Package 'CohortMethod'

July 28, 2015

Type Package

Title New-user cohort method with large scale propensity and outcome models

Version 1.1.0

Date 2015-07-01

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

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

Description CohortMethod is an R package for performing new-user cohort studies in an observational database in the OMOP Common Data Model. It extracts the necessary data from a database in OMOP Common Data Model format, and uses a large set of covariates for both the propensity and outcome model, including for example all drugs, diagnoses, procedures, as well as age, comorbidity indexes, etc. Large scale regularized regression is used to fit the propensity and outcome models. Functions are included for trimming, stratifying and matching on propensity scores, as well as diagnostic functions, such as propensity score distribution plots and plots showing covariate balance before and after matching and/or trimming. Supported outcome models are (conditional) logistic regression, (conditional) Poisson regression, and (stratified) Cox regression.

License Apache License 2.0

VignetteBuilder knitr

Depends R (>= 3.1.0),
DatabaseConnector (>= 1.3.0),
Cyclops (>= 1.1.0),
PatientLevelPrediction (>= 0.0.2)

Imports bit,
ggplot2,
ff,
ffbase (>= 0.12.1),
plyr,
Rcpp (>= 0.11.2),
RJDBC,
SqlRender (>= 1.1.1),
survival,
rjson,
OhdsiRTools

Suggests testthat,
pROC,

gnm,
knitr,
rmarkdown

LinkingTo Rcpp

NeedsCompilation yes

R topics documented:

`CohortMethod` *CohortMethod*

Description

CohortMethod

`cohortMethodDataSimulationProfile`
A simulation profile

Description

A simulation profile

Usage

```
data(cohortMethodDataSimulationProfile)
```

`computeCovariateBalance`
Compute covariate balance before and after matching and trimming

---

Description

For every covariate, prevalence in treatment and comparator groups before and after matching/trimming are computed.

Usage

```
computeCovariateBalance(restrictedCohorts, cohortMethodData, outcomeId = NULL)
```

Arguments

`restrictedCohorts`	A data frame containing the people that are remaining after matching and/or trimming.
`cohortMethodData`	An object of type `cohortMethodData` as generated using `getDbCohortMethodData`.
`outcomeId`	The concept ID of the outcome. Persons marked for removal for the outcome will be removed when computing the balance before matching/trimming.

Details

The restrictedCohorts data frame should have at least the following columns:

rowId	(integer)	A unique identifier for each row (e.g. the person ID)
treatment	(integer)	Column indicating whether the person is in the treated (1) or comparator (0) group

Value

Returns a date frame describing the covariate balance before and after matching/trimming.

`computePsAuc` *Compute the area under the ROC curve*

Description

`computePsAuc` computes the area under the ROC curve of the propensity score

Usage

```
computePsAuc(data, confidenceIntervals = FALSE)
```

Arguments

`data`	A data frame with at least the two columns described below
`confidenceIntervals`	Compute 95 percent confidence intervals (computationally expensive for large data sets)

Details

The data frame should have a least the following two columns:

treatment	(integer)	Column indicating whether the person is in the treated (1) or comparator (0) group
propensityScore	(real)	Propensity score

Value

A data frame holding the AUC and its 95 percent confidence interval

Examples

```
treatment <- rep(0:1, each = 100)
propensityScore <- c(rnorm(100, mean = 0.4, sd = 0.25), rnorm(100, mean = 0.6, sd = 0.25))
data <- data.frame(treatment = treatment, propensityScore = propensityScore)
data <- data[data$propensityScore > 0 & data$propensityScore < 1, ]
computePsAuc(data)
```

---

constructEras *Build eras*

Description

Constructs eras (continuous periods of exposure or disease).

Usage

```
constructEras(connectionDetails, sourceDatabaseSchema,
  sourceTable = "drug_exposure",
  targetDatabaseSchema = sourceDatabaseSchema, targetTable = "drug_era",
  createTargetTable = FALSE, cdmDatabaseSchema = sourceDatabaseSchema,
  gracePeriod = 30, rollUp = TRUE, rollUpConceptClassId = "Ingredient",
  rollUpVocabularyId = "RxNorm", cdmVersion = "5")
```

Arguments

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

`sourceDatabaseSchema`
: The name of the database schema that contains the source table. Requires read permissions to this database. On SQL Server, this should specifiy both the database and the schema, so for example 'cdm_instance.dbo'.

`sourceTable`
: The name of the source table.

`targetDatabaseSchema`
: The name of the database schema that contains the target table. Requires write permissions to this database. On SQL Server, this should specifiy both the database and the schema, so for example 'cdm_instance.dbo'.

`targetTable`
: The name of the target table.

`createTargetTable`
: Should the target table be created? If not, the data is inserted in an existing table.

`cdmDatabaseSchema`
: Only needed when rolling up concepts to ancestors: The name of the database schema that contains the vocabulary files. Requires read permissions to this database. On SQL Server, this should specifiy both the database and the schema, so for example 'cdm_instance.dbo'.

gracePeriod	The number of days allowed between periods for them to still be considered part of the same era.
rollUp	Should concepts be rolled up to their ancestors?
rollUpConceptClassId	The identifier of the concept class to which concepts should be rolled up.
rollUpVocabularyId	The identifier of the vocabulary to which concepts should be rolled up.
cdmVersion	The verion of the CDM that is being used.

Details

This function creates eras from source data. For example, one could use this function to create drug eras based on drug exposures. The function allows drugs to be rolled up to ingredients, and prescriptions to the same ingredient that overlap in time are merged into a single ingredient. Note that stockpiling is not assumed to take place (ie. overlap is discarded), but a grace period can be specified allowing for a small gap between prescriptions when merging. The user can specify the source and target table. These tables are assumed to have the same structure as the cohort table in the Common Data Model (CDM), except when the table names are 'drug_exposure' or 'condition_occurrence' for the source table, or 'drug_era' or 'condition_era' for the target table, in which case the tables are assumed to have the structure defined for those tables in the CDM. If both the source and target table specify a field for type_concept_id, the era construction will partition by the type_concept_id, in other words periods with different type_concept_ids will be treated independently.

