Package ‘CohortMethod’

February 1, 2024

Type Package

Title New-User Cohort Method with Large Scale Propensity and Outcome Models

Version 5.2.1

Date 2024-02-01

Maintainer Martijn Schuemie <schuemie@ohdsi.org>

Description Functions for performing new-user cohort studies
in an observational database in the OMOP Common Data Model. Can extract the
necessary data from a database and use 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,
(variable and fixed ratio) matching and weighting by 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. Also included are Kaplan-Meier plots that can adjust for
the stratification or matching.

License Apache License 2.0

VignetteBuilder knitr

URL https://ohdsi.github.io/CohortMethod, https://github.com/OHDSI/CohortMethod

BugReports https://github.com/OHDSI/CohortMethod/issues

Depends R (>= 3.6.0),
DatabaseConnector (>= 6.0.0),
Cyclops (>= 3.1.2),
FeatureExtraction (>= 3.0.0),
Andromeda (>= 0.6.3)

Imports methods,
ggplot2,
gridExtra,
grid,
readr,
plyr,
dplyr,
rlang,
cli,
pillar,

Rcpp (>= 0.11.2),
SqlRender (>= 1.12.0),
survival,
ParallelLogger (>= 3.0.1),
bit64,
checkmate,
EmpiricalCalibration,
zip

Suggests testthat,
pROC,
knitr,
rmarkdown,
Eunomia,
withr,
R.utils,
RSQLite,
ResultModelManager,
ShinyAppBuilder (>= 1.2.0),
markdown,
remotes

Remotes ohdsi/FeatureExtraction,
ohdsi/Eunomia,
ohdsi/ResultModelManager,
ohdsi/ShinyAppBuilder

LinkingTo Rcpp

NeedsCompilation yes

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

Encoding UTF-8

R topics documented:

`adjustedKm` *Compute a weight-adjusted Kaplan-Meier curve*

Description

Compute a weight-adjusted Kaplan-Meier curve

Usage

```
adjustedKm(weight, time, y)
```

Arguments

`weight`	Vector of observation weights
`time`	Vector of event times
`y`	Vector outcomes (0 indicates censoring, 1 indicates event-of-interest)

`checkCmInstallation` *Check is CohortMethod and its dependencies are correctly installed*

Description

Check is CohortMethod and its dependencies are correctly installed

Usage

```
checkCmInstallation(connectionDetails)
```

Arguments

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

Details

This function checks whether CohortMethod and its dependencies are correctly installed. This will check the database connectivity, large scale regression engine (Cyclops), and large data object handling (ff).

`CohortMethodData-class`
Cohort Method Data

Description

`CohortMethodData` is an S4 class that inherits from CoviarateData, which in turn inherits from Andromeda. It contains information on the cohorts, their outcomes, and baseline covariates. Information about multiple outcomes can be captured at once for efficiency reasons.

A `CohortMethodData` is typically created using `getDbCohortMethodData()`, can only be saved using `saveCohortMethodData()`, and loaded using `loadCohortMethodData()`.

Usage

```
## S4 method for signature 'CohortMethodData'
show(object)

## S4 method for signature 'CohortMethodData'
summary(object)
```

Arguments

`object`	An object of type `CohortMethodData`.

`cohortMethodDataSimulationProfile`
A simulation profile

Description

A simulation profile

Usage

```
data(cohortMethodDataSimulationProfile)
```

`computeCovariateBalance`
Compute covariate balance before and after PS adjustment

Description

For every covariate, prevalence in treatment and comparator groups before and after matching/trimming/weighting are computed. When variable ratio matching was used the balance score will be corrected according the method described in Austin et al (2008).

Usage

```
computeCovariateBalance(
  population,
  cohortMethodData,
  subgroupCovariateId = NULL,
  maxCohortSize = 250000,
  covariateFilter = NULL
)
```

Arguments

`population`	A data frame containing the people that are remaining after PS adjustment.
`cohortMethodData`	An object of type CohortMethodData as generated using `getDbCohortMethodData()`.
`subgroupCovariateId`	Optional: a covariate ID of a binary covariate that indicates a subgroup of interest. Both the before and after populations will be restricted to this subgroup before computing covariate balance.
`maxCohortSize`	If the target or comparator cohort are larger than this number, they will be downsampled before computing covariate balance to save time. Setting this number to 0 means no downsampling will be applied.
`covariateFilter`	Determines the covariates for which to compute covariate balance. Either a vector of covariate IDs, or a table 1 specifications object as generated for example using `FeatureExtraction::getDefaultTable1Specifications()`. If `covariateFilter = NULL`, balance will be computed for all variables found in the data.

Details

The population data frame should have the following three columns:

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

Value

Returns a tibble describing the covariate balance before and after PS adjustment, with one row per covariate, with the same data as the `covariateRef` table in the `CohortMethodData` object, and the following additional columns:

- beforeMatchingMeanTarget: The (weighted) mean value in the target before PS adjustment.
- beforeMatchingMeanComparator: The (weighted) mean value in the comparator before PS adjustment.
- beforeMatchingSumTarget: The (weighted) sum value in the target before PS adjustment.
- beforeMatchingSumComparator: The (weighted) sum value in the comparator before PS adjustment.
- beforeMatchingSdTarget: The standard deviation of the value in the target before PS adjustment.

- beforeMatchingSdComparator: The standard deviation of the value in the comparator before PS adjustment.
- beforeMatchingMean: The mean of the value across target and comparator before PS adjustment.
- beforeMatchingSd: The standard deviation of the value across target and comparator before PS adjustment.
- afterMatchingMeanTarget: The (weighted) mean value in the target after PS adjustment.
- afterMatchingMeanComparator: The (weighted) mean value in the comparator after PS adjustment.
- afterMatchingSumTarget: The (weighted) sum value in the target after PS adjustment.
- afterMatchingSumComparator: The (weighted) sum value in the comparator after PS adjustment.
- afterMatchingSdTarget: The standard deviation of the value in the target after PS adjustment.
- afterMatchingSdComparator: The standard deviation of the value in the comparator after PS adjustment.
- afterMatchingMean: The mean of the value across target and comparator after PS adjustment.
- afterMatchingSd: The standard deviation of the value across target and comparator after PS adjustment.
- beforeMatchingStdDiff: The standardized difference of means when comparing the target to the comparator before PS adjustment.
- afterMatchingStdDiff: The standardized difference of means when comparing the target to the comparator after PS adjustment.
- targetStdDiff: The standardized difference of means when comparing the target before PS adjustment to the target after PS adjustment.
- comparatorStdDiff: The standardized difference of means when comparing the comparator before PS adjustment to the comparator after PS adjustment. -targetComparatorStdDiff: The standardized difference of means when comparing the entire population before PS adjustment to the entire population after PS adjustment.

The 'beforeMatchingStdDiff' and 'afterMatchingStdDiff' columns inform on the balance: are the target and comparator sufficiently similar in terms of baseline covariates to allow for valid causal estimation?

The 'targetStdDiff', 'comparatorStdDiff', and 'targetComparatorStdDiff' columns inform on the generalizability: are the cohorts after PS adjustment sufficiently similar to the cohorts before adjustment to allow generalizing the findings to the original cohorts?

References

Austin, P.C. (2008) Assessing balance in measured baseline covariates when using many-to-one matching on the propensity-score. Pharmacoepidemiology and Drug Safety, 17: 1218-1225.

`computeEquipoise` *Compute fraction in equipoise*

Description

Compute fraction in equipoise

Usage

```
computeEquipoise(data, equipoiseBounds = c(0.3, 0.7))
```

Arguments

`data`	A data frame with at least the two columns described below.
`equipoiseBounds`	The bounds on the preference score to determine whether a subject is in equipoise.

Details

Computes the fraction of the population (the union of the target and comparator cohorts) who are in clinical equipoise (i.e. who had a reasonable chance of receiving either target or comparator, based on the baseline characteristics).

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

- treatment (integer): Column indicating whether the person is in the target (1) or comparator (0) group
- propensityScore (numeric): Propensity score

Value

A numeric value (fraction in equipoise) between 0 and 1.

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

`computeMdrr` *Compute the minimum detectable relative risk*

Description

Compute the minimum detectable relative risk

Usage

```
computeMdrr(
  population,
  alpha = 0.05,
  power = 0.8,
  twoSided = TRUE,
  modelType = "cox"
)
```

Arguments

`population`	A data frame describing the study population as created using the `createStudyPopulation` function. This should at least have these columns: personSeqId, treatment, outcomeCount, timeAtRisk.
`alpha`	Type I error.
`power`	1 - beta, where beta is the type II error.
`twoSided`	Consider a two-sided test?
`modelType`	The type of outcome model that will be used. Possible values are "logistic", "poisson", or "cox". Currently only "cox" is supported.

Details

Compute the minimum detectable relative risk (MDRR) and expected standard error (SE) for a given study population, using the actual observed sample size and number of outcomes. Currently, only computations for Cox models are implemented. For Cox model, the computations by Schoenfeld (1983) is used.

Value

A data frame with the MDRR and some counts.

References

Schoenfeld DA (1983) Sample-size formula for the proportional-hazards regression model, Biometrics, 39(3), 499-503

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

Description

Compute the area under the ROC curve of the propensity score.

