Suggests testthat,
pROC,
gnm,
knitr,
rmarkdown,
OhdsiRTools

LinkingTo Rcpp

NeedsCompilation yes

RoxygenNote 5.0.1

R topics documented:

`byMaxFf` *Compute max of values binned by a second variable*

Description

Compute max of values binned by a second variable

Usage

```
byMaxFf(values, bins)
```

Arguments

`values`	An ff object containing the numeric values to take the max of.
`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))
byMaxFf(values, bins)
```

`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)
```

`censorPlpData` *Filters the plpData based on user specified criteria*

Description

Filters the data based on user censoring specifications for classification or survival analysis

Usage

```
censorPlpData(plpData, outcomeIds = NULL, outcomeTime = NULL,
  newOutcome = NULL, predictionPeriod = NULL, dateInterval = NULL,
  minPriorObservation = 365, minCohortTime = NULL,
  excludeOutcomeOccurrence = list(`1` = c("inf", 0)),
  classificationCensor = list(insufficientCohortObservation = c("include",
  "include"), insufficientPredictionPeriod = c("include", "include"),
  minPostObservation = NULL, insufficientPostObservation = c("include",
  "include")), survivalCensor = list(useCohortObservation = F,
  usePredictionPeriod = T, maxPostObservation = NULL, useMaxPostObservation =
  F))
```

Arguments

`plpData`	An object of type `plpData` - the patient level prediction data extracted from the CDM.
`outcomeIds`	a vector of integers (corresponding to outcome ids) or NULL
`outcomeTime`	An integer - if you need to edit the time from index where you want to predict the outcome use this parameter. For example, if you created the outcome table by finding the occurrence of the outcome 30 days after cohort start but you wish to conduct a sensitivity test for this and reduce it to 20 days, then set outcome-Time=20.
`newOutcome`	A vector of existing outcomeIds - constructs a new outcomeId based on people having all the specified outcomeIds. For example, you may wish to predict the subset of the people who have the outcome who also get given a secific treatment. If you create the outcome with outcomeId 1 and the treatment with outcomeId 2, the set newOutcome =c(1,2) to find all the people who have out-comeIds 1 and 2, they are then assigned a new outcomeId -1.

`predictionPeriod`	A vector of length 2 with the first value corresponding to the number of days after index to define the start of the risk prediction period and the second value corresponding to the number of days after index defining the end of the risk prediction period. If this is NULL the the cohort start and end date will define the risk prediction period.
`dateInterval`	(a vector of 2 dates or NULL) corresponding to the inclusion dates. For example, if the user inputs c(1990-01-01,2000-01-01) then any people with an index prior to 1990 or after (Jan 1st 2000 minus minPriorObservaton) will be excluded. If NULL all dates are included.
`minPriorObservation`	an integer - people with the time in days between their observation start and cohort start less than this number will be removed from the data.
`minCohortTime`	an integer - people with the time in days between the cohort start and cohort end less than this number will be removed from the data.
`excludeOutcomeOccurrence`	A list containing named list members and vectors of two integer values, where the name corresponds to the outcomeId and the integer vector corresponds to an interval in days for filtering people who had the outcomeId recorded during this interval. For example the list: list('1'=c(180,40), '4'=c('inf',0)) would find all the people who had the outcomeId 1 recorded 180 days prior to index up to 40 days after index and filter these people from the data, it would also find the people who have outcomeId 4 anytime prior to index and filter these people from the data.
`classificationCensor`	A list detailing the exclusion criteria for classification. The list contains: • insufficientCohortObservation - a character vector of length 2 with each element either 'include' or 'exclude' - indicating whether to include or exclude patients whose observation period ends before their cohort end. The first element is applied to people with the outcome (class 1) and the second element is applied to people without the outcome (class 0) • insufficientPredictionPeriod - a character vector of length 2 with each element either 'include' or 'exclude' - indicating whether to include or exclude patients whose predictionPeriod falls outside of their observation period. The first element is applied to people with the outcome (class 1) and the second element is applied to people without the outcome (class 0) • minPostObservation - An integer - specifying the required minimum number of days after index (Used by insufficientPostObservation) • insufficientPostObservation - a character vector of length 2 with each element either 'include' or 'exclude' - indicating whether to include or exclude patients with an index date plus minPostObservation greater than their observationEndDate. The first element is applied to people with the outcome (class 1) and the second element is applied to people without the outcome (class 0)
`survivalCensor`	A list containing the criteria for censoring the data...

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 of type `plpData` containing information on the prediction problem that only contains the data satisfying the user's specified censoring options. This object will contain the following data:

`cohorts`	An ffdf object listing all persons and their prediction periods. This object 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`	An ffdf object listing all outcomes per period. This object will have these fields: row_id, outcome_id, outcome_count, time_to_event.
`exclude`	Either NULL or an ffdf object listing per outcome ID which windows had the outcome prior to the window. This object will have these fields: rowId, outcomeId.
`covariates`	An ffdf object 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 object will have three columns: rowId, covariateId, and covariateValue.
`covariateRef`	An ffdf object describing the covariates that have been extracted.
`metaData`	A list of objects with information on how the plpData object was constructed and censoring details. The list member named 'excluded' contains a ffdf of the excluded people and reason for exclusion.

Examples

```
 {
# Filter any patients with an index before 2008-01-01 or after 2011-01-01
# and who have less than 365 days observation prior to index
plpData.censor <- censorPlpData(plpData, minPriorObservation = 365,
dateInterval = c('2008-01-01','2011-01-01'))

# Filter patients with less than 100 days observtion prior to index
# also filter all people who are not observed for
# at least 100 days post index
plpData.censor <- censorPlpData(plpData, minPriorObservation= 100,
minCohortTime=NULL,
classificationCensor=list( minPostObservation=100,
                         insufficientPostObservation = c('exclude','exclude')
                         )
                         )

# Filter patients with less than 100 days observtion prior to index
# also filter people who do not have the outcome who are not observed for
# at least 100 days post index
plpData.censor censorPlpData(plpData, minPriorObservation= 100,
minCohortTime=NULL,
classificationCensor=list(minPostObservation=100,
                         insufficientPostObservation = c('include','exclude')
                         )
                         )
# Filter people with an outcomeId 2 that occurs within 180 days before index
# until 5 days after index, also filter people with less than 365 days
# observation prior to index and without a minimum of 365 days after index
plpData.censor censorPlpData(plpData, minPriorObservation= 365,
excludeOutcomeOccurrence=list('2'=c(180,5)),
classificationCensor=list(minPostObservation=365,
                         insufficientPostObservation = c('exclude','exclude')
                         )
                         )
```

`comparePlp` *function comparePlp*

Description

Compares the performance of two or more patient level prediction models

Usage

```
comparePlp(models)
```

Arguments

`models`	A list of plp models

Details

The function summarises and plots the performance of the input models for comparison

Value

A table summarising the performance value comparision and plots.