Examples

```
## Not run:
# Constructing drug eras in CDM v4:
constructEras(connectionDetails,
              sourceDatabaseSchema = cdmDatabaseSchema,
              sourceTable = "drug_exposure",
              targetTable = "drug_era",
              createTargetTable = FALSE,
              gracePeriod = 30,
              rollUpVocabularyId = 8,
              rollUpConceptClassId = "Ingredient",
              cdmVersion = "4")

# Constructing drug eras in CDM v5:
constructEras(connectionDetails,
              sourceDatabaseSchema = cdmDatabaseSchema,
              sourceTable = "drug_exposure",
              targetTable = "drug_era",
              createTargetTable = FALSE,
              gracePeriod = 30,
              rollUpVocabularyId = "RxNorm",
              rollUpConceptClassId = "Ingredient",
              cdmVersion = "5")


## End(Not run)
```

createCmAnalysis *Create a CohortMethod analysis specification*

Description

Create a CohortMethod analysis specification

Usage

```
createCmAnalysis(analysisId = 1, description = "", targetType = NULL,
  comparatorType = NULL, indicationType = NULL, getDbCohortMethodDataArgs,
  createPs = FALSE, createPsArgs = NULL, trimByPs = FALSE,
  trimByPsArgs = NULL, trimByPsToEquipoise = FALSE,
  trimByPsToEquipoiseArgs = NULL, matchOnPs = FALSE, matchOnPsArgs = NULL,
  matchOnPsAndCovariates = FALSE, matchOnPsAndCovariatesArgs = NULL,
  stratifyByPs = FALSE, stratifyByPsArgs = NULL,
  stratifyByPsAndCovariates = FALSE, stratifyByPsAndCovariatesArgs = NULL,
  computeCovariateBalance = FALSE, fitOutcomeModel = FALSE,
  fitOutcomeModelArgs = NULL)
```

Arguments

analysisId	An integer that will be used later to refer to this specific set of analysis choices.
description	A short description of the analysis.
comparatorType	If more than one comparator is provided for each drugComparatorOutcome, this field should be used to select the specific comparator to use in this analysis.
indicationType	If more than one indication is provided for each drugComparatorOutcome, this field should be used to select the specific indication to use in this analysis.
getDbCohortMethodDataArgs	An object representing the arguments to be used when calling the getDbCohortMethodData function.
createPs	Should the createPs function be used in this analysis?
createPsArgs	An object representing the arguments to be used when calling the createPs function.
trimByPs	Should the trimByPs function be used in this analysis?
trimByPsArgs	An object representing the arguments to be used when calling the trimByPs function.
trimByPsToEquipoise	Should the trimByPsToEquipoise function be used in this analysis?
trimByPsToEquipoiseArgs	An object representing the arguments to be used when calling the trimByPsToEquipoise function.
matchOnPs	Should the matchOnPs function be used in this analysis?
matchOnPsArgs	An object representing the arguments to be used when calling the matchOnPs function.
matchOnPsAndCovariates	Should the matchOnPsAndCovariates function be used in this analysis?

matchOnPsAndCovariatesArgs
: An object representing the arguments to be used when calling the matchOnPsAndCovariates function.

stratifyByPs
: Should the stratifyByPs function be used in this analysis?

stratifyByPsArgs
: An object representing the arguments to be used when calling the stratifyByPs function.

stratifyByPsAndCovariates
: Should the stratifyByPsAndCovariates function be used in this analysis?

stratifyByPsAndCovariatesArgs
: An object representing the arguments to be used when calling the stratifyByPsAndCovariates function.

computeCovariateBalance
: Should the computeCovariateBalance function be used in this analysis?

fitOutcomeModel
: Should the fitOutcomeModel function be used in this analysis?

fitOutcomeModelArgs
: An object representing the arguments to be used when calling the fitOutcomeModel function.

Details

Create a set of analysis choices, to be used with the runCmAnalyses function.

---

createCohortMethodDataSimulationProfile

Create simulation profile

Description

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

Usage

```
createCohortMethodDataSimulationProfile(cohortMethodData)
```

Arguments

cohortMethodData
: An object of type cohortMethodData as generated using getDbCohortMethodData.

Details

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

Value

An object of type cohortDataSimulationProfile.

createCreatePsArgs *Create a parameter object for the function createPs*

Description

Create a parameter object for the function createPs

Usage

```
createCreatePsArgs(excludeCovariateIds = NULL,
  prior = createPrior("laplace", exclude = c(0), useCrossValidation = TRUE),
  control = createControl(noiseLevel = "silent", cvType = "auto",
  startingVariance = 0.1))
```

Arguments

excludeCovariateIds	Exclude these covariates from the propensity model.
prior	The prior used to fit the model. See createPriorfor details.
control	The control object used to control the cross-validation used todetermine the hyperparameters of the prior (if applicable). SeecreateControl for details.

Details

Create an object defining the parameter values.

createDrugComparatorOutcomes *Create drug-comparator-outcomes combinations.*

Description

Create drug-comparator-outcomes combinations.

Usage

```
createDrugComparatorOutcomes(targetId, comparatorId, outcomeIds,
  indicationConceptIds = c(), exclusionConceptIds = c(),
  excludedCovariateConceptIds = c(), includedCovariateConceptIds = c())
```

Arguments

targetId	A concept ID indentifying the target drug in the exposure table. If multiple strategies for picking the target will be tested in the analysis, a named list of numbers can be provided instead. In the analysis, the name of the number to be used can be specified using the #' targetType parameter in the createCmAnalysis function.

`comparatorId` A concept ID indentifying the comparator drug in the exposure table. If multiple strategies for picking the comparator will be tested in the analysis, a named list of numbers can be provided instead. In the analysis, the name of the number to be used can be specified using the #' `comparatorType` parameter in the `createCmAnalysis` function.

`outcomeIds` A vector of concept IDs indentifying the outcome(s) in the outcome table.

`indicationConceptIds`
A vector of concept IDs identifying conditions that are required to appear prior to or on the index date. If multiple strategies for picking the indication will be tested in the analysis, a named list of vectors can be provided instead. In the analysis, the name of the vector to be used can be specified using the `indicationType` parameter in the `createCmAnalysis` function.

`exclusionConceptIds`
A list of concept IDs that cannot appear on or before the index date. This argument is to be used only for exclusion criteria that are specific to the drug-comparator combination.

`excludedCovariateConceptIds`
A list of concept IDs that cannot be used to construct covariates. This argument is to be used only for exclusion concepts that are specific to the drug-comparator combination.

`includedCovariateConceptIds`
A list of concept IDs that must be used to construct covariates. This argument is to be used only for inclusion concepts that are specific to the drug-comparator combination.

Details

Create a set of hypotheses of interest, to be used with the `runCmAnalyses` function.