Usage

```
computePsAuc(data, confidenceIntervals = FALSE, maxRows = 1e+05)
```

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)
`maxRows`	Maximum number of rows to use. If the number of rows is larger, a random sample will be taken. This can increase speed, with minor cost to precision. Set to 0 to use all data.

Details

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

- treatment (integer): Column indicating whether the person is in the target (1) or comparator (0) group.
- propensityScore (numeric): Propensity score.

Value

A tibble 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)
```

`createCmAnalysis` *Create a CohortMethod analysis specification*

Description

Create a CohortMethod analysis specification

Usage

```
createCmAnalysis(
  analysisId = 1,
  description = "",
  getDbCohortMethodDataArgs,
  createStudyPopArgs,
  createPsArgs = NULL,
  trimByPsArgs = NULL,
  trimByPsToEquipoiseArgs = NULL,
  trimByIptwArgs = NULL,
  truncateIptwArgs = NULL,
  matchOnPsArgs = NULL,
  matchOnPsAndCovariatesArgs = NULL,
  stratifyByPsArgs = NULL,
  stratifyByPsAndCovariatesArgs = NULL,
  computeSharedCovariateBalanceArgs = NULL,
  computeCovariateBalanceArgs = NULL,
  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.
`getDbCohortMethodDataArgs`	An object representing the arguments to be used when calling the `getDbCohortMethodData()` function.
`createStudyPopArgs`	An object representing the arguments to be used when calling the `createStudyPopulation()` function.
`createPsArgs`	An object representing the arguments to be used when calling the `createPs()` function.
`trimByPsArgs`	An object representing the arguments to be used when calling the `trimByPs()` function.
`trimByPsToEquipoiseArgs`	An object representing the arguments to be used when calling the `trimByPsToEquipoise()` function.
`trimByIptwArgs`	An object representing the arguments to be used when calling the `trimByIptw()` function.
`truncateIptwArgs`	An object representing the arguments to be used when calling the `truncateIptw()` function.
`matchOnPsArgs`	An object representing the arguments to be used when calling the `matchOnPs()` function.
`matchOnPsAndCovariatesArgs`	An object representing the arguments to be used when calling the `matchOnPsAndCovariates()` function.
`stratifyByPsArgs`	An object representing the arguments to be used when calling the `stratifyByPs()` function.
`stratifyByPsAndCovariatesArgs`	An object representing the arguments to be used when calling the `stratifyByPsAndCovariates()` function.
`computeSharedCovariateBalanceArgs`	An object representing the arguments to be used when calling the `computeCovariateBalance()` function per target-comparator-analysis.
`computeCovariateBalanceArgs`	An object representing the arguments to be used when calling the `computeCovariateBalance()` function per target-comparator-outcome-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.

Providing a NULL value for any of the argument applies the corresponding step will not be executed. For example, if `createPsArgs = NULL`, no propensity scores will be computed.

---

createCmDiagnosticThresholds

Create CohortMethod diagnostics thresholds

Description

Threshold used when calling `exportToCsv()` to determine if we pass or fail diagnostics.

Usage

```
createCmDiagnosticThresholds(
  mdrrThreshold = 10,
  easeThreshold = 0.25,
  sdmThreshold = 0.1,
  equipoiseThreshold = 0.2,
  attritionFractionThreshold = NULL,
  generalizabilitySdmThreshold = 1
)
```

Arguments

`mdrrThreshold`	What is the maximum allowed minimum detectable relative risk (MDRR)?
`easeThreshold`	What is the maximum allowed expected absolute systematic error (EASE).
`sdmThreshold`	What is the maximum allowed standardized difference of mean (SDM)? If any covariate has an SDM exceeding this threshold, the diagnostic will fail.
`equipoiseThreshold`	What is the minimum required equipoise?
`attritionFractionThreshold`	DEPRECATED. See `generalizabilitySdmThreshold` instead.
`generalizabilitySdmThreshold`	What is the maximum allowed standardized difference of mean (SDM)when comparing the population before and after PS adjustments? If the SDM is greater than this value, the diagnostic will fail.

Value

An object of type `CmDiagnosticThresholds`.

createCmTable1 *Create a table 1*

Description

Creates a formatted table of cohort characteristics, to be included in publications or reports.

Usage

```
createCmTable1(
  balance,
  specifications = getDefaultCmTable1Specifications(),
  beforeTargetPopSize = NULL,
  beforeComparatorPopSize = NULL,
  afterTargetPopSize = NULL,
  afterComparatorPopSize = NULL,
  beforeLabel = "Before matching",
  afterLabel = "After matching",
  targetLabel = "Target",
  comparatorLabel = "Comparator",
  percentDigits = 1,
  stdDiffDigits = 2
)
```

Arguments

`balance`	A data frame created by the `computeCovariateBalance` function.
`specifications`	Specifications of which covariates to display, and how.
`beforeTargetPopSize`	The number of people in the target cohort before matching/stratification/trimming, to mention in the table header. If not provide, no number will be included in the header.
`beforeComparatorPopSize`	The number of people in the comparator cohort before matching/stratification/trimming, to mention in the table header. If not provide, no number will be included in the header.
`afterTargetPopSize`	The number of people in the target cohort after matching/stratification/trimming, to mention in the table header. If not provide, no number will be included in the header.
`afterComparatorPopSize`	The number of people in the comparator cohort after matching/stratification/trimming, to mention in the table header. If not provide, no number will be included in the header.
`beforeLabel`	Label for identifying columns before matching / stratification / trimming.
`afterLabel`	Label for identifying columns after matching / stratification / trimming.
`targetLabel`	Label for identifying columns of the target cohort.
`comparatorLabel`	Label for identifying columns of the comparator cohort.
`percentDigits`	Number of digits to be used for percentages.
`stdDiffDigits`	Number of digits to be used for the standardized differences.

Value

A data frame with the formatted table 1.

`createCohortMethodDataSimulationProfile` *Create simulation profile*

Description

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`.

`createComputeCovariateBalanceArgs` *Create a parameter object for the function computeCovariateBalance*

Description

Create a parameter object for the function computeCovariateBalance

Usage

```
createComputeCovariateBalanceArgs(
  subgroupCovariateId = NULL,
  maxCohortSize = 250000,
  covariateFilter = NULL
)
```

Arguments

`subgroupCovariateId`	Optional: a covariate ID of a binary covariate that indicates a subgroup of interest. Both the before and after populations will be restricted to this subgroup before computing covariate balance.
`maxCohortSize`	If the target or comparator cohort are larger than this number, they will be downsampled before computing covariate balance to save time. Setting this number to 0 means no downsampling will be applied.
`covariateFilter`	Determines the covariates for which to compute covariate balance. Either a vector of covariate IDs, or a table 1 specifications object as generated for example using FeatureExtraction::getDefaultTable1Specifications(). If covariateFilter = NULL, balance will be computed for all variables found in the data.

Details

Create an object defining the parameter values.

`createCreatePsArgs` *Create a parameter object for the function createPs*

Description

Create a parameter object for the function createPs

Usage

```
createCreatePsArgs(
  excludeCovariateIds = c(),
  includeCovariateIds = c(),
  maxCohortSizeForFitting = 250000,
  errorOnHighCorrelation = TRUE,
  stopOnError = TRUE,
  prior = createPrior("laplace", exclude = c(0), useCrossValidation = TRUE),
  control = createControl(noiseLevel = "silent", cvType = "auto", seed = 1,
   resetCoefficients = TRUE, tolerance = 2e-07, cvRepetitions = 10, startingVariance =
    0.01),
  estimator = "att"
)
```

Arguments

`excludeCovariateIds`	Exclude these covariates from the propensity model.
`includeCovariateIds`	Include only these covariates in the propensity model.
`maxCohortSizeForFitting`	If the target or comparator cohort are larger than this number, they will be downsampled before fitting the propensity model. The model will be used to compute propensity scores for all subjects. The purpose of the sampling is to gain speed. Setting this number to 0 means no downsampling will be applied.

`errorOnHighCorrelation`	If true, the function will test each covariate for correlation with the treatment assignment. If any covariate has an unusually high correlation (either positive or negative), this will throw and error.
`stopOnError`	If an error occur, should the function stop? Else, the two cohorts will be assumed to be perfectly separable.
`prior`	The prior used to fit the model. See Cyclops::createPrior() for details.
`control`	The control object used to control the cross-validation used to determine the hyperparameters of the prior (if applicable). See Cyclops::createControl() for details.
`estimator`	The type of estimator for the IPTW. Options are estimator = "ate" for the average treatment effect, estimator = "att" for the average treatment effect in the treated, and estimator = "ato" for the average treatment effect in the overlap population.

Details

Create an object defining the parameter values.