Examples

```
modset_llr  <- list(model='lr_lasso',
                   param=list(variance =0.001, cohortId=c(1,2), outcomeId=2))
class(modset_llr) <- 'modelSettings'
model1 <- developModel2(plpData= plpData,
                        featureSettings = NULL,
                        modelSettings = modset_llr ,
                        type='year')

featSet_gbm <- list(method='wrapperGA', param=list(cohortId=c(1,2), outcomeId=2, varSize=300, iter=25))
class(featSet_gbm) <- 'featureSettings'
modset_gbm <- list(model='gbm_plp',
               param=list(rsampRate=0.8, ntrees=c(100,150), max_depth=c(2,4,5), cohortId=c(1,2), outcomeId=
class(modset_gbm) <- 'modelSettings'
model2 <- developModel2(plpData= plpData.censor,
                         featureSettings = featSet_gbm,
                         modelSettings = modset_gbm,
                         type='year')

model3 <- developModel2(plpData= plpData.censor,
                         featureSettings = NULL,
                         modelSettings = modset_gbm,
                         type='year')

allModels <- list(model1[[1]], model2[[1]], model3[[1]])
```

```
comparePlp(allModels)
```

---

computeAuc — *Compute the area under the ROC curve*

Description

Compute the area under the ROC curve

Usage

```
computeAuc(prediction, plpData, removeDropoutsForLr = TRUE,
  confidenceInterval = FALSE)
```

Arguments

`prediction`	A prediction object as generated using the `predictProbabilities` function.
`plpData`	An object of type `plpData`.
`removeDropoutsForLr`	If TRUE and modelType is "logistic", subjects that do not have the full observation window (i.e. are censored earlier) and do not have the outcome are removed prior to evaluating the model.
`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.

`createConceptCovariateSettings` *Create Concept covariate settings*

Description

Create Concept covariate settings

Usage

```
createConceptCovariateSettings(conceptList, useDemo = TRUE)
```

Arguments

`conceptList`	A list of lists - each inner list contains two objects: conceptSet a vector of conceptSetIds and prior an integer specifying the number of days prior to index to search for the concepts in the set

Details

creates an object specifying how covariates should be contructed from data in the CDM model.

Value

An object of type `conceptCovariateSettings`, to be used in other functions.

`createCovariateSettings` *Create covariate settings*

Description

Create covariate settings

Usage

```
createCovariateSettings(useCovariateCohortIdIs1 = FALSE,
  useCovariateDemographics = TRUE, useCovariateDemographicsGender = TRUE,
  useCovariateDemographicsRace = TRUE,
  useCovariateDemographicsEthnicity = TRUE,
  useCovariateDemographicsAge = TRUE, useCovariateDemographicsYear = TRUE,
  useCovariateDemographicsMonth = TRUE,
  useCovariateConditionOccurrence = TRUE,
  useCovariateConditionOccurrence365d = TRUE,
  useCovariateConditionOccurrence30d = FALSE,
  useCovariateConditionOccurrenceInpt180d = FALSE,
  useCovariateConditionEra = FALSE, useCovariateConditionEraEver = FALSE,
  useCovariateConditionEraOverlap = FALSE,
  useCovariateConditionGroup = FALSE,
  useCovariateConditionGroupMeddra = FALSE,
  useCovariateConditionGroupSnomed = FALSE,
  useCovariateDrugExposure = FALSE, useCovariateDrugExposure365d = FALSE,
  useCovariateDrugExposure30d = FALSE, useCovariateDrugEra = FALSE,
  useCovariateDrugEra365d = FALSE, useCovariateDrugEra30d = FALSE,
  useCovariateDrugEraOverlap = FALSE, useCovariateDrugEraEver = FALSE,
  useCovariateDrugGroup = FALSE, useCovariateProcedureOccurrence = FALSE,
  useCovariateProcedureOccurrence365d = FALSE,
  useCovariateProcedureOccurrence30d = FALSE,
  useCovariateProcedureGroup = FALSE, useCovariateObservation = FALSE,
  useCovariateObservation365d = FALSE, useCovariateObservation30d = FALSE,
  useCovariateObservationCount365d = FALSE, useCovariateMeasurement = FALSE,
  useCovariateMeasurement365d = FALSE, useCovariateMeasurement30d = FALSE,
  useCovariateMeasurementCount365d = FALSE,
  useCovariateMeasurementBelow = FALSE,
  useCovariateMeasurementAbove = FALSE, useCovariateConceptCounts = FALSE,
  useCovariateRiskScores = FALSE, useCovariateRiskScoresCharlson = FALSE,
  useCovariateRiskScoresDCSI = FALSE, useCovariateRiskScoresCHADS2 = FALSE,
  useCovariateRiskScoresCHADS2VASc = FALSE,
  useCovariateInteractionYear = FALSE, useCovariateInteractionMonth = FALSE,
  excludedCovariateConceptIds = c(), includedCovariateConceptIds = c(),
  deleteCovariatesSmallCount = 100)
```