`createFitOutcomeModelArgs` *Create a parameter object for the function fitOutcomeModel*

Description

Create a parameter object for the function fitOutcomeModel

Usage

```
createFitOutcomeModelArgs(stratifiedCox = TRUE, riskWindowStart = 0,
  riskWindowEnd = 9999, addExposureDaysToEnd = FALSE,
  useCovariates = TRUE, fitModel = TRUE, modelType = "cox",
  prior = createPrior("laplace", useCrossValidation = TRUE),
  control = createControl(cvType = "auto", startingVariance = 0.1,
  selectorType = "byPid", noiseLevel = "quiet"))
```

Arguments

stratifiedCox	Specifically for Cox regressions: specify whether to use the stratadefined in subPopulation in the analysis. For Poissonregression and logistic regression, this is implied in 'clr' and'cpr'.
riskWindowStart	The start of the risk window (in days) relative to the index data.
riskWindowEnd	The end of the risk window (in days) relative to the index data (+days of exposure if the addExposureDaysToEnd parameter isspecified).
addExposureDaysToEnd	Add the length of exposure the risk window?
useCovariates	Whether to use the covariate matrix in the cohortMethodData in theoutcome model.
fitModel	If false, the model will not be fit, and only summary statistics areavailable.
modelType	The type of model to be fitted. See details for options.
prior	The prior used to fit the model. SeecreatePrior for details.
control	The control object used to control the cross-validation used todetermine the hyperparameters of the prior (if applicable). SeecreateControl for details.

Details

Create an object defining the parameter values.

---

createGetDbCohortMethodDataArgs

Create a parameter object for the function getDbCohortMethodData

---

Description

Create a parameter object for the function getDbCohortMethodData

Usage

```
createGetDbCohortMethodDataArgs(washoutWindow = 183,
  indicationLookbackWindow = 183, studyStartDate = "", studyEndDate = "",
  exclusionConceptIds = c(), outcomeConditionTypeConceptIds = c(),
  cdmVersion = "4", excludeDrugsFromCovariates = TRUE, covariateSettings)
```

Arguments

washoutWindow	The mininum required continuous observation time prior to indexdate for a person to be included in the cohort.
indicationLookbackWindow	NA
studyStartDate	A calendar date specifying the minimum date that a cohort indexdate can appear. Date format is 'yyyymmdd'.
studyEndDate	A calendar date specifying the maximum date that a cohort indexdate can appear. Date format is 'yyyymmdd'.

`exclusionConceptIds`
: A list of CONCEPT_IDs used to restrict the cohorts, based on anydescendant conditions/drugs/procedures occurring at least onceanytime prior to the cohort index date.

`outcomeConditionTypeConceptIds`
: A list of TYPE_CONCEPT_ID values that will restrictcondition occurrences. Only applicable if outcomeTable =CONDITION_OCCURRENCE.

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

`excludeDrugsFromCovariates`
: Should the target and comparator drugs (and their descendantconcepts) be excluded from the covariates? Note that this willwork if the drugs are actualy drug concept IDs (and not cohortIDs).

`covariateSettings`
: An object of type covariateSettings as created using thecreateCovariateSettings function in the PatientLevelPrediction package..

Details

Create an object defining the parameter values.

`createMatchOnPsAndCovariatesArgs` *Create a parameter object for the function matchOnPsAndCovariates*

Description

Create a parameter object for the function matchOnPsAndCovariates

Usage

```
createMatchOnPsAndCovariatesArgs(caliper = 0.25,
  caliperScale = "standardized", maxRatio = 1, covariateIds)
```

Arguments

`caliper`
: The caliper for matching. A caliper is the distance which is acceptablefor any match. Observations which are outside of the caliper are dropped.A caliper of 0 means no caliper is used.

`caliperScale`
: The scale on which the caliper is defined. Two scales aresupported:caliperScale = 'propensity score' orcaliperScale = 'standardized'. On the standardized scale, thecaliper is interpreted in standard deviations of the propensity scoredistribution.

`maxRatio`
: The maximum number of persons int the comparator arm to be matched to eachperson in the treatment arm. A maxRatio of 0 means no maximum: allcomparators will be assigned to a treated person.

`covariateIds`
: One or more covariate IDs in the cohortMethodData object on whichsubjects should be also matched.

Details

Create an object defining the parameter values.

createMatchOnPsArgs *Create a parameter object for the function matchOnPs*

Description

Create a parameter object for the function matchOnPs

Usage

```
createMatchOnPsArgs(caliper = 0.25, caliperScale = "standardized",
  maxRatio = 1, stratificationColumns = c())
```

Arguments

caliper	The caliper for matching. A caliper is the distance which isacceptable for any match. Observations which are outside of thecaliper are dropped. A caliper of 0 means no caliper is used.
caliperScale	The scale on which the caliper is defined. Two scales are supported:caliperScale = 'propensity score' or caliperScale ='standardized'. On the standardized scale, the caliper isinterpreted in standard deviations of the propensity scoredistribution.
maxRatio	The maximum number of persons int the comparator arm to be matched toeach person in the treatment arm. A maxRatio of 0 means no maximum:all comparators will be assigned to a treated person.
stratificationColumns	Names or numbers of one or more columns in the data data.frameon which subjects should be stratified prior to matching. No personswill be matched with persons outside of the strata identified by thevalues in these columns.

Details

Create an object defining the parameter values.

createPs *Create propensity scores*

Description

createPs creates propensity scores using a regularized logistic regression.

Usage

```
createPs(cohortMethodData, outcomeId = NULL, excludeCovariateIds = NULL,
  prior = createPrior("laplace", exclude = c(0), useCrossValidation = TRUE),
  control = createControl(noiseLevel = "silent", cvType = "auto",
  startingVariance = 0.1))
```

Arguments

cohortMethodData	An object of type cohortMethodData as generated using getDbCohortMethodData.
outcomeId	The concept ID of the outcome. Persons marked for removal for the outcome will be removed prior to creating the propensity score model.
excludeCovariateIds	Exclude these covariates from the propensity 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.

Details

createPs creates propensity scores using a regularized logistic regression.

Examples

```
data(cohortDataSimulationProfile)
cohortMethodData <- simulateCohortMethodData(cohortDataSimulationProfile, n = 1000)
ps <- createPs(cohortMethodData)
```

---

createStratifyByPsAndCovariatesArgs

Create a parameter object for the function stratifyByPsAndCovariates

---

Description

Create a parameter object for the function stratifyByPsAndCovariates

Usage

```
createStratifyByPsAndCovariatesArgs(numberOfStrata = 5, covariateIds)
```

Arguments

numberOfStrata	Into how many strata should the propensity score be divided? Theboundaries of the strata are automatically defined to contain equalnumbers of treated persons.
covariateIds	One or more covariate IDs in the cohortMethodData object on whichsubjects should also be stratified.