`createCreateStudyPopulationArgs`
Create a parameter object for the function createStudyPopulation

---

Description

Create a parameter object for the function createStudyPopulation

Usage

```
createCreateStudyPopulationArgs(
  firstExposureOnly = FALSE,
  restrictToCommonPeriod = FALSE,
  washoutPeriod = 0,
  removeDuplicateSubjects = "keep all",
  removeSubjectsWithPriorOutcome = TRUE,
  priorOutcomeLookback = 99999,
  minDaysAtRisk = 1,
  maxDaysAtRisk = 99999,
  riskWindowStart = 0,
  startAnchor = "cohort start",
  riskWindowEnd = 0,
  endAnchor = "cohort end",
  censorAtNewRiskWindow = FALSE
)
```

Arguments

`firstExposureOnly`	Should only the first exposure per subject be included?
`restrictToCommonPeriod`	Restrict the analysis to the period when both exposures are observed?

`washoutPeriod`	The minimum required continuous observation time prior to index date for a person to be included in the cohort.
`removeDuplicateSubjects`	Remove subjects that are in both the target and comparator cohort? See details for allowed values.
`removeSubjectsWithPriorOutcome`	Remove subjects that have the outcome prior to the risk window start?
`priorOutcomeLookback`	How many days should we look back when identifying prior outcomes?
`minDaysAtRisk`	The minimum required number of days at risk. Risk windows with fewer days than this number are removed from the analysis.
`maxDaysAtRisk`	The maximum allowed number of days at risk. Risk windows that are longer will be truncated to this number of days.
`riskWindowStart`	The start of the risk window (in days) relative to the startAnchor.
`startAnchor`	The anchor point for the start of the risk window. Can be "cohort start" or "cohort end".
`riskWindowEnd`	The end of the risk window (in days) relative to the endAnchor.
`endAnchor`	The anchor point for the end of the risk window. Can be "cohort start" or "cohort end".
`censorAtNewRiskWindow`	If a subject is in multiple cohorts, should time-at-risk be censored when the new time-at-risk starts to prevent overlap?

Details

Create an object defining the parameter values.

`createDefaultMultiThreadingSettings`

Create default CohortMethod multi-threading settings

Description

Create CohortMethod multi-threading settings based on the maximum number of cores to be used.

Usage

```
createDefaultMultiThreadingSettings(maxCores)
```

Arguments

`maxCores`	Maximum number of CPU cores to use.

Value

An object of type `CmMultiThreadingSettings`.

See Also

createMultiThreadingSettings()

Examples

```
settings <- createDefaultMultiThreadingSettings(10)
```

---

createFitOutcomeModelArgs

Create a parameter object for the function fitOutcomeModel

Description

Create a parameter object for the function fitOutcomeModel

Usage

```
createFitOutcomeModelArgs(
  modelType = "logistic",
  stratified = FALSE,
  useCovariates = FALSE,
  inversePtWeighting = FALSE,
  interactionCovariateIds = c(),
  excludeCovariateIds = c(),
  includeCovariateIds = c(),
  profileGrid = NULL,
  profileBounds = c(log(0.1), log(10)),
  prior = createPrior("laplace", useCrossValidation = TRUE),
  control = createControl(cvType = "auto", seed = 1, resetCoefficients = TRUE,
   startingVariance = 0.01, tolerance = 2e-07, cvRepetitions = 10, noiseLevel = "quiet")
)
```

Arguments

`modelType`	The type of outcome model that will be used. Possible values are "logistic", "poisson", or "cox".
`stratified`	Should the regression be conditioned on the strata defined in the population object (e.g. by matching or stratifying on propensity scores)?
`useCovariates`	Whether to use the covariates in the cohortMethodData object in the outcome model.
`inversePtWeighting`	Use inverse probability of treatment weighting (IPTW)
`interactionCovariateIds`	An optional vector of covariate IDs to use to estimate interactions with the main treatment effect.
`excludeCovariateIds`	Exclude these covariates from the outcome model.
`includeCovariateIds`	Include only these covariates in the outcome model.

`profileGrid`	A one-dimensional grid of points on the log(relative risk) scale where the likelihood for coefficient of variables is sampled. See details.
`profileBounds`	The bounds (on the log relative risk scale) for the adaptive sampling of the likelihood function. See details.
`prior`	The prior used to fit the model. See Cyclops::createPrior() for details.
`control`	The control object used to control the cross-validation used to determine the hyperparameters of the prior (if applicable). See Cyclops::createControl() 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(
  studyStartDate = "",
  studyEndDate = "",
  firstExposureOnly = FALSE,
  removeDuplicateSubjects = "keep all",
  restrictToCommonPeriod = FALSE,
  washoutPeriod = 0,
  maxCohortSize = 0,
  covariateSettings
)
```

Arguments

`studyStartDate`	A calendar date specifying the minimum date that a cohort index date can appear. Date format is 'yyyymmdd'.
`studyEndDate`	A calendar date specifying the maximum date that a cohort index date can appear. Date format is 'yyyymmdd'. Important: the study end data is also used to truncate risk windows, meaning no outcomes beyond the study end date will be considered.
`firstExposureOnly`	Should only the first exposure per subject be included? Note that this is typically done in the createStudyPopulation() function, but can already be done here for efficiency reasons.
`removeDuplicateSubjects`	Remove subjects that are in both the target and comparator cohort? See details for allowed values.Note that this is typically done in the createStudyPopulation function, but can already be done here for efficiency reasons.

`restrictToCommonPeriod`	Restrict the analysis to the period when both treatments are observed?
`washoutPeriod`	The minimum required continuous observation time prior to index date for a person to be included in the cohort. Note that this is typically done in the createStudyPopulation function, but can already be done here for efficiency reasons.
`maxCohortSize`	If either the target or the comparator cohort is larger than this number it will be sampled to this size. maxCohortSize = 0 indicates no maximum size.
`covariateSettings`	An object of type covariateSettings as created using the FeatureExtraction::createCovariateSettings() function.

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 = 0.2,
  caliperScale = "standardized logit",
  maxRatio = 1,
  allowReverseMatch = FALSE,
  covariateIds
)
```

Arguments

`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. Three scales are supported: caliperScale = 'propensity score', caliperScale = 'standardized', or caliperScale = 'standardized logit'. On the standardized scale, the caliper is interpreted in standard deviations of the propensity score distribution. 'standardized logit' is similar, except that the propensity score is transformed to the logit scale because the PS is more likely to be normally distributed on that scale (Austin, 2011).
`maxRatio`	The maximum number of persons in 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 target person.
`allowReverseMatch`	Allows n-to-1 matching if target arm is larger
`covariateIds`	One or more covariate IDs in the cohortMethodData object on which subjects 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 = 0.2,
  caliperScale = "standardized logit",
  maxRatio = 1,
  allowReverseMatch = FALSE,
  stratificationColumns = c()
)
```

Arguments

`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. Three scales are supported: caliperScale = 'propensity score', caliperScale = 'standardized', or caliperScale = 'standardized logit'. On the standardized scale, the caliper is interpreted in standard deviations of the propensity score distribution. 'standardized logit' is similar, except that the propensity score is transformed to the logit scale because the PS is more likely to be normally distributed on that scale (Austin, 2011).
`maxRatio`	The maximum number of persons in 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 target person.
`allowReverseMatch`	Allows n-to-1 matching if target arm is larger
`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

Create an object defining the parameter values.

`createMultiThreadingSettings`
Create CohortMethod multi-threading settings

Description

Create CohortMethod multi-threading settings

Usage

```
createMultiThreadingSettings(
  getDbCohortMethodDataThreads = 1,
  createPsThreads = 1,
  psCvThreads = 1,
  createStudyPopThreads = 1,
  trimMatchStratifyThreads = 1,
  computeSharedBalanceThreads = 1,
  computeBalanceThreads = 1,
  prefilterCovariatesThreads = 1,
  fitOutcomeModelThreads = 1,
  outcomeCvThreads = 1,
  calibrationThreads = 1
)
```

Arguments

`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`.
`createStudyPopThreads`	The number of parallel threads to use for creating the study population.
`trimMatchStratifyThreads`	The number of parallel threads to use for trimming, matching and stratifying.
`computeSharedBalanceThreads`	The number of parallel threads to use for computing shared covariate balance.
`computeBalanceThreads`	The number of parallel threads to use for computing covariate balance.
`prefilterCovariatesThreads`	The number of parallel threads to use for prefiltering covariates.
`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`.

`calibrationThreads`	The number of parallel threads to use for empirical calibration.

Value

An object of type `CmMultiThreadingSettings`.

See Also

`createDefaultMultiThreadingSettings()`

`createOutcome` *Create outcome definition*

Description

Create outcome definition

Usage

```
createOutcome(
  outcomeId,
  outcomeOfInterest = TRUE,
  trueEffectSize = NA,
  priorOutcomeLookback = NULL,
  riskWindowStart = NULL,
  startAnchor = NULL,
  riskWindowEnd = NULL,
  endAnchor = NULL
)
```

Arguments

`outcomeId`	An integer used to identify the outcome in the outcome cohort table.
`outcomeOfInterest`	Is this an outcome of interest? If not, creation of non-essential files will be skipped, including outcome=specific covariate balance files. This could be helpful to speed up analyses with many controls, for which we're only interested in the effect size estimate.
`trueEffectSize`	For negative and positive controls: the known true effect size. To be used for empirical calibration. Negative controls have `trueEffectSize = 1`. If the true effect size is unknown, use `trueEffectSize = NA`
`priorOutcomeLookback`	How many days should we look back when identifying prior. outcomes?
`riskWindowStart`	The start of the risk window (in days) relative to the `startAnchor`.
`startAnchor`	The anchor point for the start of the risk window. Can be `"cohort start"` or `"cohort end"`.
`riskWindowEnd`	The end of the risk window (in days) relative to the `endAnchor`.
`endAnchor`	The anchor point for the end of the risk window. Can be `"cohort start"` or `"cohort end"`.

Details

Any settings here that are not NULL will override any values set in `createCreateStudyPopulationArgs()`.

Value

An object of type `outcome`, to be used in `createTargetComparatorOutcomes()`.

`createPs` *Create propensity scores*

Description

Creates propensity scores and inverse probability of treatment weights (IPTW) using a regularized logistic regression.

Usage