Arguments

`useCovariateCohortIdIs1`	A boolean value (TRUE/FALSE) to determine if a covariate should be contructed for whether the cohort ID is 1 (currently primarily used in CohortMethod).
`useCovariateDemographics`	A boolean value (TRUE/FALSE) to determine if demographic covariates (age in 5-yr increments, gender, race, ethnicity, year of index date, month of index date) will be created and included in future models.
`useCovariateDemographicsGender`	A boolean value (TRUE/FALSE) to determine if gender should be included in the model.
`useCovariateDemographicsRace`	A boolean value (TRUE/FALSE) to determine if race should be included in the model.
`useCovariateDemographicsEthnicity`	A boolean value (TRUE/FALSE) to determine if ethnicity should be included in the model.
`useCovariateDemographicsAge`	A boolean value (TRUE/FALSE) to determine if age (in 5 year increments) should be included in the model.
`useCovariateDemographicsYear`	A boolean value (TRUE/FALSE) to determine if calendar year should be included in the model.
`useCovariateDemographicsMonth`	A boolean value (TRUE/FALSE) to determine if calendar month should be included in the model.
`useCovariateConditionOccurrence`	A boolean value (TRUE/FALSE) to determine if covariates derived from CONDITION_OCCURRENCE table will be created and included in future models.
`useCovariateConditionOccurrence365d`	A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of condition in 365d window prior to or on cohort index date. Only applicable if useCovariateConditionOccurrence = TRUE.
`useCovariateConditionOccurrence30d`	A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of condition in 30d window prior to or on cohort index date. Only applicable if useCovariateConditionOccurrence = TRUE.
`useCovariateConditionOccurrenceInpt180d`	A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of condition within inpatient type in 180d window prior to or on cohort index date. Only applicable if useCovariateConditionOccurrence = TRUE.
`useCovariateConditionEra`	A boolean value (TRUE/FALSE) to determine if covariates derived from CONDITION_ERA table will be created and included in future models.
`useCovariateConditionEraEver`	A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of condition era anytime prior to or on cohort index date. Only applicable if useCovariateConditionEra = TRUE.

`useCovariateConditionEraOverlap`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of condition era that overlaps the cohort index date. Only applicable if useCovariateConditionEra = TRUE.

`useCovariateConditionGroup`
: A boolean value (TRUE/FALSE) to determine if all CONDITION_OCCURRENCE and CONDITION_ERA covariates should be aggregated or rolled-up to higher-level concepts based on vocabluary classification.

`useCovariateConditionGroupMeddra`
: A boolean value (TRUE/FALSE) to determine if all CONDITION_OCCURRENCE and CONDITION_ERA covariates should be aggregated or rolled-up to higher-level concepts based on the MEDDRA classification.

`useCovariateConditionGroupSnomed`
: A boolean value (TRUE/FALSE) to determine if all CONDITION_OCCURRENCE and CONDITION_ERA covariates should be aggregated or rolled-up to higher-level concepts based on the SNOMED classification.

`useCovariateDrugExposure`
: A boolean value (TRUE/FALSE) to determine if covariates derived from DRUG_EXPOSURE table will be created and included in future models.

`useCovariateDrugExposure365d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of drug in 365d window prior to or on cohort index date. Only applicable if useCovariateDrugExposure = TRUE.

`useCovariateDrugExposure30d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of drug in 30d window prior to or on cohort index date. Only applicable if useCovariateDrugExposure = TRUE.

`useCovariateDrugEra`
: A boolean value (TRUE/FALSE) to determine if covariates derived from DRUG_ERA table will be created and included in future models.

`useCovariateDrugEra365d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of drug era in 365d window prior to or on cohort index date. Only applicable if useCovariateDrugEra = TRUE.

`useCovariateDrugEra30d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of drug era in 30d window prior to or on cohort index date. Only applicable if useCovariateDrugEra = TRUE.

`useCovariateDrugEraOverlap`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of drug era that overlaps the cohort index date. Only applicable if useCovariateDrugEra = TRUE.

`useCovariateDrugEraEver`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of drug era anytime prior to or on cohort index date. Only applicable if useCovariateDrugEra = TRUE.

`useCovariateDrugGroup`
: A boolean value (TRUE/FALSE) to determine if all DRUG_EXPOSURE and DRUG_ERA covariates should be aggregated or rolled-up to higher-level concepts of drug classes based on vocabluary classification.

`useCovariateProcedureOccurrence`
: A boolean value (TRUE/FALSE) to determine if covariates derived from PROCEDURE_OCCURRENCE table will be created and included in future models.

`useCovariateProcedureOccurrence365d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of procedure in 365d window prior to or on cohort index date. Only applicable if useCovariateProcedureOccurrence = TRUE.

`useCovariateProcedureOccurrence30d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of procedure in 30d window prior to or on cohort index date. Only applicable if useCovariateProcedureOccurrence = TRUE.

`useCovariateProcedureGroup`
: A boolean value (TRUE/FALSE) to determine if all PROCEDURE_OCCURRENCE covariates should be aggregated or rolled-up to higher-level concepts based on vocabluary classification.

`useCovariateObservation`
: A boolean value (TRUE/FALSE) to determine if covariates derived from OBSERVATION table will be created and included in future models.

`useCovariateObservation365d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of observation in 365d window prior to or on cohort index date. Only applicable if useCovariateObservation = TRUE.

`useCovariateObservation30d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of observation in 30d window prior to or on cohort index date. Only applicable if useCovariateObservation = TRUE.

`useCovariateObservationCount365d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for the count of each observation concept in 365d window prior to or on cohort index date. Only applicable if useCovariateObservation = TRUE.

`useCovariateMeasurement`
: A boolean value (TRUE/FALSE) to determine if covariates derived from OBSERVATION table will be created and included in future models.

`useCovariateMeasurement365d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of measurement in 365d window prior to or on cohort index date. Only applicable if useCovariateMeasurement = TRUE.

`useCovariateMeasurement30d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of measurement in 30d window prior to or on cohort index date. Only applicable if useCovariateMeasurement = TRUE.

`useCovariateMeasurementCount365d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for the count of each measurement concept in 365d window prior to or on cohort index date. Only applicable if useCovariateMeasurement = TRUE.

`useCovariateMeasurementBelow`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of measurement with a numeric value below normal range for latest value within 180d of cohort index. Only applicable if useCovariateMeasurement = TRUE (CDM v5+) or useCovariateObservation = TRUE (CDM v4).

`useCovariateMeasurementAbove`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of measurement with a numeric value above normal range for latest value within 180d of cohort index. Only applicable if useCovariateMeasurement = TRUE (CDM v5+) or useCovariateObservation = TRUE (CDM v4).

`useCovariateConceptCounts`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that count the number of concepts that a person has within each domain (CONDITION, DRUG, PROCEDURE, OBSERVATION)

`useCovariateRiskScores`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that calculate various Risk Scores, including Charlson, DCSI.

`useCovariateRiskScoresCharlson`
: A boolean value (TRUE/FALSE) to determine if the Charlson comorbidity index should be included in the model.

`useCovariateRiskScoresDCSI`
: A boolean value (TRUE/FALSE) to determine if the DCSI score should be included in the model.

`useCovariateRiskScoresCHADS2`
: A boolean value (TRUE/FALSE) to determine if the CHADS2 score should be included in the model.

