Package ‘PatientLevelPrediction’

July 26, 2015

Type Package

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

Version 0.0.2

Date 2015-06-27

Author Martijn J. Schuemie [aut, cre],
Marc A. Suchard [aut],
Patrick B. Ryan [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 (>= 3.1.0),
DatabaseConnector (>= 1.1.2),
Cyclops (>= 1.0.0)

Imports ggplot2,
bit,
ff,
ffbase (>= 0.12.1),
plyr,
survAUC,
Rcpp (>= 0.11.2),
RJDBC,
SqlRender (>= 1.1.0),
survival

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

LinkingTo Rcpp

NeedsCompilation yes

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, outcomeData, confidenceInterval = FALSE)
```

Arguments

prediction	A prediction object as generated using the predictProbabilities function.
outcomeData	An object of type outcomeData.
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.

createCovariateSettings

Create covariate settings

Description

Create covariate settings

Usage

```
createCovariateSettings(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,
  useCovariateInteractionYear = FALSE, useCovariateInteractionMonth = FALSE,
  excludedCovariateConceptIds = c(), includedCovariateConceptIds = c(),
  deleteCovariatesSmallCount = 100)
```

Arguments

`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.

`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 `covariateSettings`, to be used in other functions.

fitPredictiveModel *Fit a predictive model*

Description

Fit a predictive model

Usage

```
fitPredictiveModel(cohortData, covariateData, outcomeData,
  modelType = "logistic", cohortId = NULL, outcomeId = NULL,
  prior = createPrior("laplace", exclude = c(0), useCrossValidation = TRUE),
  control = createControl(noiseLevel = "silent", cvType = "auto",
  startingVariance = 0.1))
```

Arguments

`cohortData`	An object of type `cohortData`.
`covariateData`	An object of type `covariateData`.
`outcomeData`	An object of type `outcomeData`.
`modelType`	The type of predictive model. Options are "logistic", "poisson", and "survival".
`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.

getDbCohortData *Get cohorts of interest*

Description

Gets the cohorts of interest from the database

Usage

```
getDbCohortData(connectionDetails = NULL, connection = NULL,
  cdmDatabaseSchema, oracleTempSchema = cdmDatabaseSchema,
  useExistingCohortPerson = FALSE, cohortDatabaseSchema = cdmDatabaseSchema,
  cohortTable = "cohort", cohortIds = c(0, 1), cdmVersion = "4")
```

Arguments

`connectionDetails`
: An R object of type `ConnectionDetails` created using the function `createConnectionDetails` in the `DatabaseConnector` package.

`connection`
: A connection to the server containing the schema as created using the `connect` function 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.#'

`useExistingCohortPerson`
: Does the temporary table `cohort_person` already exists? Can only be used when the `connection` parameter is not NULL.

`cohortDatabaseSchema`
: If not using an existing `cohort_person` temp table, 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`
: If not using an existing temp table, what is the name of the table holding the cohort?

`cohortIds`
: The list of IDs in the cohortTable that identify the cohort(s) of interest.

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

Value

An object of type `cohortData` containing information on who are in the cohorts.

`getDbCovariateData` *Get covariate information from the database*

Description

Constructs a large set of covariates for one or more cohorts using data in the CDM schema.

Usage

```
getDbCovariateData(connectionDetails = NULL, connection = NULL,
  oracleTempSchema = NULL, cdmDatabaseSchema,
  useExistingCohortPerson = FALSE, cohortDatabaseSchema = cdmDatabaseSchema,
  cohortTable = "cohort", cohortIds = c(0, 1), covariateSettings,
  cdmVersion = "4")
```

Arguments

`connectionDetails`
: An R object of type `connectionDetails` created using the function `createConnectionDetails` in the `DatabaseConnector` package.

`connection`
: A connection to the server containing the schema as created using the `connect` function 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'.

`useExistingCohortPerson`
: Does the temporary table `cohort_person` already exists? Can only be used when the `connection` parameter is not NULL.

`cohortDatabaseSchema`
: If not using an existing `cohort_person` temp table, 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`
: If not using an existing `cohort_person` temp table, what is the name of the source cohort table?

`cohortIds`
: The IDs of the cohortsin the cohort table for which we want to build covariates.

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

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

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

Details

This function uses the data in the CDM to construct a large set of covariates for the provided cohorts. The cohorts are assumed to be in a table with the same structure as the cohort table in the OMOP CDM. The subject_id in this table must refer to person_ids in the CDM. One person can occurr multiple times, but the combination of subject_id and cohort_start_date is assumed to be unique.

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 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 covariateData object was constructed.

getDbOutcomeData *Get outcomes for persons in the cohort*

Description

Gets the outcomes for the specified cohort(s).

Usage