Details

Create an object defining the parameter values.

createStratifyByPsArgs *Create a parameter object for the function stratifyByPs*

Description

Create a parameter object for the function stratifyByPs

Usage

```
createStratifyByPsArgs(numberOfStrata = 5, stratificationColumns = c())
```

Arguments

numberOfStrata	How many strata? The boundaries of the strata are automaticallydefined to contain equal numbers of treated persons.
stratificationColumns	Names of one or more columns in the data data.frame on whichsubjects should also be stratified in addition to stratification onpropensity score.

Details

Create an object defining the parameter values.

createTrimByPsArgs *Create a parameter object for the function trimByPs*

Description

Create a parameter object for the function trimByPs

Usage

```
createTrimByPsArgs(trimFraction = 0.05)
```

Arguments

trimFraction	This fraction will be removed from each treatment group. In the treatmentgroup, persons with the highest propensity scores will be removed, in thecomparator group person with the lowest scores will be removed.

Details

Create an object defining the parameter values.

createTrimByPsToEquipoiseArgs *Create a parameter object for the function trimByPsToEquipoise*

Description

Create a parameter object for the function trimByPsToEquipoise

Usage

```
createTrimByPsToEquipoiseArgs(bounds = c(0.25, 0.75))
```

Arguments

bounds	The upper and lower bound on the preference score for keeping persons

Details

Create an object defining the parameter values.

drawAttritionDiagram *Draw the attrition diagram*

Description

drawAttritionDiagram draws the attition diagram, showing how many people were excluded from the study population, and for what reasons.

Usage

```
drawAttritionDiagram(outcomeModel, treatmentLabel = "Treated",
  comparatorLabel = "Comparator", fileName = NULL)
```

Arguments

outcomeModel	An object of type outcomeModel as generated using he createOutcomeMode function.
treatmentLabel	A label to us for the treated cohort.
comparatorLabel	A label to us for the comparator cohort.
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.

`fitOutcomeModel` *Create an outcome model, and compute the relative risk*

Description

`fitOutcomeModel` creates an outcome model, and computes the relative risk

Usage

```
fitOutcomeModel(outcomeId, cohortMethodData, subPopulation = NULL,
  stratifiedCox = TRUE, riskWindowStart = 0, riskWindowEnd = 9999,
  addExposureDaysToEnd = FALSE, useCovariates = TRUE, fitModel = TRUE,
  modelType = "cox", prior = createPrior("laplace", useCrossValidation =
  TRUE), control = createControl(cvType = "auto", startingVariance = 0.1,
  selectorType = "byPid", noiseLevel = "quiet"))
```

Arguments

`outcomeId`	The concept ID of the outcome. Persons marked for removal for the outcome will be removed prior to creating the outcome model.
`cohortMethodData`	An object of type `cohortMethodData` as generated using `getDbCohortMethodData`.
`subPopulation`	A data frame specifying the (matched and/or trimmed) subpopulation to be used in the study, as well as their strata (for conditional models). This data frame should have at least a `RowId`, and a `StratumId` when including stratification.
`stratifiedCox`	Specifically for Cox regressions: specify whether to use the strata defined in `subPopulation` in the analysis. For Poisson regression and logistic regression, this is implied in 'clr' and 'cpr'.
`riskWindowStart`	The start of the risk window (in days) relative to the index data.
`riskWindowEnd`	The end of the risk window (in days) relative to the index data (+ days of exposure if the `addExposureDaysToEnd` parameter is specified).
`addExposureDaysToEnd`	Add the length of exposure the risk window?
`useCovariates`	Whether to use the covariate matrix in the cohortMethodData in the outcome model.
`fitModel`	If false, the model will not be fit, and only summary statistics are available.
`modelType`	The type of model to be fitted. See details for options.
`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.

Details

The model type can be one of these:

lr	Logistic regression
clr	Conditional logistic regression

cox	Cox regression (stratified or not, depending on whether `stata` is specified)
pr	Poisson regression
cpr	Conditional Poisson regression

Value

An object of class `outcomeModel`. Generic function `summary`, `coef`, and `confint` are available.

Examples

```
# todo
```

`getDbCohortMethodData` *Get the cohort data from the server*

Description

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

Usage

```
getDbCohortMethodData(connectionDetails, cdmDatabaseSchema,
  oracleTempSchema = cdmDatabaseSchema, targetId, comparatorId,
  indicationConceptIds = c(), washoutWindow = 183,
  indicationLookbackWindow = 183, studyStartDate = "", studyEndDate = "",
  exclusionConceptIds = c(), outcomeIds,
  outcomeConditionTypeConceptIds = c(),
  exposureDatabaseSchema = cdmDatabaseSchema, exposureTable = "drug_era",
  outcomeDatabaseSchema = cdmDatabaseSchema,
  outcomeTable = "condition_occurrence", cdmVersion = "4",
  excludeDrugsFromCovariates = TRUE, covariateSettings)
```

Arguments

`connectionDetails`	An R object of type `connectionDetails` created using the function `createConnectionDetails` in the `DatabaseConnector` package.
`cdmDatabaseSchema`	The name of the database schema that contains the OMOP CDM instance. Requires read permissions to this database. On SQL Server, this should specifiy both the database and the schema, so for example 'cdm_instance.dbo'.
`oracleTempSchema`	For Oracle only: the name of the database schema where you want all temporary tables to be managed. Requires create/insert permissions to this database.
`targetId`	A unique identifier to define the target cohort. If exposureTable = DRUG_ERA, targetId is a CONCEPT_ID and all descendant concepts within that CONCEPT_ID will be used to define the cohort. If exposureTable <> DRUG_ERA, targetId is used to select the cohort_concept_id in the cohort-like table.

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

`indicationConceptIds`
A list of CONCEPT_IDs used to restrict the target and comparator cohorts, based on any descendant condition of this list occurring at least once within the indicationLookbackWindow prior to the cohort index date. If no concept IDs are specified, the cohorts are not restricted to any indication.

`washoutWindow` The mininum required continuous observation time prior to index date for a person to be included in the cohort.

`indicationLookbackWindow`
The window to look back prior to cohort index date to identify records of a indication condition. Only applicable if indicationConceptIds != ".