```
createPs(
  cohortMethodData,
  population = NULL,
  excludeCovariateIds = c(),
  includeCovariateIds = c(),
  maxCohortSizeForFitting = 250000,
  errorOnHighCorrelation = TRUE,
  stopOnError = TRUE,
  prior = createPrior("laplace", exclude = c(0), useCrossValidation = TRUE),
  control = createControl(noiseLevel = "silent", cvType = "auto", seed = 1,
   resetCoefficients = TRUE, tolerance = 2e-07, cvRepetitions = 10, startingVariance =
    0.01),
  estimator = "att"
)
```

Arguments

`cohortMethodData`	An object of type CohortMethodData as generated using `getDbCohortMethodData()`.
`population`	A data frame describing the population. This should at least have a `rowId` column corresponding to the `rowId` column in the CohortMethodData covariates object and a `treatment` column. If population is not specified, the full population in the CohortMethodData will be used.
`excludeCovariateIds`	Exclude these covariates from the propensity model.
`includeCovariateIds`	Include only these covariates in the propensity model.
`maxCohortSizeForFitting`	If the target or comparator cohort are larger than this number, they will be downsampled before fitting the propensity model. The model will be used to compute propensity scores for all subjects. The purpose of the sampling is to gain speed. Setting this number to 0 means no downsampling will be applied.

`errorOnHighCorrelation`	If true, the function will test each covariate for correlation with the treatment assignment. If any covariate has an unusually high correlation (either positive or negative), this will throw and error.
`stopOnError`	If an error occur, should the function stop? Else, the two cohorts will be assumed to be perfectly separable.
`prior`	The prior used to fit the model. See `Cyclops::createPrior()` for details.
`control`	The control object used to control the cross-validation used to determine the hyperparameters of the prior (if applicable). See `Cyclops::createControl()` for details.
`estimator`	The type of estimator for the IPTW. Options are `estimator = "ate"` for the average treatment effect, `estimator = "att"` for the average treatment effect in the treated, and `estimator = "ato"` for the average treatment effect in the overlap population.

Details

IPTW estimates either the average treatment effect (ate) or average treatment effect in the treated (att) using stabilized inverse propensity scores (Xu et al. 2010).

References

Xu S, Ross C, Raebel MA, Shetterly S, Blanchette C, Smith D. Use of stabilized inverse propensity scores as weights to directly estimate relative risk and its confidence intervals. Value Health. 2010;13(2):273-277. doi:10.1111/j.1524-4733.2009.00671.x

Examples

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

`createResultsDataModel` *Create the results data model tables on a database server.*

Description

Create the results data model tables on a database server.

Usage

```
createResultsDataModel(
  connectionDetails = NULL,
  databaseSchema,
  tablePrefix = ""
)
```

Arguments

`connectionDetails`	DatabaseConnector connectionDetails instance @seealsoDatabaseConnector::createConnectionDetai
`databaseSchema`	The schema on the server where the tables will be created.
`tablePrefix`	(Optional) string to insert before table names for database table names

Details

Only PostgreSQL and SQLite servers are supported.

`createStratifyByPsAndCovariatesArgs`
Create a parameter object for the function stratifyByPsAndCovariates

---

Description

Create a parameter object for the function stratifyByPsAndCovariates

Usage

```
createStratifyByPsAndCovariatesArgs(
  numberOfStrata = 5,
  baseSelection = "all",
  covariateIds
)
```

Arguments

`numberOfStrata`	Into how many strata should the propensity score be divided? The boundaries of the strata are automatically defined to contain equal numbers of target persons.
`baseSelection`	What is the base selection of subjects where the strata bounds are to be determined? Strata are defined as equally-sized strata inside this selection. Possible values are "all", "target", and "comparator".
`covariateIds`	One or more covariate IDs in the cohortMethodData object on which subjects 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(),
  baseSelection = "all"
)
```

Arguments

`numberOfStrata`	How many strata? The boundaries of the strata are automatically defined to contain equal numbers of target 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.
`baseSelection`	What is the base selection of subjects where the strata bounds are to be determined? Strata are defined as equally-sized strata inside this selection. Possible values are "all", "target", and "comparator".

Details

Create an object defining the parameter values.

`createStudyPopulation` *Create a study population*

Description

Create a study population

Usage

```
createStudyPopulation(
  cohortMethodData,
  population = NULL,
  outcomeId = NULL,
  firstExposureOnly = FALSE,
  restrictToCommonPeriod = FALSE,
  washoutPeriod = 0,
  removeDuplicateSubjects = "keep all",
  removeSubjectsWithPriorOutcome = TRUE,
  priorOutcomeLookback = 99999,
  minDaysAtRisk = 1,
  maxDaysAtRisk = 99999,
  riskWindowStart = 0,
  startAnchor = "cohort start",
  riskWindowEnd = 0,
  endAnchor = "cohort end",
  censorAtNewRiskWindow = FALSE
)
```

Arguments

`cohortMethodData`	An object of type CohortMethodData as generated using `getDbCohortMethodData()`.
`population`	If specified, this population will be used as the starting point instead of the cohorts in the `cohortMethodData` object.
`outcomeId`	The ID of the outcome. If NULL, no outcome-specific transformations will be performed.
`firstExposureOnly`	Should only the first exposure per subject be included?
`restrictToCommonPeriod`	Restrict the analysis to the period when both exposures are observed?
`washoutPeriod`	The minimum required continuous observation time prior to index date for a person to be included in the cohort.
`removeDuplicateSubjects`	Remove subjects that are in both the target and comparator cohort? See details for allowed values.
`removeSubjectsWithPriorOutcome`	Remove subjects that have the outcome prior to the risk window start?
`priorOutcomeLookback`	How many days should we look back when identifying prior outcomes?
`minDaysAtRisk`	The minimum required number of days at risk. Risk windows with fewer days than this number are removed from the analysis.
`maxDaysAtRisk`	The maximum allowed number of days at risk. Risk windows that are longer will be truncated to this number of days.
`riskWindowStart`	The start of the risk window (in days) relative to the `startAnchor`.
`startAnchor`	The anchor point for the start of the risk window. Can be `"cohort start"` or `"cohort end"`.
`riskWindowEnd`	The end of the risk window (in days) relative to the `endAnchor`.
`endAnchor`	The anchor point for the end of the risk window. Can be `"cohort start"` or `"cohort end"`.
`censorAtNewRiskWindow`	If a subject is in multiple cohorts, should time-at-risk be censored when the new time-at-risk starts to prevent overlap?

Details

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

The `removeduplicateSubjects` argument can have one of the following values:

- `"keep all"`: Do not remove subjects that appear in both target and comparator cohort
- `"keep first"`: When a subjects appear in both target and comparator cohort, only keep whichever cohort is first in time. If both cohorts start simultaneous, the person is removed from the analysis.
- `"remove all"`: Remove subjects that appear in both target and comparator cohort completely from the analysis."

Value

A `tibble` specifying the study population. This `tibble` will have the following columns:

- `rowId`: A unique identifier for an exposure.
- `personSeqId`: The person sequence ID of the subject.
- `cohortStartdate`: The index date.
- `outcomeCount` The number of outcomes observed during the risk window.
- `timeAtRisk`: The number of days in the risk window.
- `survivalTime`: The number of days until either the outcome or the end of the risk window.

`createTargetComparatorOutcomes`
Create target-comparator-outcomes combinations.

Description

Create target-comparator-outcomes combinations.

Usage

```
createTargetComparatorOutcomes(
  targetId,
  comparatorId,
  outcomes,
  excludedCovariateConceptIds = c(),
  includedCovariateConceptIds = c()
)
```

Arguments

`targetId`	A cohort ID identifying the target exposure in the exposure table.
`comparatorId`	A cohort ID identifying the comparator exposure in the exposure table.
`outcomes`	A list of object of type `outcome` as created by `createOutcome()`.

`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 target-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 target-comparator combination.

Details

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

Value

An object of type `targetComparatorOutcomes`.

`createTrimByIptwArgs` *Create a parameter object for the function trimByIptw*

Description

Create a parameter object for the function trimByIptw

Usage

```
createTrimByIptwArgs(maxWeight = 10)
```

Arguments

`maxWeight`	The maximum allowed IPTW.