`useCovariateRiskScoresCHADS2VASc`
: A boolean value (TRUE/FALSE) to determine if the CHADS2VASc score should be included in the model.

`useCovariateInteractionYear`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that represent interaction terms between all other covariates and the year of the cohort index date.

`useCovariateInteractionMonth`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that represent interaction terms between all other covariates and the month of the cohort index date.

`excludedCovariateConceptIds`
: A list of concept IDs that should NOT be used to construct covariates.

`includedCovariateConceptIds`
: A list of concept IDs that should be used to construct covariates.

`deleteCovariatesSmallCount`
: A numeric value used to remove covariates that occur in both cohorts fewer than deleteCovariateSmallCounts time.

Details

creates an object specifying how covariates should be contructed from data in the CDM model.

Value

An object of type `defaultCovariateSettings`, to be used in other functions.

`createHdpsCovariateSettings` *Create HDPS covariate settings*

Description

Create HDPS covariate settings

Usage

```
createHdpsCovariateSettings(useCovariateCohortIdIs1 = FALSE,
  useCovariateDemographics = TRUE, useCovariateDemographicsGender = TRUE,
  useCovariateDemographicsRace = TRUE,
  useCovariateDemographicsEthnicity = TRUE,
  useCovariateDemographicsAge = TRUE, useCovariateDemographicsYear = TRUE,
  useCovariateDemographicsMonth = TRUE,
  useCovariateConditionOccurrence = TRUE,
  useCovariate3DigitIcd9Inpatient180d = FALSE,
  useCovariate3DigitIcd9Inpatient180dMedF = FALSE,
  useCovariate3DigitIcd9Inpatient180d75F = FALSE,
  useCovariate3DigitIcd9Ambulatory180d = FALSE,
  useCovariate3DigitIcd9Ambulatory180dMedF = FALSE,
  useCovariate3DigitIcd9Ambulatory180d75F = FALSE,
  useCovariateDrugExposure = FALSE,
  useCovariateIngredientExposure180d = FALSE,
  useCovariateIngredientExposure180dMedF = FALSE,
  useCovariateIngredientExposure180d75F = FALSE,
  useCovariateProcedureOccurrence = FALSE,
  useCovariateProcedureOccurrenceInpatient180d = FALSE,
  useCovariateProcedureOccurrenceInpatient180dMedF = FALSE,
  useCovariateProcedureOccurrenceInpatient180d75F = FALSE,
  useCovariateProcedureOccurrenceAmbulatory180d = FALSE,
  useCovariateProcedureOccurrenceAmbulatory180dMedF = FALSE,
  useCovariateProcedureOccurrenceAmbulatory180d75F = FALSE,
  excludedCovariateConceptIds = c(), includedCovariateConceptIds = c(),
  deleteCovariatesSmallCount = 100)
```

Arguments

`useCovariateCohortIdIs1`	A boolean value (TRUE/FALSE) to determine if a covariate should be contructed for whether the cohort ID is 1 (currently primarily used in CohortMethod).
`useCovariateDemographics`	A boolean value (TRUE/FALSE) to determine if demographic covariates (age in 5-yr increments, gender, race, ethnicity, year of index date, month of index date) will be created and included in future models.

`useCovariateDemographicsGender`
: A boolean value (TRUE/FALSE) to determine if gender should be included in the model.

`useCovariateDemographicsRace`
: A boolean value (TRUE/FALSE) to determine if race should be included in the model.

`useCovariateDemographicsEthnicity`
: A boolean value (TRUE/FALSE) to determine if ethnicity should be included in the model.

`useCovariateDemographicsAge`
: A boolean value (TRUE/FALSE) to determine if age (in 5 year increments) should be included in the model.

`useCovariateDemographicsYear`
: A boolean value (TRUE/FALSE) to determine if calendar year should be included in the model.

`useCovariateDemographicsMonth`
: A boolean value (TRUE/FALSE) to determine if calendar month should be included in the model.

`useCovariateConditionOccurrence`
: A boolean value (TRUE/FALSE) to determine if covariates derived from CONDITION_OCCURRENCE table will be created and included in future models.

`useCovariate3DigitIcd9Inpatient180d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of condition within inpatient setting in 180d window prior to or on cohort index date. Conditions are aggregated at the ICD-9 3-digit level. Only applicable if useCovariateConditionOccurrence = TRUE.

`useCovariate3DigitIcd9Inpatient180dMedF`
: Similar to `useCovariate3DigitIcd9Inpatient180d`, but now only if the frequency of the ICD-9 code is higher than the median.

`useCovariate3DigitIcd9Inpatient180d75F`
: Similar to `useCovariate3DigitIcd9Inpatient180d`, but now only if the frequency of the ICD-9 code is higher than the 75th percentile.

`useCovariate3DigitIcd9Ambulatory180d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of condition within ambulatory setting in 180d window prior to or on cohort index date. Conditions are aggregated at the ICD-9 3-digit level. Only applicable if useCovariateConditionOccurrence = TRUE.

`useCovariate3DigitIcd9Ambulatory180dMedF`
: Similar to `useCovariate3DigitIcd9Ambulatory180d`, but now only if the frequency of the ICD-9 code is higher than the median.

`useCovariate3DigitIcd9Ambulatory180d75F`
: Similar to `useCovariate3DigitIcd9Ambulatory180d`, but now only if the frequency of the ICD-9 code is higher than the 75th percentile.

`useCovariateDrugExposure`
: A boolean value (TRUE/FALSE) to determine if covariates derived from DRUG_EXPOSURE table will be created and included in future models.

`useCovariateIngredientExposure180d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of drug ingredients within inpatient setting in 180d window prior to or on cohort index date. Only applicable if useCovariateDrugExposure = TRUE.

`useCovariateIngredientExposure180dMedF`
: Similar to `useCovariateIngredientExposure180d`, but now only if the frequency of the ingredient is higher than the median.

`useCovariateIngredientExposure180d75F`
: Similar to `useCovariateIngredientExposure180d`, but now only if the frequency of the ingredient is higher than the 75th percentile.

`useCovariateProcedureOccurrence`
: A boolean value (TRUE/FALSE) to determine if covariates derived from PROCEDURE_OCCURRENCE table will be created and included in future models.

`useCovariateProcedureOccurrenceInpatient180d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of procedures within inpatient setting in 180d window prior to or on cohort index date. Only applicable if useCovariateProcedureOccurrence = TRUE.