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

`studyEndDate` A calendar date specifying the maximum date that a cohort index date can appear. Date format is 'yyyymmdd'.

`exclusionConceptIds`
A list of CONCEPT_IDs used to restrict the cohorts, based on any descendant conditions/drugs/procedures occurring at least once anytime prior to the cohort index date.

`outcomeConditionTypeConceptIds`
A list of TYPE_CONCEPT_ID values that will restrict condition occurrences. Only applicable if outcomeTable = CONDITION_OCCURRENCE.

`exposureDatabaseSchema`
The name of the database schema that is the location where the exposure data used to define the exposure cohorts is available. If exposureTable = DRUG_ERA, exposureDatabaseSchema is not used by assumed to be cdmSchema. Requires read permissions to this database.

`exposureTable` The tablename that contains the exposure cohorts. If exposureTable <> DRUG_ERA, then expectation is exposureTable has format of COHORT table: cohort_concept_id, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.

`outcomeDatabaseSchema`
The name of the database schema that is the location where the data used to define the outcome cohorts is available. If 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_DEFINITION_ID, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.

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

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

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

`outcomeConceptIds`
: A list of CONCEPT_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.

Details

Based on the parameters, the treatment and comparator cohorts are constructed. Baseline covariates at or before the index date are extracted, as well as outcomes occurring on or after the index date. The treatment and comparator cohorts can be identified using the drug_era table, or through user-defined cohorts in a cohort table either inside the CDM instance or in a separate schema. Similarly, outcomes are identified using the condition_occurrence or condition_era table, or through user-defined cohorts in a cohort table either inside the CDM instance or in a separate schema. Covariates are automatically extracted from the appropriate tables within the CDM.This function calls the `getDbCovariates` and `getDbOutcomes` functions.

Important: The target and comparator drug must not be included in the covariates, including any descendant concepts. If the `targetId` and `comparatorId` arguments represent real concept IDs, you can set the `excludeDrugsFromCovariates` argument to TRUE and automatically the drugs and their descendants will be excluded from the covariates. However, if the `targetId` and `comparatorId` arguments do not represent concept IDs, you will need to manually add the drugs and descendants to the `excludedCovariateConceptIds` of the `covariateSettings` argument.

Value

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

outcomes An ffdf object listing the outcomes per person, including the time to event, and the outcome conncept ID. Outcomes are not yet filtered based on risk window, since this is done at a later stage.

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

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

exclude An ffdf object listing for each outcome concept ID the persons that need to be excluded from the analysis because of prior outcomes.

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

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

The generic `summary()` function has been implemented for this object.

getDbOutcomes *Get outcomes for persons in the cohorts*

Description

Gets the outcomes for the cohorts in the cohortMethodData object.

Usage

```
getDbOutcomes(connectionDetails = NULL, connection = NULL,
  cdmDatabaseSchema, oracleTempSchema = cdmDatabaseSchema, cohortMethodData,
  outcomeDatabaseSchema = cdmDatabaseSchema,
  outcomeTable = "condition_occurrence", outcomeIds = "",
  outcomeConditionTypeConceptIds = "", 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'.
cohortMethodData	An object of type cohortMethodData as generated using getDbCohortMethodData.
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_DEFINITION_ID, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.
outcomeConditionTypeConceptIds	A list of TYPE_CONCEPT_ID values that will restrict condition occurrences. Only applicable if outcomeTable = CONDITION_OCCURRENCE.
cdmVersion	Define the OMOP CDM version used: currently support "4" and "5".
oracleTempSchemaA	schema where temp tables can be created in Oracle.
outcomeConceptIds	A list of CONCEPT_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.

Details

If the connection parameter is specified, the cohorts are already assumed to be on the server in the appropriate temp table. Else, the temp table will be created by loading the cohorts from the cohortMethodData object to the server. This function can be used to add additional outcomes to an existing cohortMethodData object.

Value

The original cohortMethodData object with the new outcome data added.

---

getOutcomeModel *Get the outcome model*

Description

getOutcomeModel shows the full outcome model, so showing the betas of all variables included in the outcome model, not just the treatment variable.

Usage

```
getOutcomeModel(outcomeModel, cohortMethodData)
```

Arguments

outcomeModel	An object of type outcomeModel as generated using he createOutcomeMode function.
cohortMethodData	An object of type cohortMethodData as generated using getDbCohortMethodData.

Details

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

Examples

```
# todo
```

---

getPsModel *Get the propensity model*

Description

getPsModel shows the propensity score model

Usage

```
getPsModel(propensityScore, cohortMethodData)
```

Arguments

propensityScore
: The propensity scores as generated using the createPs function.

cohortMethodData
: An object of type cohortMethodData as generated using getDbCohortMethodData.

Details

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

Examples

```
# todo
```

---

grepCovariateNames *Extract covariate names*

Description

Extracts covariate names using a regular-expression.

Usage

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

Arguments

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

Details

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

Value

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

covariateId Numerical identifier for use in model fitting using these covariates

covariateName Text identifier

analysisId Analysis identifier

conceptId OMOP common data model concept identifier, or 0

loadCmAnalysisList *Load a list of cmAnalysis from file*

Description

Load a list of objects of type cmAnalysis from file. The file is in JSON format.

Usage

```
loadCmAnalysisList(file)
```

Arguments

file	The name of the file

Value

A list of objects of type cmAnalysis.

loadCohortMethodData *Load the cohort data from a folder*

Description

loadCohortMethodData loads an object of type cohortMethodData from a folder in the file system.

Usage

```
loadCohortMethodData(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 cohortMethodData.

Examples

```
# todo
```

loadDrugComparatorOutcomesList *Load a list of drugComparatorOutcomes from file*

Description

Load a list of objects of type drugComparatorOutcomes from file. The file is in JSON format.

Usage

```
loadDrugComparatorOutcomesList(file)
```

Arguments

file	The name of the file

Value

A list of objects of type drugComparatorOutcome.

matchOnPs *Match persons by propensity score*

Description

matchOnPs uses the provided propensity scores to match treated to comparator persons.