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 target group, persons with the highest propensity scores will be removed, in the comparator 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(0.3, 0.7))
```

Arguments

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

Details

Create an object defining the parameter values.

`createTruncateIptwArgs` *Create a parameter object for the function truncateIptw*

Description

Create a parameter object for the function truncateIptw

Usage

```
createTruncateIptwArgs(maxWeight = 10)
```

Arguments

`maxWeight`	The maximum allowed IPTW.

Details

Create an object defining the parameter values.

`drawAttritionDiagram` *Draw the attrition diagram*

Description

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

Usage

```
drawAttritionDiagram(
  object,
  targetLabel = "Target",
  comparatorLabel = "Comparator",
  fileName = NULL
)
```

Arguments

`object`	Either an object of type `cohortMethodData`, a population object generated by functions like `createStudyPopulation`, or an object of type `outcomeModel`.
`targetLabel`	A label to us for the target 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.

`exportToCsv` *Export cohort method results to CSV files*

Description

Export cohort method results to CSV files

Usage

```
exportToCsv(
  outputFolder,
  exportFolder = file.path(outputFolder, "export"),
  databaseId,
  minCellCount = 5,
  maxCores = 1,
  cmDiagnosticThresholds = createCmDiagnosticThresholds()
)
```

Arguments

`outputFolder`	The folder where runCmAnalyses() generated all results.
`exportFolder`	The folder where the CSV files will written.
`databaseId`	A unique ID for the database. This will be appended to most tables.
`minCellCount`	To preserve privacy: the minimum number of subjects contributing to a count before it can be included in the results. If the count is below this threshold, it will be set to `-minCellCount`.
`maxCores`	How many parallel cores should be used?
`cmDiagnosticThresholds`	An object of type `CmDiagnosticThresholds` as created using `createCmDiagnosticThresholds()`.

Details

This requires that `runCmAnalyses()` has been executed first. It exports all the results in the `outputFolder` to CSV files for sharing with other sites.

Value

Does not return anything. Is called for the side-effect of populating the `exportFolder` with CSV files.

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

Description

Create an outcome model, and computes the relative risk

Usage

```
fitOutcomeModel(
  population,
  cohortMethodData = NULL,
  modelType = "logistic",
  stratified = FALSE,
  useCovariates = FALSE,
  inversePtWeighting = FALSE,
  interactionCovariateIds = c(),
  excludeCovariateIds = c(),
  includeCovariateIds = c(),
  profileGrid = NULL,
  profileBounds = c(log(0.1), log(10)),
  prior = createPrior("laplace", useCrossValidation = TRUE),
  control = createControl(cvType = "auto", seed = 1, resetCoefficients = TRUE,
   startingVariance = 0.01, tolerance = 2e-07, cvRepetitions = 10, noiseLevel = "quiet")
)
```

Arguments

`population`	A population object generated by `createStudyPopulation()`, potentially filtered by other functions.
`cohortMethodData`	An object of type CohortMethodData as generated using `getDbCohortMethodData()`. Can be omitted if not using covariates and not using interaction terms.
`modelType`	The type of outcome model that will be used. Possible values are "logistic", "poisson", or "cox".
`stratified`	Should the regression be conditioned on the strata defined in the population object (e.g. by matching or stratifying on propensity scores)?
`useCovariates`	Whether to use the covariates in the `cohortMethodData` object in the outcome model.
`inversePtWeighting`	Use inverse probability of treatment weighting (IPTW)
`interactionCovariateIds`	An optional vector of covariate IDs to use to estimate interactions with the main treatment effect.
`excludeCovariateIds`	Exclude these covariates from the outcome model.
`includeCovariateIds`	Include only these covariates in the outcome model.
`profileGrid`	A one-dimensional grid of points on the log(relative risk) scale where the likelihood for coefficient of variables is sampled. See details.
`profileBounds`	The bounds (on the log relative risk scale) for the adaptive sampling of the likelihood function. See details.
`prior`	The prior used to fit the model. See `Cyclops::createPrior()` for details.
`control`	The control object used to control the cross-validation used to determine the hyperparameters of the prior (if applicable). See `Cyclops::createControl()` for details.

Details

For likelihood profiling, either specify the `profileGrid` for a completely user- defined grid, or `profileBounds` for an adaptive grid. Both should be defined on the log effect size scale. When both `profileGrid` and `profileGrid` are `NULL` likelihood profiling is disabled.

Value

An object of class `OutcomeModel`. Generic function `print`, `coef`, and `confint` are available.

`getAttritionTable` *Get the attrition table for a population*

Description

Get the attrition table for a population

Usage

```
getAttritionTable(object)
```

Arguments

`object`	Either an object of type CohortMethodData, a population object generated by functions like `createStudyPopulation()`, or an object of type `outcomeModel`.

Value

A `tibble` specifying the number of people and exposures in the population after specific steps of filtering.

`getDataMigrator` *Get database migrations instance*

Description

Returns ResultModelManager DataMigrationsManager instance.

Usage

```
getDataMigrator(connectionDetails, databaseSchema, tablePrefix = "")
```

Arguments

`connectionDetails`	DatabaseConnector connection details object
`databaseSchema`	String schema where database schema lives
`tablePrefix`	(Optional) Use if a table prefix is used before table names (e.g. "cd_")

Value

Instance of ResultModelManager::DataMigrationManager that has interface for converting existing data models

---

`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,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  targetId,
  comparatorId,
  outcomeIds,
  studyStartDate = "",
  studyEndDate = "",
  exposureDatabaseSchema = cdmDatabaseSchema,
  exposureTable = "drug_era",
  outcomeDatabaseSchema = cdmDatabaseSchema,
  outcomeTable = "condition_occurrence",
  cdmVersion = "5",
  firstExposureOnly = FALSE,
  removeDuplicateSubjects = "keep all",
  restrictToCommonPeriod = FALSE,
  washoutPeriod = 0,
  maxCohortSize = 0,
  covariateSettings
)
```

Arguments

`connectionDetails`	An R object of type `connectionDetails` created using the `DatabaseConnector::createConnectio` function.
`cdmDatabaseSchema`	The name of the database schema that contains the OMOP CDM instance. Requires read permissions to this database. On SQL Server, this should specify both the database and the schema, so for example 'cdm_instance.dbo'.
`tempEmulationSchema`	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
`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_DEFINITION_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_DEFINITION_ID in the cohort-like table.
`outcomeIds`	A list of cohort IDs used to define outcomes.
`studyStartDate`	A calendar date specifying the minimum date that a cohort index date can appear. Date format is 'yyyymmdd'.
`studyEndDate`	A calendar date specifying the maximum date that a cohort index date can appear. Date format is 'yyyymmdd'. Important: the study end data is also used to truncate risk windows, meaning no outcomes beyond the study end date will be considered.

`exposureDatabaseSchema`	The name of the database schema that is the location where the exposure data used to define the exposure cohorts is available.
`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.
`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 supports "5".
`firstExposureOnly`	Should only the first exposure per subject be included? Note that this is typically done in the `createStudyPopulation()` function, but can already be done here for efficiency reasons.
`removeDuplicateSubjects`	Remove subjects that are in both the target and comparator cohort? See details for allowed values.Note that this is typically done in the `createStudyPopulation` function, but can already be done here for efficiency reasons.
`restrictToCommonPeriod`	Restrict the analysis to the period when both treatments are observed?
`washoutPeriod`	The minimum required continuous observation time prior to index date for a person to be included in the cohort. Note that this is typically done in the `createStudyPopulation` function, but can already be done here for efficiency reasons.
`maxCohortSize`	If either the target or the comparator cohort is larger than this number it will be sampled to this size. `maxCohortSize = 0` indicates no maximum size.
`covariateSettings`	An object of type `covariateSettings` as created using the `FeatureExtraction::createCovariat` function.

Details

Based on the arguments, the treatment and comparator cohorts are retrieved, as well as outcomes occurring in exposed subjects. 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 schema or in a separate schema. Similarly, outcomes are identified using the CONDITION_ERA table or through user-defined cohorts in a cohort table either inside the CDM schema or in a separate schema. Covariates are automatically extracted from the appropriate tables within the CDM.

Important: The target and comparator drug must not be included in the covariates, including any descendant concepts. You will need to manually add the drugs and descendants to the `excludedCovariateConceptIds` of the `covariateSettings` argument.

The `removeduplicateSubjects` argument can have one of the following values:

- `"keep all"`: Do not remove subjects that appear in both target and comparator cohort
- `"keep first"`: When a subjects appear in both target and comparator cohort, only keep whichever cohort is first in time.

- `"remove all"`: Remove subjects that appear in both target and comparator cohort completely from the analysis."

If the `covariateSettings` include cohort-based covariates, and the `covariateCohortTable` is `NULL`, the `covariateCohortDatabaseSchema` and `covariateCohortTable` will be set to the `exposureDatabaseSchema` and `exposureTable`, respectively .

Value

A CohortMethodData object.

---

getDefaultCmTable1Specifications *Get the default table 1 specifications*

Description

Loads the default specifications for a table 1, to be used with the `createTable1` function.

Important: currently only works for binary covariates.

Usage

```
getDefaultCmTable1Specifications()
```

Value

A specifications objects.

---

getFileReference *Get file reference*

Description

Get file reference

Usage

```
getFileReference(outputFolder)
```

Arguments

`outputFolder`	Name of the folder where all the outputs have been written to.