`useCovariateProcedureOccurrenceInpatient180dMedF`
: Similar to `useCovariateProcedureOccurrenceInpatient180d`, but now only if the frequency of the procedure code is higher than the median.

`useCovariateProcedureOccurrenceInpatient180d75F`
: Similar to `useCovariateProcedureOccurrenceInpatient180d`, but now only if the frequency of the procedure code is higher than the 75th percentile.

`useCovariateProcedureOccurrenceAmbulatory180d`
: A boolean value (TRUE/FALSE) to determine if covariates will be created and used in models that look for presence/absence of procedures within ambulatory setting in 180d window prior to or on cohort index date. Only applicable if useCovariateProcedureOccurrence = TRUE.

`useCovariateProcedureOccurrenceAmbulatory180dMedF`
: Similar to `useCovariateProcedureOccurrenceAmbulatory180d`, but now only if the frequency of the procedure code is higher than the median.

`useCovariateProcedureOccurrenceAmbulatory180d75F`
: Similar to `useCovariateProcedureOccurrenceAmbulatory180d`, but now only if the frequency of the procedure code is higher than the 75th percentile.

`excludedCovariateConceptIds`
: A list of concept IDs that should NOT be used to construct covariates.

`includedCovariateConceptIds`
: A list of concept IDs that should be used to construct covariates.

`deleteCovariatesSmallCount`
: A numeric value used to remove covariates that occur in both cohorts fewer than deleteCovariateSmallCounts time.

Details

creates an object specifying how covariates should be contructed from data in the CDM model.

Value

An object of type `hdpsCovariateSettings`, to be used in other functions.

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`.

createTextCovariateSettings

Create text covariate settings

Description

Create text covariate settings

Usage

```
createTextCovariateSettings(language = "eng", removeNegations = TRUE,
  deleteCovariatesSmallCount = 100)
```

Arguments

`language`	Specify the language of the free-text.
`removeNegations`	Remove negated text prior to constructing features.
`deleteCovariatesSmallCount`	A numeric value used to remove covariates that occur in both cohorts fewer than deleteCovariateSmallCounts time.

Details

creates an object specifying how covariates should be constructed from text in notes table in the CDM model.

Value

An object of type `covariateSettings`, to be used in other functions.

`describePlpData` *describePlpData*

Description

describePlpData

Usage

```
describePlpData(plpData, covariateVals = NULL, cdmDatabase = NULL,
  outcomeId = 2, agehist = TRUE, plot = T, plotFile = NULL,
  saveTable = T, tableFile = NULL, outcomeName = "Pregnancy with GDM",
  cohortName = "Total pregnancy", perYear = T)
```

Arguments

`plpData`	An object of type `plpData`.
`covariateVals`	A vector of the covariateIds to focus on
`cdmDatabase`	The database the data comes from
`outcomeId`	The outcome of interest
`agehist`	Whether to plot a histogram of ages
`plot`	Whether to plot results
`plotFile`	Location to save plot
`saveTable`	Whether to save results
`tableFile`	Location to save results table
`outcomeName`	Outcome name to be used on plots
`cohortName`	Cohort name to be used on plots
`perYear`	Whether to do descriptive stats per year

Details

Describes the data

developModel2 *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

```
developModel2(plpData, modelSettings, featureSettings, type = c("year",
  "both", "patient"), validationFraction = 0.2, fileLoc = file.path(getwd(),
  "models"))
```

Arguments

`plpData` An object of type `plpData` - the patient level prediction data extracted from the CDM.

`modelSettings` A list of class `modelSettings` containing:

- model - a string specifying the name of classifier function (e.g. 'lr-lasso')
- param - a list containing the model parameters, cohortIds, the outcomeIds.

The default models are:

- lr-lasso - Logistic regression with lasso regularisation - parameters: variance
- nnet_plp - Neural network from caret package- parameters: size/decay
- svmRadial_plp - SVM with radial kernal from caret package - parameters: C, ...
- randomForest_plp - Random forest from h2o package
- gbm_plp - Gradient boosting machine from h2o package
- lr_enet_plp - Logistic regression with elastic new regularisation from h2o package

`featureSettings`
A list of class `featureSettings` containing:

- method - a string specifying the name of feature modifying function (e.g. 'wrapperGA')
- param - a list containing the method parameters, cohortIds, the outcomeIds.

The default preprocessing methods are:

- lassolr - Feature selection using lasso logistic regression
- wrapperGA - Genetic algorithm wrapper
- glrm - (IN PROGRESS)Generaised low rank models
- varImp - Variable importance
- filterCovariates - Filtering covariates

`type`	A subset of c('year','both','patient') specifying the type of evaluation used. 'year' find the date where validationFraction 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 'both' splits the data by the year but removes any patient in the test set from the training set 'patient' splits the data into test (1-validationFraction of the data) and train (validationFraction of the data) sets. The split is stratified by the class label.
`validationFraction`	The fraction of the data to be used as the validation set in the patient split evaluation.
`fileLoc`	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 meta-data
`dataSummary`	A list detailing the size of the train/test sets and outcome prevalence
`evalType`	The type of evaluation that was performed
`prediction`	An ffdf object containing the prediction for each person in the validation set
`performance`	A list detailing the performance of the model
`time`	The complete time taken to do the model framework

Examples

```
#******** EXAMPLE 1 *********
#lasso logistic regression oredicting outcome 2 in cohorts 1 and 3
#using no feature selection with a year split evaluation:
modset_llr  <- list(model='lr_lasso',
                   param=list(variance =0.001, cohortId=c(1,2), outcomeId=2))
class(modset_llr) <- 'modelSettings'
mod_llr <- developModel2(plpData= plpData,
                        featureSettings = NULL,
                        modelSettings = modset_llr ,
                        type='year')

#******** EXAMPLE 2 *********
# Gradient boosting machine using a genetic algorimth wrapper to
# select the feature subset and a grid search to select hyper parameters
featSet_gbm <- list(method='wrapperGA', param=list(cohortId=c(1,2), outcomeId=2, varSize=300, iter=25))
class(featSet_gbm) <- 'featureSettings'
modset_gbm <- list(model='gbm_plp',
               param=list(rsampRate=0.8, ntrees=c(100,150), max_depth=c(2,4,5), cohortId=c(1,2), outcomeId=
class(modset_gbm) <- 'modelSettings'
mod_gbm <- developModel2(plpData= plpData.censor,
                         featureSettings = featSet_gbm,
                         modelSettings = modset_gbm,
                         type='year')
```

evaluatePlp *evaluatePlp*

Description

Evaluates the performance of the patient level prediction model

Usage

```
evaluatePlp(plpPredict, plpData, sparse = T)
```

Arguments

plpPredict	The patient level prediction model's prediction
plpData	An object of type `plpData` - the patient level prediction data extracted from the CDM.
sparse	(boolean) Whether the metrics should be calculated in sparse format