Usage

```
matchOnPs(data, caliper = 0.25, caliperScale = "standardized",
  maxRatio = 1, stratificationColumns = c())
```

Arguments

data	A data frame with the three columns described below.
caliper	The caliper for matching. A caliper is the distance which is acceptable for any match. Observations which are outside of the caliper are dropped. A caliper of 0 means no caliper is used.
caliperScale	The scale on which the caliper is defined. Two scales are supported: caliperScale = 'propensity score' or caliperScale = 'standardized'. On the standardized scale, the caliper is interpreted in standard deviations of the propensity score distribution.
maxRatio	The maximum number of persons int the comparator arm to be matched to each person in the treatment arm. A maxRatio of 0 means no maximum: all comparators will be assigned to a treated person.
stratificationColumns	Names or numbers of one or more columns in the data data.frame on which subjects should be stratified prior to matching. No persons will be matched with persons outside of the strata identified by the values in these columns.

Details

The data frame should have at least the following three columns:

rowId	(integer)	A unique identifier for each row (e.g. the person ID)
treatment	(integer)	Column indicating whether the person is in the treated (1) or comparator (0) group
propensityScore	(real)	Propensity score

This function implements the greedy variable-ratio matching algorithm described in Rassen et al (2012).

Value

Returns a date frame with the same columns as the input data plus one extra column: stratumId. Any rows that could not be matched are removed

References

Rassen JA, Shelat AA, Myers J, Glynn RJ, Rothman KJ, Schneeweiss S. (2012) One-to-many propensity score matching in cohort studies, Pharmacoepidemiology and Drug Safety, May, 21 Suppl 2:69-80.

Examples

```
rowId <- 1:5
treatment <- c(1, 0, 1, 0, 1)
propensityScore <- c(0, 0.1, 0.3, 0.4, 1)
age_group <- c(1, 1, 1, 1, 1)
data <- data.frame(rowId = rowId,
                   treatment = treatment,
                   propensityScore = propensityScore,
                   age_group = age_group)
result <- matchOnPs(data, caliper = 0, maxRatio = 1, stratificationColumns = "age_group")
```

---

matchOnPsAndCovariates

Match by propensity score as well as other covariates

Description

`matchOnPsAndCovariates` uses the provided propensity scores and a set of covariates to match treated to comparator persons.

Usage

```
matchOnPsAndCovariates(data, caliper = 0.25, caliperScale = "standardized",
  maxRatio = 1, cohortMethodData, covariateIds)
```

Arguments

`data`	A data frame with the three columns described below.
`caliper`	The caliper for matching. A caliper is the distance which is acceptable for any match. Observations which are outside of the caliper are dropped. A caliper of 0 means no caliper is used.
`caliperScale`	The scale on which the caliper is defined. Two scales are supported: `caliperScale = 'propensity score'` or `caliperScale = 'standardized'`. On the standardized scale, the caliper is interpreted in standard deviations of the propensity score distribution.
`maxRatio`	The maximum number of persons int the comparator arm to be matched to each person in the treatment arm. A maxRatio of 0 means no maximum: all comparators will be assigned to a treated person.
`cohortMethodData`	An object of type `cohortMethodData` as generated using `getDbCohortMethodData`.
`covariateIds`	One or more covariate IDs in the `cohortMethodData` object on which subjects should be also matched.

Details

The data frame should have at least the following three columns:

rowId	(integer)	A unique identifier for each row (e.g. the person ID)
treatment	(integer)	Column indicating whether the person is in the treated (1) or comparator (0) group
propensityScore	(real)	Propensity score

This function implements the greedy variable-ratio matching algorithm described in Rassen et al (2012).

Value

Returns a date frame with the same columns as the input data plus one extra column: stratumId. Any rows that could not be matched are removed

References

Rassen JA, Shelat AA, Myers J, Glynn RJ, Rothman KJ, Schneeweiss S. (2012) One-to-many propensity score matching in cohort studies, Pharmacoepidemiology and Drug Safety, May, 21 Suppl 2:69-80.

Examples

```
# todo
```

plotCovariateBalanceOfTopVariables

Plot variables with largest imbalance

Description

Create a plot showing those variables having the largest imbalance before matching, and those variables having the largest imbalance after matching. Requires running `computeCovariateBalance` first.

Usage

```
plotCovariateBalanceOfTopVariables(balance, n = 20, maxNameWidth = 100,
  fileName = NULL)
```

Arguments

`balance`	A data frame created by the `computeCovariateBalance` 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.

plotCovariateBalanceScatterPlot

Create a scatterplot of the covariate balance

Description

Create a scatterplot of the covariate balance, showing all variables with balance before and after matching on the x and y axis respectively. Requires running `computeCovariateBalance` first.

Usage

```
plotCovariateBalanceScatterPlot(balance, fileName = NULL)
```

Arguments

`balance`	A data frame created by the `computeCovariateBalance` funcion.
`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.

plotKaplanMeier *Plot the Kaplan-Meier curve*

Description

plotKaplanMeier creates the Kaplain-Meier survival plot

Usage

```
plotKaplanMeier(outcomeModel, censorMarks = FALSE,
  confidenceIntervals = TRUE, includeZero = TRUE, dataCutoff = 0.99,
  treatmentLabel = "Treated", comparatorLabel = "Comparator",
  fileName = NULL)
```

Arguments

outcomeModel	An object of type outcomeModel as generated using he fitOutcomeModel function.
censorMarks	Whether or not to include censor marks in the plot.
confidenceIntervals	Plot 95 percent confidence intervals?
includeZero	Should the y axis include zero, or only go down to the lowest observed survival?
dataCutoff	Fraction of the data (number censored) after which the graph will not be shown.
treatmentLabel	A label to us for the treated cohort.
comparatorLabel	A label to us for the comparator cohort.
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.

Examples

```
# todo
```

plotPs *Plot the propensity score distribution*

Description

plotPs shows the propensity (or preference) score distribution

Usage

```
plotPs(data, unfilteredData = NULL, scale = "preference",
  type = "density", binWidth = 0.05, fileName = NULL)
```

Arguments

`data`	A data frame with at least the two columns described below
`unfilteredData`	To be used when computing preference scores on data from which subjects have already been removed, e.g. through trimming and/or matching. This data frame should have the same structure as `data`.
`scale`	The scale of the graph. Two scales are supported: `scale = 'propensity'` or `scale = 'preference'`. The preference score scale is defined by Walker et al (2013).
`type`	Type of plot. Two possible values: `type = 'density'` or `type = 'histogram'`
`binWidth`	For histograms, the width of the bins
`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

The data frame should have a least the following two columns:

treatment	(integer)	Column indicating whether the person is in the treated (1) or comparator (0) group
propensityScore	(real)	Propensity score

Value

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

References

Walker AM, Patrick AR, Lauer MS, Hornbrook MC, Marin MG, Platt R, Roger VL, Stang P, and Schneeweiss S. (2013) A tool for assessing the feasibility of comparative effectiveness research, Comparative Effective Research, 3, 11-20

Examples

```
treatment <- rep(0:1, each = 100)
propensityScore <- c(rnorm(100, mean = 0.4, sd = 0.25), rnorm(100, mean = 0.6, sd = 0.25))
data <- data.frame(treatment = treatment, propensityScore = propensityScore)
data <- data[data$propensityScore > 0 & data$propensityScore < 1, ]
plotPs(data)
```

`recomputePsForFullData` *Recompute the PS for the full data set*

Description

Recompute the PS for the full data set

Usage

```
recomputePsForFullData(psSample, cohortMethodDataSample, cohortMethodData)
```

Arguments

psSample	The propensity scores as created on the sample data.
cohortMethodDataSample	The sample data on which the PS model was fitted.
cohortMethodData	The full data.