Value

A tibble containing file names of artifacts generated for each target-comparator-outcome-analysis combination.

`getFollowUpDistribution` *Get the distribution of follow-up time*

Description

Get the distribution of follow-up time

Usage

```
getFollowUpDistribution(population, quantiles = c(0, 0.25, 0.5, 0.75, 1))
```

Arguments

`population`	A data frame describing the study population as created using the `createStudyPopulation` function. This should at least have these columns: treatment, timeAtRisk.
`quantiles`	The quantiles of the population to compute minimum follow-up time for.

Details

Get the distribution of follow-up time as quantiles. Follow-up time is defined as time-at-risk, so not censored at the outcome.

Value

A data frame with per treatment group at each quantile the amount of follow-up time available.

`getGeneralizabilityTable` *Get information on generalizability*

Description

to assess generalizability we compare the distribution of covariates before and after any (propensity score) adjustments. We compute the standardized difference of mean as our metric of generalizability. (Lipton et al., 2017)

Depending on our target estimand, we need to consider a different base population for generalizability. For example, if we aim to estimate the average treatment effect in thetreated (ATT), our base population should be the target population, meaning we should consider the covariate distribution before and after PS adjustment in the target population only. By default this function will attempt to select the right base population based on what operations have been performed on the population. For example, if PS matching has been performed we assume the target estimand is the ATT, and the target population is selected as base.

Requires running `computeCovariateBalance()`‘ first.

Usage

```
getGeneralizabilityTable(balance, baseSelection = "auto")
```

Arguments

`balance`	A data frame created by the `computeCovariateBalance` function.
`baseSelection`	The selection of the population to consider for generalizability. Options are "auto", "target", "comparator", and "both". The "auto" option will attempt to use the balance meta-data to pick the most appropriate population based on the target estimator.

Value

A tibble with the following columns:

- covariateId: The ID of the covariate. Can be linked to the `covariates` and `covariateRef` tables in the `CohortMethodData` object.
- covariateName: The name of the covariate.
- beforeMatchingMean: The mean covariate value before any (propensity score) adjustment.
- afterMatchingMean: The mean covariate value after any (propensity score) adjustment.
- stdDiff: The standardized difference of means between before and after adjustment.

The tibble also has a 'baseSelection' attribute, documenting the base population used to assess generalizability.

References

Tipton E, Hallberg K, Hedges LV, Chan W (2017) Implications of Small Samples for Generalization: Adjustments and Rules of Thumb, Eval Rev. Oct;41(5):472-505.

`getInteractionResultsSummary`

Get a summary report of the analyses results

Description

Get a summary report of the analyses results

Usage

```
getInteractionResultsSummary(outputFolder)
```

Arguments

`outputFolder`	Name of the folder where all the outputs have been written to.

Value

A tibble containing summary statistics for each target-comparator-outcome-analysis combination.

`getOutcomeModel` *Get the outcome model*

Description

Get 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 `fitOutcomeModel()` function.
`cohortMethodData`	An object of type CohortMethodData as generated using `getDbCohortMethodData()`.

Value

A tibble.

`getPsModel` *Get the propensity model*

Description

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

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()`.

Value

A tibble.

getResultsDataModelSpecifications *Get specifications for CohortMethod results data model*

Description

Get specifications for CohortMethod results data model

Usage

```
getResultsDataModelSpecifications()
```

Value

A tibble data frame object with specifications

getResultsSummary *Get a summary report of the analyses results*

Description

Get a summary report of the analyses results

Usage

```
getResultsSummary(outputFolder)
```

Arguments

`outputFolder`	Name of the folder where all the outputs have been written to.

Value

A tibble containing summary statistics for each target-comparator-outcome-analysis combination.

`insertExportedResultsInSqlite` *Insert exported results into a SQLite database*

Description

Insert exported results into a SQLite database

Usage

```
insertExportedResultsInSqlite(sqliteFileName, exportFolder, cohorts)
```

Arguments

`sqliteFileName`	The name of the SQLite file to store the results in. If the file does not exist it will be created.
`exportFolder`	The folder containing the CSV files to upload, as generated using the `exportToCsv()` function.
`cohorts`	A data frame describing the cohorts used in the study. Should include the target, comparator, and outcome of interest cohorts. The data frame should at least have a `cohortId` and `cohortName` columns.

Value

Does not return anything. Called for the side effect of inserting data into the SQLite database.

`isCohortMethodData` *Check whether an object is a CohortMethodData object*

Description

Check whether an object is a CohortMethodData object

Usage

```
isCohortMethodData(x)
```

Arguments

`x`	The object to check.

Value

A logical value.

`launchResultsViewer` *Launch Shiny app using*

Description

Launch Shiny app using

Usage

```
launchResultsViewer(connectionDetails, databaseSchema)
```

Arguments

`connectionDetails`	An R object of type `connectionDetails` created using the `DatabaseConnector::createConnectic` function.
`databaseSchema`	The name of the database schema where the results were written using `uploadExportedResults()`.

Value

Does not return anything. Is called for the side-effect of launching the Shiny app.

`launchResultsViewerUsingSqlite` *Launch Shiny app using a SQLite database*

Description

Launch Shiny app using a SQLite database

Usage

```
launchResultsViewerUsingSqlite(sqliteFileName)
```

Arguments

`sqliteFileName`	The name of the SQLite file where the results were stored using the `insertExportedResultsInSqli` function.

Value

Does not return anything. Is called for the side-effect of launching the Shiny app.

`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 method data from a file*

Description

Loads an object of type CohortMethodData from a file in the file system.

Usage

```
loadCohortMethodData(file)
```

Arguments

`file`	The name of the file containing the data.

Value

An object of class CohortMethodData.

`loadTargetComparatorOutcomesList` *Load a list of targetComparatorOutcomes from file*

Description

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

Usage

```
loadTargetComparatorOutcomesList(file)
```

Arguments

`file`	The name of the file

Value

A list of objects of type `targetComparatorOutcomes`.

`matchOnPs` *Match persons by propensity score*

Description

Use the provided propensity scores to match target to comparator persons.

Usage

```
matchOnPs(
  population,
  caliper = 0.2,
  caliperScale = "standardized logit",
  maxRatio = 1,
  allowReverseMatch = FALSE,
  stratificationColumns = c()
)
```

Arguments

`population`	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. Three scales are supported: `caliperScale = 'propensity score'`, `caliperScale = 'standardized'`, or `caliperScale = 'standardized logit'`. On the standardized scale, the caliper is interpreted in standard deviations of the propensity score distribution. 'standardized logit' is similar, except that the propensity score is transformed to the logit scale because the PS is more likely to be normally distributed on that scale (Austin, 2011).

`maxRatio`	The maximum number of persons in 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 target person.
`allowReverseMatch`	Allows n-to-1 matching if target arm is larger
`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 the following three columns:

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

The default caliper (0.2 on the standardized logit scale) is the one recommended by Austin (2011).

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.

Austin, PC. (2011) Optimal caliper widths for propensity-score matching when estimating differences in means and differences in proportions in observational studies, Pharmaceutical statistics, March, 10(2):150-161.

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

Use the provided propensity scores and a set of covariates to match target to comparator persons.

Usage

```
matchOnPsAndCovariates(
  population,
  caliper = 0.2,
  caliperScale = "standardized logit",
  maxRatio = 1,
  allowReverseMatch = FALSE,
  cohortMethodData,
  covariateIds
)
```

Arguments

`population`	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. Three scales are supported: `caliperScale = 'propensity score'`, `caliperScale = 'standardized'`, or `caliperScale = 'standardized logit'`. On the standardized scale, the caliper is interpreted in standard deviations of the propensity score distribution. 'standardized logit' is similar, except that the propensity score is transformed to the logit scale because the PS is more likely to be normally distributed on that scale (Austin, 2011).
`maxRatio`	The maximum number of persons in 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 target person.
`allowReverseMatch`	Allows n-to-1 matching if target arm is larger
`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 the following three columns:

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

The default caliper (0.2 on the standardized logit scale) is the one recommended by Austin (2011).

Value

Returns a tibble 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.

Austin, PC. (2011) Optimal caliper widths for propensity-score matching when estimating differences in means and differences in proportions in observational studies, Pharmaceutical statistics, March, 10(2):150-161.

`migrateDataModel` *Migrate Data model*

Description

Migrate data from current state to next state

It is strongly advised that you have a backup of all data (either sqlite files, a backup database (in the case you are using a postgres backend) or have kept the csv/zip files from your data generation.