```
getDbOutcomeData(connectionDetails = NULL, connection = NULL,
  cdmDatabaseSchema, oracleTempSchema = cdmDatabaseSchema,
  useExistingCohortPerson = FALSE, cohortDatabaseSchema = cdmDatabaseSchema,
  cohortTable = "cohort", cohortIds = c(0, 1),
  outcomeDatabaseSchema = cdmDatabaseSchema,
  outcomeTable = "condition_occurrence", outcomeIds = c(),
  outcomeConditionTypeConceptIds = "", firstOutcomeOnly = FALSE,
  cdmVersion = "4")
```

Arguments

`connectionDetails`
: An R object of type `connectionDetails` created using the function `createConnectionDetails` in the `DatabaseConnector` package.

`connection`
: A connection to the server containing the schema as created using the `connect` function 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.#'

`useExistingCohortPerson`
: Does the temporary table `cohort_person` already exists? Can only be used when the `connection` parameter is not NULL.

`cohortDatabaseSchema`
: If not using an existing `cohort_person` temp table, 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`
: If not using an existing temp table, what is the name of the table holding the cohort?

`cohortIds`
: The IDs of the cohorts.

`outcomeDatabaseSchema`
: The name of the database schema that is the location where the data used to define the outcome cohorts is available. If exposureTable = 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_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 during cohort start and end date. Either a `connectionDetails` or a `connection` object has to be specified.

Value

An object of type `outcomeData` containing information on the outcomes in the cohort(s).

`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, covariateData)
```

Arguments

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

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

Details

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

`loadCohortData` *Load the cohorts data from a folder*

Description

`loadCohortData` loads an object of type cohortData from a folder in the file system.

Usage

```
loadCohortData(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 cohortData

`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
```

`loadOutcomeData` *Load the outcome data from a folder*

Description

`loadOutcomeData` loads an object of type outcomeData from a folder in the file system.

Usage

```
loadOutcomeData(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 outcomeData

`PatientLevelPrediction` *PatientLevelPrediction*

Description

PatientLevelPrediction

`plotCalibration` *Plot the calibration*

Description

Plot the calibration

Usage

```
plotCalibration(prediction, outcomeData, numberOfStrata = 5,
  fileName = NULL)
```

Arguments

`prediction`	A prediction object as generated using the `predictProbabilities` function.
`outcomeData`	An object of type `outcomeData`.
`numberOfStrata`	The number of strata in the 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.

Details

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

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, outcomeData, fileName = NULL)
```

Arguments

`prediction`	A prediction object as generated using the `predictProbabilities` function.
`outcomeData`	An object of type `outcomeData`.
`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.

`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

`predictProbabilities` *Create predictive probabilities*

Description

Create predictive probabilities

Usage

```
predictProbabilities(predictiveModel, cohortData, covariateData)
```

Arguments

predictiveModel
: An object of type predictiveModel as generated using fitPredictiveModel.

cohortData
: An object of type cohortData as generated using getDbCohortData.

covariateData
: An object of type covariateData as generated using getDbCovarteData.

Details

Note that the cohortData and covariateData objects need to come from the same population.

saveCohortData *Save the cohort data to folder*

Description

saveCohortData saves an object of type cohortData to folder.

Usage

```
saveCohortData(cohortData, file)
```

Arguments

cohortData
: An object of type cohortData as generated using getDbcohortData.

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
```

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
```

saveOutcomeData *Save the outcome data to folder*

Description

`saveOutcomeData` saves an object of type outcomeData to folder.

Usage

```
saveOutcomeData(outcomeData, file)
```

Arguments

outcomeData	An object of type `outcomeData` as generated using `getDbOutcomeData`.
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.

splitData *Split data into random subsets*

Description

Split data into random subsets

Usage

```
splitData(cohortData, covariateData, outcomeData, splits = 2)
```

Arguments

`cohortData` An object of type `cohortData`.

`covariateData` An object of type `covariateData`.

`outcomeData` An object of type `outcomeData`.

`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 list containing a cohortData, covariateData, and outcomeData object.

Index