Details

After using the sampleCohorts or sampleComparator to reduce the population size, this function can be used to apply a propensity model fitted on the sample to the full data.

Value

A new propensity score object.

---

runCmAnalyses *Run a list of analyses*

Description

Run a list of analyses

Usage

```
runCmAnalyses(connectionDetails, cdmDatabaseSchema,
  oracleTempSchema = cdmDatabaseSchema,
  exposureDatabaseSchema = cdmDatabaseSchema, exposureTable = "drug_era",
  outcomeDatabaseSchema = cdmDatabaseSchema,
  outcomeTable = "condition_occurrence",
  outputFolder = "./CohortMethodOutput", cmAnalysisList,
  drugComparatorOutcomesList, refitPsForEveryOutcome = FALSE,
  underSampleComparatorToTreatedRatio = 0, getDbCohortMethodDataThreads = 1,
  createPsThreads = 1, psCvThreads = 1, trimMatchStratifyThreads = 1,
  computeCovarBalThreads = 1, fitOutcomeModelThreads = 1,
  outcomeCvThreads = 1)
```

Arguments

connectionDetails	An R object of type connectionDetails created using the function createConnectionDetails in the DatabaseConnector package.
cdmDatabaseSchema	The name of the database schema that contains the OMOP CDM instance. Requires read permissions to this database. On SQL Server, this should specifiy both the database and the schema, so for example 'cdm_instance.dbo'.
oracleTempSchema	For Oracle only: the name of the database schema where you want all temporary tables to be managed. Requires create/insert permissions to this database.

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

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

`outcomeDatabaseSchema`
: The name of the database schema that is the location where the data used to define the outcome cohorts is available. If 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_DEFINITION_ID, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.

`outputFolder`
: Name of the folder where all the outputs will written to.

`cmAnalysisList`
: A list of objects of type `cmAnalysis` as created using the `createCmAnalysis` function.

`drugComparatorOutcomesList`
: A list of objects of type `drugComparatorOutcomes` as created using the `createdrugComparatorOutc` function.

`refitPsForEveryOutcome`
: Should the propensity model be fitted for every outcome (i.e. after people who already had the outcome are removed)? If false, a single propensity model will be fitted, and people who had the outcome previously will be removed afterwards.

`underSampleComparatorToTreatedRatio`
: If the comparator group size exceeds the treated group size by this factor, the comparator group will be down-sampled before fitting the PS model. This can be useful when the comparator group is extremely large.

`getDbCohortMethodDataThreads`
: The number of parallel threads to use for building the cohortMethod data objects.

`createPsThreads`
: The number of parallel threads to use for fitting the propensity models.

`psCvThreads`
: The number of parallel threads to use for the cross- validation when estimating the hyperparameter for the propensity model. Note that the total number of CV threads at one time could be 'createPsThreads * psCvThreads'.

`trimMatchStratifyThreads`
: The number of parallel threads to use for trimming, matching and stratifying.

`computeCovarBalThreads`
: The number of parallel threads to use for computing the covariate balance.

`fitOutcomeModelThreads`
: The number of parallel threads to use for fitting the outcome models.

`outcomeCvThreads`
: The number of parallel threads to use for the cross- validation when estimating the hyperparameter for the outcome model. Note that the total number of CV threads at one time could be 'fitOutcomeModelThreads * outcomeCvThreads'.

Details

Run a list of analyses for the drug-comparator-outcomes of interest. This function will run all specified analyses against all hypotheses of interest, meaning that the total number of outcome models is 'length(cmAnalysisList) * length(drugComparatorOutcomesList)' (if all analyses specify an outcome model should be fitted). When you provide several analyses it will determine whether any of the analyses have anything in common, and will take advantage of this fact. For example, if we specify several analyses that only differ in the way the outcome model is fitted, then this function will extract the data and fit the propensity model only once, and re-use this in all the analysis.

---

sampleCohorts *Sample the target or comparator group down*

Description

Sample the target or comparator group down

Usage

```
sampleCohorts(cohortMethodData, treatedSampleSize = 10000,
  comparatorSampleSize = 20000)
```

Arguments

`cohortMethodData`
: The original cohortMethodData.

`treatedSampleSize`
: The sampe size for the treated cohort.

`comparatorSampleSize`
: The sample size for the comparator cohort.

Details

When the target and/or comparator cohorts are extremely large, it may be more efficient to only use a sample to fit the propensity model. This function creates a new cohortMethodData object with sampled populations.

Value

An object of type `cohortMethodData` with the sampled populations.

sampleComparator *Sample the comparator group down*

Description

Sample the comparator group down

Usage

```
sampleComparator(cohortMethodData, comparatorToTreatedRatio = 2)
```

Arguments

cohortMethodData	The original cohortMethodData.
comparatorToTreatedRatio	The ratio between comparator and treated group.

Details

When the comparator group is extremely large, it may be more efficient to only use a sample to fit the propensity model. This function creates a new cohortMethodData object where to comparator group is sampled down to a size relative to the treated group.

Value

An object of type cohortMethodData with the sampled populations.

saveCmAnalysisList *Save a list of cmAnalysis to file*

Description

Write a list of objects of type cmAnalysis to file. The file is in JSON format.

Usage

```
saveCmAnalysisList(cmAnalysisList, file)
```

Arguments

cmAnalysisList	The cmAnalysis list to be written to file
file	The name of the file where the results will be written

saveCohortMethodData *Save the cohort data to folder*

Description

saveCohortMethodData saves an object of type cohortMethodData to folder.