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)
```

`fitPlp2` *fitModel*

Description

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

Usage

```
fitPlp2(data, modelSettings, featureSettings, outcomeId, cohortId, loc)
```

Arguments

`data`	An object of type `plpData` - the patient level prediction data extracted from the CDM.
`modelSettings`	A list of class `modelSettings` containing: • model - a string specifying the name of classifier function (e.g. 'lr-lasso') • param - a list containing the model parameters, cohortIds, the outcomeIds. The default models are: • lr-lasso - Logistic regression with lasso regularisation - parameters: variance • nnet_plp - Neural network from caret package- parameters: size/decay • svmRadial_plp - SVM with radial kernal from caret package - parameters: C, ...

- randomForest_plp - Random forest from h2o package
- gbm_plp - Gradient boosting machine from h2o package
- lr_enet_plp - Logistic regression with elastic new regularisation from h2o package

`featureSettings`	A list of class `featureSettings` containing: • method - a string specifying the name of feature modifying function (e.g. 'wrapperGA') • param - a list containing the method parameters, cohortIds, the outcomeIds. The default preprocessing methods are: • lassolr - Feature selection using lasso logistic regression • wrapperGA - Genetic algorithm wrapper • glrm - (IN PROGRESS)Generaised low rank models • varImp - Variable importance • filterCovariates - Filtering covariates
`outcomeId`	The outcomeId the user is aiming to predict
`cohortId`	The id of the cohort being used. Default is NULL.
`loc`	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(plpData, modelType = "logistic",
  removeDropoutsForLr = TRUE, cohortId = NULL, outcomeId = NULL,
  prior = createPrior("laplace", exclude = c(0), useCrossValidation = TRUE),
  control = createControl(noiseLevel = "silent", cvType = "auto",
  startingVariance = 0.1))
```

Arguments

`plpData`	An object of type `plpData`.
`modelType`	The type of predictive model. Options are "logistic", "poisson", and "survival".
`removeDropoutsForLr`	If TRUE and modelType is "logistic", subjects that do not have the full observation window (i.e. are censored earlier) and do not have the outcome are removed prior to fitting the model.
`cohortId`	The ID of the specific cohort for which to fit a model.
`outcomeId`	The ID of the specific outcome for which to fit a 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.

`getDbConceptCovariateData` *Get HDPS covariate information from the database*

Description

Constructs the set of covariates for one or more cohorts using data in the CDM schema. This implements the covariates based on user input concept sets. This is a way to incorporate known risk factors.

Usage

```
getDbConceptCovariateData(connection, oracleTempSchema = NULL,
  cdmDatabaseSchema, cdmVersion = "5", cohortTempTable = "cohort_person",
  rowIdField = "subject_id", covariateSettings)
```

Arguments

`connection`	A connection to the server containing the schema as created using the `connect` function in the `DatabaseConnector` package.
`oracleTempSchema`	A schema where temp tables can be created in Oracle.
`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'.
`cdmVersion`	Define the OMOP CDM version used: currently support "4" and "5".

`cohortTempTable`
: Name of the temp table holding the cohort for which we want to construct covaraites

`rowIdField`
: The name of the field in the cohort temp table that is to be used as the row_id field in the output table. This can be especially usefull if there is more than one period per person.

`covariateSettings`
: An object of type `covariateSettings` as created using the `createConceptCovariateSettings` function.

Details

This function uses the data in the CDM to construct a large set of covariates for the provided cohort. The cohort is assumed to be in an existing temp table with these fields: 'subject_id', 'cohort_definition_id', 'cohort_start_date'. Optionally, an extra field can be added containing the unique identifier that will be used as rowID in the output. This function is called automatically by the `getDbPlpData` function.

Value

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

covariates An ffdf object 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 object will have three columns: rowId, covariateId, and covariateValue. The rowId is usually equal to the person_id, unless specified otherwise in the rowIdField argument.

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

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

`getDbCovariateData` *Get covariate information from the database*

Description

Uses one or several covariate builder functions to construct covariates.

Usage

```
getDbCovariateData(connection, oracleTempSchema = NULL, cdmDatabaseSchema,
  cdmVersion = "4", cohortTempTable = "cohort_person",
  rowIdField = "subject_id", covariateSettings, normalize = TRUE)
```

Arguments

`connection`
: A connection to the server containing the schema as created using the `connect` function in the `DatabaseConnector` package.

`oracleTempSchema`
: A schema where temp tables can be created in Oracle.

`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'.
`cdmVersion`	Define the OMOP CDM version used: currently support "4" and "5".
`cohortTempTable`	Name of the temp table holding the cohort for which we want to construct covaraites
`rowIdField`	The name of the field in the cohort temp table that is to be used as the row_id field in the output table. This can be especially usefull if there is more than one period per person.
`covariateSettings`	Either an object of type `covariateSettings` as created using one of the createCovariate functions, or a list of such objects.
`normalize`	Should covariate values be normalized? If true, values will be divided by the max value per covariate.

Details

This function uses the data in the CDM to construct a large set of covariates for the provided cohort. The cohort is assumed to be in an existing temp table with these fields: 'subject_id', 'cohort_definition_id', 'cohort_start_date'. Optionally, an extra field can be added containing the unique identifier that will be used as rowID in the output. This function is called automatically by the `getDbPlpData` function.

Value

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

covariates An ffdf object 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 object will have three columns: rowId, covariateId, and covariateValue. The rowId is usually equal to the person_id, unless specified otherwise in the rowIdField argument.

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

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

`getDbDefaultCovariateData`

Get default covariate information from the database

Description

Constructs a large default set of covariates for one or more cohorts using data in the CDM schema. Includes covariates for all drugs, drug classes, condition, condition classes, procedures, observations, etc.