Usage

```
migrateDataModel(connectionDetails, databaseSchema, tablePrefix = "")
```

Arguments

`connectionDetails`	DatabaseConnector connection details object
`databaseSchema`	String schema where database schema lives
`tablePrefix`	(Optional) Use if a table prefix is used before table names (e.g. "cd_")

`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,
  title = NULL,
  fileName = NULL,
  beforeLabel = "before matching",
  afterLabel = "after matching"
)
```

Arguments

`balance`	A data frame created by the `computeCovariateBalance` function.
`n`	(Maximum) count of covariates to plot.
`maxNameWidth`	Covariate names longer than this number of characters are truncated to create a nicer plot.
`title`	Optional: the main title for 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.
`beforeLabel`	Label for identifying data before matching / stratification / trimming.
`afterLabel`	Label for identifying data after matching / stratification / trimming.

Value

A ggplot object. Use the ggplot2::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,
  absolute = TRUE,
  threshold = 0,
  title = "Standardized difference of mean",
  fileName = NULL,
  beforeLabel = "Before matching",
  afterLabel = "After matching",
  showCovariateCountLabel = FALSE,
  showMaxLabel = FALSE
)
```

Arguments

`balance`	A data frame created by the `computeCovariateBalance` function.
`absolute`	Should the absolute value of the difference be used?
`threshold`	Show a threshold value for after matching standardized difference.
`title`	The main title for 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.
`beforeLabel`	Label for the x-axis.
`afterLabel`	Label for the y-axis.
`showCovariateCountLabel`	Show a label with the number of covariates included in the plot?
`showMaxLabel`	Show a label with the maximum absolute standardized difference after matching/stratification?

Value

A ggplot object. Use the ggplot2::ggsave function to save to file in a different format.

`plotCovariatePrevalence` *Plot covariate prevalence*

Description

Plot prevalence of binary covariates in the target and comparator cohorts, before and after matching. Requires running `computeCovariateBalance` first.

Usage

```
plotCovariatePrevalence(
  balance,
  threshold = 0,
  title = "Covariate prevalence",
  fileName = NULL,
  beforeLabel = "Before matching",
  afterLabel = "After matching",
  targetLabel = "Target",
  comparatorLabel = "Comparator"
)
```

Arguments

`balance`	A data frame created by the `computeCovariateBalance` function.
`threshold`	A threshold value for standardized difference. When exceeding the threshold, covariates will be marked in a different color. If `threshold = 0`, no color coding will be used.
`title`	The main title for 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.
`beforeLabel`	Label for the before matching / stratification panel.
`afterLabel`	Label for the after matching / stratification panel.
`targetLabel`	Label for the x-axis.
`comparatorLabel`	Label for the y-axis.

Value

A ggplot object. Use the ggplot2::ggsave function to save to file in a different format.

`plotFollowUpDistribution` *Plot the distribution of follow-up time*

Description

Plot the distribution of follow-up time

Usage

```
plotFollowUpDistribution(
  population,
  targetLabel = "Target",
  comparatorLabel = "Comparator",
  yScale = "percent",
  logYScale = FALSE,
  dataCutoff = 0.95,
  title = NULL,
  fileName = NULL
)
```

Arguments

`population`	A data frame describing the study population as created using the `createStudyPopulation` function. This should at least have these columns: treatment, timeAtRisk.
`targetLabel`	A label to us for the target cohort.
`comparatorLabel`	A label to us for the comparator cohort.
`yScale`	Should be either 'percent' or 'count'.
`logYScale`	Should the Y axis be on the log scale?
`dataCutoff`	Fraction of the data (number censored) after which the graph will not be shown.
`title`	The main title of 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

Plot the distribution of follow-up time, stratified by treatment group.Follow-up time is defined as time-at-risk, so not censored at the outcome.

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 Kaplan-Meier (KM) survival plot. Based (partially) on recommendations in Pocock et al (2002).

When variable-sized strata are detected, an adjusted KM plot is computed to account for stratified data, as described in Galimberti eta al (2002), using the closed form variance estimator described in Xie et al (2005).

Usage

```
plotKaplanMeier(
  population,
  censorMarks = FALSE,
  confidenceIntervals = TRUE,
  includeZero = FALSE,
  dataTable = TRUE,
  dataCutoff = 0.9,
  targetLabel = "Treated",
  comparatorLabel = "Comparator",
  title = NULL,
  fileName = NULL
)
```

Arguments

`population`	A population object generated by `createStudyPopulation`, potentially filtered by other functions.
`censorMarks`	Whether or not to include censor marks in the plot.
`confidenceIntervals`	Plot 95 percent confidence intervals? Default is TRUE, as recommended by Pocock et al.
`includeZero`	Should the y axis include zero, or only go down to the lowest observed survival? The default is FALSE, as recommended by Pocock et al.
`dataTable`	Should the numbers at risk be shown in a table? Default is TRUE, as recommended by Pocock et al.
`dataCutoff`	Fraction of the data (number censored) after which the graph will not be shown. The default is 90 percent as recommended by Pocock et al.
`targetLabel`	A label to us for the target cohort.

`comparatorLabel`	A label to us for the comparator cohort.
`title`	The main title of 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.

Value

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

References

Pocock SJ, Clayton TC, Altman DG. (2002) Survival plots of time-to-event outcomes in clinical trials: good practice and pitfalls, Lancet, 359:1686-89.

Galimberti S, Sasieni P, Valsecchi MG (2002) A weighted Kaplan-Meier estimator for matched data with application to the comparison of chemotherapy and bone-marrow transplant in leukaemia. Statistics in Medicine, 21(24):3847-64.

Xie J, Liu C. (2005) Adjusted Kaplan-Meier estimator and log-rank test with inverse probability of treatment weighting for survival data. Statistics in Medicine, 26(10):2276.

`plotPs` *Plot the propensity score distribution*

Description

Plots the propensity (or preference) score distribution.

Usage

```
plotPs(
  data,
  unfilteredData = NULL,
  scale = "preference",
  type = "density",
  binWidth = 0.05,
  targetLabel = "Target",
  comparatorLabel = "Comparator",
  showCountsLabel = FALSE,
  showAucLabel = FALSE,
  showEquiposeLabel = FALSE,
  equipoiseBounds = c(0.3, 0.7),
  unitOfAnalysis = "subjects",
  title = NULL,
  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. Four possible values: `type = 'density'` `type = 'histogram'`, `type = 'histogramCount'`, or `type = 'histogramProportion'`. `'histogram'` defaults to `'histogramCount'`.
`binWidth`	For histograms, the width of the bins
`targetLabel`	A label to us for the target cohort.
`comparatorLabel`	A label to us for the comparator cohort.
`showCountsLabel`	Show subject counts?
`showAucLabel`	Show the AUC?
`showEquiposeLabel`	Show the percentage of the population in equipoise?
`equipoiseBounds`	The bounds on the preference score to determine whether a subject is in equipoise.
`unitOfAnalysis`	The unit of analysis in the input data. Defaults to 'subjects'.
`title`	Optional: the main title for the plot.
`fileName`	Name of the file where the plot should be saved, for example 'plot.png'. See the function `ggplot2::ggsave()` 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 target (1) or comparator (0) group
- propensityScore (numeric): Propensity score

Value

A ggplot object. Use the `ggplot2::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)
```

---

plotTimeToEvent *Plot time-to-event*

Description

Plot time-to-event

Usage

```
plotTimeToEvent(
  cohortMethodData,
  population = NULL,
  outcomeId = NULL,
  firstExposureOnly = FALSE,
  restrictToCommonPeriod = FALSE,
  washoutPeriod = 0,
  removeDuplicateSubjects = "keep all",
  minDaysAtRisk = 1,
  riskWindowStart = 0,
  startAnchor = "cohort start",
  riskWindowEnd = 0,
  endAnchor = "cohort end",
  censorAtNewRiskWindow = FALSE,
  periodLength = 7,
  numberOfPeriods = 52,
  highlightExposedEvents = TRUE,
  includePostIndexTime = TRUE,
  showFittedLines = TRUE,
  targetLabel = "Target",
  comparatorLabel = "Comparator",
  title = NULL,
  fileName = NULL
)
```

Arguments

`cohortMethodData`	An object of type CohortMethodData as generated using `getDbCohortMethodData()`.
`population`	If specified, this population will be used as the starting point instead of the cohorts in the `cohortMethodData` object.
`outcomeId`	The ID of the outcome. If NULL, no outcome-specific transformations will be performed.

`firstExposureOnly`	(logical) Should only the first exposure per subject be included?
`restrictToCommonPeriod`	(logical) Restrict the analysis to the period when both exposures are observed?
`washoutPeriod`	The minimum required continuous observation time prior to index date for a person to be included in the cohort.
`removeDuplicateSubjects`	Remove subjects that are in both the target and comparator cohort? See details for allowed values.
`minDaysAtRisk`	The minimum required number of days at risk.
`riskWindowStart`	The start of the risk window (in days) relative to the `startAnchor`.
`startAnchor`	The anchor point for the start of the risk window. Can be `"cohort start"` or `"cohort end"`.
`riskWindowEnd`	The end of the risk window (in days) relative to the `endAnchor`.
`endAnchor`	The anchor point for the end of the risk window. Can be `"cohort start"` or `"cohort end"`.
`censorAtNewRiskWindow`	If a subject is in multiple cohorts, should time-at-risk be censored when the new time-at-risk starts to prevent overlap?
`periodLength`	The length in days of each period shown in the plot.
`numberOfPeriods`	Number of periods to show in the plot. The periods are equally divided before and after the index date.
`highlightExposedEvents`	(logical) Highlight event counts during exposure in a different color?
`includePostIndexTime`	(logical) Show time after the index date?
`showFittedLines`	(logical) Fit lines to the proportions and show them in the plot?
`targetLabel`	A label to us for the target cohort.
`comparatorLabel`	A label to us for the comparator cohort.
`title`	Optional: the main title for the plot.
`fileName`	Name of the file where the plot should be saved, for example 'plot.png'. See `ggplot2::ggsave()` for supported file formats.