Usage

```
saveCohortMethodData(cohortMethodData, file)
```

Arguments

cohortMethodData	An object of type cohortMethodData as generated using getDbCohortMethodData.
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
```

saveDrugComparatorOutcomesList *Save a list of drugComparatorOutcome to file*

Description

Write a list of objects of type drugComparatorOutcomes to file. The file is in JSON format.

Usage

```
saveDrugComparatorOutcomesList(drugComparatorOutcomesList, file)
```

Arguments

drugComparatorOutcomesList	The drugComparatorOutcomes list to be written to file
file	The name of the file where the results will be written

simulateCohortMethodData *Generate simulated data*

Description

simulateCohortMethodData creates a cohortMethodData object with simulated data.

Usage

```
simulateCohortMethodData(cohortDataSimulationProfile, n = 10000)
```

Arguments

cohortDataSimulationProfile	An object of type cohortDataSimulationProfile as generated using the createCohortMethodDataSimulationProfile 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 cohortMethodData.

stratifyByPs *Stratify persons by propensity score*

Description

stratifyByPs uses the provided propensity scores to stratify persons. Additional stratification variables for stratifications can also be used.

Usage

```
stratifyByPs(data, numberOfStrata = 5, stratificationColumns = c())
```

Arguments

data	A data frame with the three columns described below
numberOfStrata	How many strata? The boundaries of the strata are automatically defined to contain equal numbers of treated persons.
stratificationColumns	Names of one or more columns in the data data.frame on which subjects should also be stratified in addition to stratification on propensity score.

Details

The data frame should have the following three columns:

rowId	(integer)	A unique identifier for each row (e.g. the person ID)
treatment	(integer)	Column indicating whether the person is in the treated (1) or comparator (0) group
propensityScore	(real)	Propensity score

Value

Returns a date frame with the same columns as the input data plus one extra column: stratumId.

Examples

```
rowId <- 1:200
treatment <- rep(0:1, each = 100)
propensityScore <- c(runif(100, min = 0, max = 1), runif(100, min = 0, max = 1))
data <- data.frame(rowId = rowId, treatment = treatment, propensityScore = propensityScore)
result <- stratifyByPs(data, 5)
```

---

stratifyByPsAndCovariates

Stratify persons by propensity score and other covariates

Description

`stratifyByPsAndCovariates` uses the provided propensity scores and covariatesto stratify persons.

Usage

```
stratifyByPsAndCovariates(data, numberOfStrata = 5, cohortMethodData,
  covariateIds)
```

Arguments

`data`	A data frame with the three columns described below
`numberOfStrata`	Into how many strata should the propensity score be divided? The boundaries of the strata are automatically defined to contain equal numbers of treated persons.
`cohortMethodData`	An object of type `cohortMethodData` as generated using `getDbCohortMethodData`.
`covariateIds`	One or more covariate IDs in the `cohortMethodData` object on which subjects should also be stratified.

Details

The data frame should have the following three columns:

rowId	(integer)	A unique identifier for each row (e.g. the person ID)
treatment	(integer)	Column indicating whether the person is in the treated (1) or comparator (0) group
propensityScore	(real)	Propensity score

Value

Returns a date frame with the same columns as the input data plus one extra column: stratumId.

Examples

```
# todo
```

---

summarizeAnalyses *Create a summary report of the analyses*

Description

Create a summary report of the analyses

Usage

```
summarizeAnalyses(outcomeReference)
```

Arguments

outcomeReference	A data.frame as created by the runAnalyses function.

---

trimByPs *Trim persons by propensity score*

Description

trimByPs uses the provided propensity scores to trim subjects with extreme scores.

Usage

```
trimByPs(data, trimFraction = 0.05)
```

Arguments

data	A data frame with the three columns described below
trimFraction	This fraction will be removed from each treatment group. In the treatment group, persons with the highest propensity scores will be removed, in the comparator group person with the lowest scores will be removed.

Details

The data frame should have the following three columns:

rowId	(integer)	A unique identifier for each row (e.g. the person ID)
treatment	(integer)	Column indicating whether the person is in the treated (1) or comparator (0) group
propensityScore	(real)	Propensity score

Value

Returns a date frame with the same three columns as the input.

Examples

```
rowId <- 1:2000
treatment <- rep(0:1, each = 1000)
propensityScore <- c(runif(1000, min = 0, max = 1), runif(1000, min = 0, max = 1))
data <- data.frame(rowId = rowId, treatment = treatment, propensityScore = propensityScore)
result <- trimByPs(data, 0.05)
```

---

trimByPsToEquipoise *Keep only persons in clinical equipoise*

Description

`trimByPsToEquipoise` uses the preference score to trim subjects that are not in clinical equipoise

Usage

```
trimByPsToEquipoise(data, bounds = c(0.25, 0.75))
```

Arguments

`data`	A data frame with at least the three columns described below
`bounds`	The upper and lower bound on the preference score for keeping persons

Details

The data frame should have the following three columns:

rowId	(integer)	A unique identifier for each row (e.g. the person ID)
treatment	(integer)	Column indicating whether the person is in the treated (1) or comparator (0) group
propensityScore	(real)	Propensity score

Value

Returns a date frame with the same three columns as the input.

References

Walker AM, Patrick AR, Lauer MS, Hornbrook MC, Marin MG, Platt R, Roger VL, Stang P, and Schneeweiss S. (2013) A tool for assessing the feasibility of comparative effectiveness research, Comparative Effective Research, 3, 11-20

Examples

```
rowId <- 1:2000
treatment <- rep(0:1, each = 1000)
propensityScore <- c(runif(1000, min = 0, max = 1), runif(1000, min = 0, max = 1))
data <- data.frame(rowId = rowId, treatment = treatment, propensityScore = propensityScore)
result <- trimByPsToEquipoise(data)
```

---

vignetteBalance *Balance data for the vignette*

Description

Balance data for the vignette

Usage

```
data(vignetteBalance)
```

---

vignetteOutcomeModel1 *Outcome data for the vignette*

Description

Outcome data for the vignette

Usage

```
data(vignetteOutcomeModel1)
```

---

vignetteOutcomeModel2 *Outcome data for the vignette*

Description

Outcome data for the vignette

Usage

```
data(vignetteOutcomeModel2)
```

vignetteOutcomeModel3 *Outcome data for the vignette*

Description

Outcome data for the vignette

Usage

```
data(vignetteOutcomeModel3)
```

vignettePs *Propensity scores for the vignette*

Description

Propensity scores for the vignette

Usage

```
data(vignettePs)
```

Index