Usage

```
getDbDefaultCovariateData(connection, oracleTempSchema = NULL,
  cdmDatabaseSchema, cdmVersion = "4", cohortTempTable = "cohort_person",
  rowIdField = "subject_id", covariateSettings)
```

Arguments

`connection`	A connection to the server containing the schema as created using the `connect` function in the `DatabaseConnector` package.
`oracleTempSchema`	A schema where temp tables can be created in Oracle.
`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'.
`cdmVersion`	Define the OMOP CDM version used: currently support "4" and "5".
`cohortTempTable`	Name of the temp table holding the cohort for which we want to construct covaraites
`rowIdField`	The name of the field in the cohort temp table that is to be used as the row_id field in the output table. This can be especially usefull if there is more than one period per person.
`covariateSettings`	An object of type `defaultCovariateSettings` as created using the `createCovariateSettings` function.

Details

This function uses the data in the CDM to construct a large set of covariates for the provided cohort. The cohort is assumed to be in an existing temp table with these fields: 'subject_id', 'cohort_definition_id', 'cohort_start_date'. Optionally, an extra field can be added containing the unique identifier that will be used as rowID in the output. This function is called automatically by the `getDbPlpData` function.

Value

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

covariates An ffdf object 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 object will have three columns: rowId, covariateId, and covariateValue. The rowId is usually equal to the person_id, unless specified otherwise in the rowIdField argument.

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

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

getDbHdpsCovariateData

Get HDPS covariate information from the database

Description

Constructs the set of covariates for one or more cohorts using data in the CDM schema. This implements the covariates typically used in the HDPS algorithm.

Usage

```
getDbHdpsCovariateData(connection, oracleTempSchema = NULL, cdmDatabaseSchema,
  cdmVersion = "4", cohortTempTable = "cohort_person",
  rowIdField = "subject_id", covariateSettings)
```

Arguments

`connection`	A connection to the server containing the schema as created using the `connect` function in the `DatabaseConnector` package.
`oracleTempSchema`	A schema where temp tables can be created in Oracle.
`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'.
`cdmVersion`	Define the OMOP CDM version used: currently support "4" and "5".
`cohortTempTable`	Name of the temp table holding the cohort for which we want to construct covaraites
`rowIdField`	The name of the field in the cohort temp table that is to be used as the row_id field in the output table. This can be especially usefull if there is more than one period per person.
`covariateSettings`	An object of type `covariateSettings` as created using the `createHdpsCovariateSettings` function.

Details

This function uses the data in the CDM to construct a large set of covariates for the provided cohort. The cohort is assumed to be in an existing temp table with these fields: 'subject_id', 'cohort_definition_id', 'cohort_start_date'. Optionally, an extra field can be added containing the unique identifier that will be used as rowID in the output. This function is called automatically by the `getDbPlpData` function.

Value

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

covariates An ffdf object 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 object will have three columns: rowId, covariateId, and covariateValue. The rowId is usually equal to the person_id, unless specified otherwise in the rowIdField argument.

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

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

`getDbPlpData` *Get outcomes for persons in the cohort*

Description

Get all the data for the prediction problem from the server.

Usage

```
getDbPlpData(connectionDetails = NULL, cdmDatabaseSchema,
  oracleTempSchema = NULL, cohortDatabaseSchema = cdmDatabaseSchema,
  cohortTable = "cohort", cohortIds = c(0, 1), washoutWindow = 183,
  useCohortEndDate = TRUE, windowPersistence = 0, covariateSettings,
  outcomeDatabaseSchema = cdmDatabaseSchema,
  outcomeTable = "condition_occurrence", outcomeIds = c(),
  outcomeIdsExclude = c(), startAdd = 0, endAdd = NULL,
  outcomeConditionTypeConceptIds = "", firstOutcomeOnly = FALSE,
  excludeHistory = FALSE, cdmVersion = "4")
```

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`	A schema where temp tables can be created in Oracle.
`cohortDatabaseSchema`	Where is the source cohort table located? Note that on SQL Server, one should include both the database and schema, e.g. "cdm_schema.dbo".
`cohortTable`	What is the name of the table holding the cohort?
`cohortIds`	The IDs of the cohorts for which we want to create models.
`washoutWindow`	The mininum required continuous observation time prior to index date for a person to be included in the cohort.
`useCohortEndDate`	Use the cohort end date as the basis for the end of the risk window? If FALSE, the cohort start date will be used instead.
`windowPersistence`	The number of days the risk window should persist.

`covariateSettings`	An object of type `covariateSettings` as created using the `createCovariateSettings` function.
`outcomeDatabaseSchema`	The name of the database schema that is the location where the data used to define the outcome cohorts is available. If outcomeTable = CONDITION_ERA, outcomeDatabaseSchema is not used. 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_CONCEPT_ID, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.
`outcomeIds`	A list of ids used to define outcomes. If outcomeTable = CONDITION_OCCURRENCE, the list is a set of ancestor CONCEPT_IDs, and all occurrences of all descendant concepts will be selected. If outcomeTable <> CONDITION_OCCURRENCE, the list contains records found in COHORT_DEFINITION_ID field.
`outcomeConditionTypeConceptIds`	A list of TYPE_CONCEPT_ID values that will restrict condition occurrences. Only applicable if outcomeTable = CONDITION_OCCURRENCE.
`firstOutcomeOnly`	Only keep the first outcome per person?
`cdmVersion`	Define the OMOP CDM version used: currently support "4" and "5".

Details

For the specified cohorts, retrieve the outcomes of interest and covariates to be used for the prediction problem.

Value

An object of type `plpData` containing information on the prediction problem. This object will contain the following data:

cohorts An ffdf object listing all persons and their prediction periods. This object 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 An ffdf object listing all outcomes per period. This object will have these fields: row_id, outcome_id, outcome_count, time_to_event.

exclude Either NULL or an ffdf object listing per outcome ID which windows had the outcome prior to the window. This object will have these fields: rowId, outcomeId.

covariates An ffdf object 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 object will have three columns: rowId, covariateId, and covariateValue.

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

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

getDbTextCovariateData *Get text covariate information from the database*

Description

Uses a bag-of-words approach to construct covariates based on free-text.

Usage

```
getDbTextCovariateData(connection, oracleTempSchema = NULL, cdmDatabaseSchema,
  cdmVersion = "4", cohortTempTable = "cohort_person",
  rowIdField = "subject_id", covariateSettings)
```

Arguments

`connection`	A connection to the server containing the schema as created using the `connect` function in the `DatabaseConnector` package.
`oracleTempSchema`	A schema where temp tables can be created in Oracle.
`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'.
`cdmVersion`	Define the OMOP CDM version used: currently support "4" and "5".
`cohortTempTable`	Name of the temp table holding the cohort for which we want to construct covaraites
`rowIdField`	The name of the field in the cohort temp table that is to be used as the row_id field in the output table. This can be especially usefull if there is more than one period per person.
`covariateSettings`	An object of type `covariateSettings` as created using the `createTextCovariateSettings` function.

Details

This function uses the data in the CDM to construct a large set of covariates for the provided cohort. The cohort is assumed to be in an existing temp table with these fields: 'subject_id', 'cohort_definition_id', 'cohort_start_date'. Optionally, an extra field can be added containing the unique identifier that will be used as rowID in the output. This function is called automatically by the `getDbPlpData` function.

Value

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

covariates An ffdf object 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 object will have three columns: rowId, covariateId, and covariateValue. The rowId is usually equal to the person_id, unless specified otherwise in the rowIdField argument.

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

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

`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.

`getTPR` *getTPR*

Description

Extracts TPR for specified FPR

Usage

```
getTPR(roc, FPR = 0.05)
```

Arguments

`roc`	A dataframe containing TPR and FPR at range of thresholds

Value

TPR at specified FPR

`loadCovariateData` *Load the covariate data from a folder*

Description

`loadCovariateData` loads an object of type covariateData from a folder in the file system.

Usage

```
loadCovariateData(file, readOnly = FALSE)
```

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 covariateData

Examples

```
# todo
```

`loadPlpData` *Load the PatientLevelPrediction data from a folder*

Description

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

Usage

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

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
```

`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

---

`newDataEval` *Make predictions using model on new data*

Description

Make predictions using model on new data

Usage

```
newDataEval(newData, plpModel)
```

Arguments

`newData`	An object of type `plpData`.
`plpModel`	A model returned by calling developModel2()

Details

Computes the performance of the model on new data

normalizeCovariates *Normalize covariate values*

Description

Normalize covariate values

Usage

```
normalizeCovariates(covariates)
```

Arguments

`covariates`	An ffdf object as generated using the `getDbCovariateData` function.#'

Details

Normalize covariate values by dividing by the max. This is to avoid numeric problems when fitting models.

paramSettings *paramSettings*

Description

A function for specifying the hyperparameters to create a grid search when training the GBM model

Usage

```
paramSettings(model = "gbm", bal = F, ntrees = 100, rsampRate = 1,
  csampRate = 1, learn_rate = 0.01, nbins = 10, max_depth = 4,
  min_rows = 20, mtries = -1, alpha = 0.5, lambda = 1e-06,
  lambda_search = T, nlambdas = 50, lambda_min_ratio = 0.001, ...)
```

Arguments

`bal`	A vector of boolean values - specifying whether to balance the class labels during training
`ntrees`	A vector of integers -specifying the number of trees to train
`rsampRate`	A vector of values between 0 and 1 specifying the fraction of rows to use for each tree
`csampRate`	A vector of values between 0 and 1 specifying the fraction of features to use for each tree

Details

The function takes a list of the model's hyperparameters and values to investigate while training the model

Value

A list the parameters expanded out like a grid ready to be investigated during model training

---

`PatientLevelPrediction` *PatientLevelPrediction*

Description

PatientLevelPrediction

`plotCalibration` *Plot the calibration*

Description

Plot the calibration

Usage

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

Arguments

`prediction`	A prediction object as generated using the `predictProbabilities` function.
`plpData`	An object of type `plpData`.
`removeDropoutsForLr`	If TRUE and modelType is "logistic", subjects that do not have the full observation window (i.e. are censored earlier) and do not have the outcome are removed prior to evaluating the model.
`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 stratefied 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, plpData, removeDropoutsForLr = TRUE, fileName = NULL)
```

Arguments

`prediction`	A prediction object as generated using the `predictProbabilities` function.
`plpData`	An object of type `plpData`.
`removeDropoutsForLr`	If TRUE and modelType is "logistic", subjects that do not have the full observation window (i.e. are censored earlier) and do not have the outcome are removed prior to evaluating the model.
`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, outcomes, 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.
`outcomes`	A data frame or ffdf object containing the outcomes with predefined columns (see below).
`covariates`	A data frame or ffdf object containing the covariates with predefined columns (see below).
`modelType`	Current supported types are "logistic", "poisson", or "survival".

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

`predictPlp2` *predictPlp*

Description

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

Usage

```
predictPlp2(plpModel, plpData)
```

Arguments

`plpModel`	An object of type `plpModel` - a patient level prediction model
`plpData`	An object of type `plpData` - the patient level prediction data extracted from the CDM.

Details

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

Value

An ffdf object containing the prediction for each person in the cohort

`predictProbabilities` *Create predictive probabilities*

Description

Create predictive probabilities

Usage

```
predictProbabilities(predictiveModel, plpData)
```

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.

`saveCovariateData` *Save the covariate data to folder*

Description

`saveCovariateData` saves an object of type covariateData to folder.

Usage

```
saveCovariateData(covariateData, file)
```

Arguments

`covariateData`	An object of type `covariateData` as generated using `getDbCovariateData`.
`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
```

`savePlpData` *Save the PatientLevelPrediction 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`.

`splitData` *Split data into random subsets*

Description

Split data into random subsets

Usage

```
splitData(plpData, splits = 2)
```

Arguments

`plpData`	An object of type `plpData`.
`splits`	This can be either a single integer, in which case the data will be split up into equally sized parts. If a vector is provided instead, these are interpreted as the relative sizes of each part.

Details

Splits cohort, covariate, and outcome data into random subsets, to be used for validation.

Value

A list with entries for each part. An entry itself is a plpData object.

`spliter` *spliter*

Description

Various train/test splitting techniques

Usage

```
spliter(plpData, type, frac)
```

Arguments

`plpData`	An object of type `plpData` - the patient level prediction data extracted from the CDM.
`type`	The type of train/test split: • 'year'Split the data based on cohort start date - test:pre 2013/train:post2013 • 'both'Split the data on year 2013 but exclude any train people who are in the test set. • 'patient'Split the data into (1-frac) train set and frac test set. This is stratified by the outcome
`frac`	The fraction of people who will go into the test set

Details

The function splits the plpData into test/train sets

Value

A list with the training set as the first element and the testing set as the second element

`varImportance` *varImportance*

Description

Plots the variable importance of the patient level prediction model/s and returns a table of variable importances

Usage

```
varImportance(models.list)
```

Arguments

`models.list`	A list of plp models or simple plp model

Value

A table containing the variable importances for each model

Index