Details

Creates a plot showing the number of events over time in the target and comparator cohorts, both before and after index date. The plot also distinguishes between events inside and outside the time-at-risk period. This requires the user to (re)specify the time-at-risk using the same arguments as the `createStudyPopulation()` function. Note that it is not possible to specify that people with the outcome prior should be removed, since the plot will show these prior events.

Value

A ggplot object. Use the `ggplot2::ggsave()` function to save to file in a different format.

`runCmAnalyses` *Run a list of analyses*

Description

Run a list of analyses

Usage

```
runCmAnalyses(
  connectionDetails,
  cdmDatabaseSchema,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  exposureDatabaseSchema = cdmDatabaseSchema,
  exposureTable = "drug_era",
  outcomeDatabaseSchema = cdmDatabaseSchema,
  outcomeTable = "condition_occurrence",
  cdmVersion = "5",
  outputFolder = "./CohortMethodOutput",
  cmAnalysisList,
  targetComparatorOutcomesList,
  analysesToExclude = NULL,
  refitPsForEveryOutcome = FALSE,
  refitPsForEveryStudyPopulation = TRUE,
  multiThreadingSettings = createMultiThreadingSettings()
)
```

Arguments

`connectionDetails`
: An R object of type `connectionDetails` created using the `DatabaseConnector::createConnecti`
function.

`cdmDatabaseSchema`
: The name of the database schema that contains the OMOP CDM instance. Requires read permissions to this database. On SQL Server, this should specify both the database and the schema, so for example 'cdm_instance.dbo'.

`tempEmulationSchema`
: Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.

`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.
`cdmVersion`	Define the OMOP CDM version used: currently support "4" and "5".
`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.
`targetComparatorOutcomesList`	A list of objects of type `targetComparatorOutcomes` as created using the createTargetComparatorOutcomes function.
`analysesToExclude`	Analyses to exclude. See the Analyses to Exclude section for details.
`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.
`refitPsForEveryStudyPopulation`	Should the propensity model be fitted for every study population definition? If false, a single propensity model will be fitted, and the study population criteria will be applied afterwards.
`multiThreadingSettings`	An object of type `CmMultiThreadingSettings` as created using the `createMultiThreadingSettin` or `createDefaultMultiThreadingSettings()` functions.

Details

Run a list of analyses for the target-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(targetComparatorOutcomesList)` (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.

After completion, a tibble containing references to all generated files can be obtained using the `getFileReference()` function. A summary of the analysis results can be obtained using the `getResultsSummary()` function.

Analyses to Exclude:

Normally, `runCmAnalyses` will run all combinations of target-comparator-outcome-analyses settings. However, sometimes we may not need all those combinations. Using the `analysesToExclude` argument, we can remove certain items from the full matrix. This argument should be a data frame with at least one of the following columns:

- targetId

- comparatorId
- outcomeId
- analysisId

This data frame will be joined to the outcome model reference table before executing, and matching rows will be removed. For example, if one specifies only one target ID and analysis ID, then any analyses with that target and that analysis ID will be skipped.

Value

A tibble describing for each target-comparator-outcome-analysisId combination where the intermediary and outcome model files can be found, relative to the `outputFolder`.

`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 method data to file*

Description

Saves an object of type CohortMethodData to a file.

Usage

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

Arguments

`cohortMethodData`	An object of type CohortMethodData as generated using `getDbCohortMethodData()`.
`file`	The name of the file where the data will be written. If the file already exists it will be overwritten.

Value

Returns no output.

`saveTargetComparatorOutcomesList` *Save a list of targetComparatorOutcomes to file*

Description

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

Usage

```
saveTargetComparatorOutcomesList(targetComparatorOutcomesList, file)
```

Arguments

`targetComparatorOutcomesList`	The targetComparatorOutcomes list to be written to file
`file`	The name of the file where the results will be written

`simulateCohortMethodData` *Generate simulated data*

Description

Creates a CohortMethodData object with simulated data.

Usage

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

Arguments

`profile`	An object of type `CohortMethodDataSimulationProfile` 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

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

Usage

```
stratifyByPs(
  population,
  numberOfStrata = 5,
  stratificationColumns = c(),
  baseSelection = "all"
)
```

Arguments

`population`	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 target 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.
`baseSelection`	What is the base selection of subjects where the strata bounds are to be determined? Strata are defined as equally-sized strata inside this selection. Possible values are "all", "target", and "comparator".

Details

The data frame should have the following three columns:

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

Value

Returns a tibble 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

Use the provided propensity scores and covariates to stratify persons.

Usage

```
stratifyByPsAndCovariates(
  population,
  numberOfStrata = 5,
  baseSelection = "all",
  cohortMethodData,
  covariateIds
)
```

Arguments

`population`	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 target persons.
`baseSelection`	What is the base selection of subjects where the strata bounds are to be determined? Strata are defined as equally-sized strata inside this selection. Possible values are "all", "target", and "comparator".
`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 (numeric): A unique identifier for each row (e.g. the person ID).
- treatment (integer): Column indicating whether the person is in the target (1) or comparator (0) group.
- propensityScore (numeric): Propensity score.

Value

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

`trimByIptw` *Remove subjects with a high IPTW*

Description

Remove subjects having a weight higher than the user-specified threshold.

Usage

```
trimByIptw(population, maxWeight = 10)
```

Arguments

`population`	A data frame with at least the two columns described in the details
`maxWeight`	The maximum allowed IPTW.

Details

The data frame should have the following two columns:

- treatment (integer): Column indicating whether the person is in the target (1) or comparator (0) group.
- iptw (numeric): Propensity score.

Value

Returns a tibble with the same columns as the input.

Examples

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

`trimByPs` *Trim persons by propensity score*

Description

Use the provided propensity scores to trim subjects with extreme scores.

Usage

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

Arguments

`population`	A data frame with the three columns described below
`trimFraction`	This fraction will be removed from each treatment group. In the target 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 (numeric): A unique identifier for each row (e.g. the person ID).
- treatment (integer): Column indicating whether the person is in the target (1) or comparator (0) group.
- propensityScore (numeric): Propensity score.

Value

Returns a tibble 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

Use the preference score to trim subjects that are not in clinical equipoise

Usage

```
trimByPsToEquipoise(population, bounds = c(0.3, 0.7))
```

Arguments

`population`	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 (numeric): A unique identifier for each row (e.g. the person ID).
- treatment (integer): Column indicating whether the person is in the target (1) or comparator (0) group.
- propensityScore (numeric): Propensity score.

Value

Returns a tibble 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)
```

`truncateIptw` *Truncate IPTW values*

Description

Set the inverse probability of treatment weights (IPTW) to the user-specified threshold if it exceeds said threshold.

Usage

```
truncateIptw(population, maxWeight = 10)
```

Arguments

`population`	A data frame with at least the two columns described in the details
`maxWeight`	The maximum allowed IPTW.

Details

The data frame should have the following two columns:

- treatment (integer): Column indicating whether the person is in the target (1) or comparator (0) group.
- iptw (numeric): Propensity score.

Value

Returns a tibble with the same columns as the input.

Examples

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

uploadExportedResults *Upload exported results to a database*

Description

Upload exported results to a database

Usage

```
uploadExportedResults(
  connectionDetails,
  databaseSchema,
  append = FALSE,
  exportFolder,
  cohorts
)
```

Arguments

`connectionDetails`	An R object of type `connectionDetails` created using the `DatabaseConnector::createConnectio` function.
`databaseSchema`	The name of the database schema where the results will be written.
`append`	Append the results to existing tables? Can be used for uploading results from multiple databases into a single results schema.
`exportFolder`	The folder containing the CSV files to upload, as generated using the `exportToCsv()` function.
`cohorts`	A data frame describing the cohorts used in the study. Should include the target, comparator, and outcome of interest cohorts. The data frame should at least have a `cohortId` and `cohortName` columns.

Value

Does not return anything. Is called for the side-effect of having the results uploaded to the server.

uploadResults *Upload results to the database server.*

Description

Requires the results data model tables have been created using the `createResultsDataModel` function.

Usage

```
uploadResults(
  connectionDetails,
  schema,
  zipFileName,
  forceOverWriteOfSpecifications = FALSE,
  purgeSiteDataBeforeUploading = TRUE,
  tempFolder = tempdir(),
  tablePrefix = "",
  ...
)
```

Arguments

`connectionDetails`	An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package.
`schema`	The schema on the server where the tables have been created.
`zipFileName`	The name of the zip file.
`forceOverWriteOfSpecifications`	If TRUE, specifications of the phenotypes, cohort definitions, and analysis will be overwritten if they already exist on the database. Only use this if these specifications have changed since the last upload.
`purgeSiteDataBeforeUploading`	If TRUE, before inserting data for a specific databaseId all the data for that site will be dropped. This assumes the input zip file contains the full data for that data site.
`tempFolder`	A folder on the local file system where the zip files are extracted to. Will be cleaned up when the function is finished. Can be used to specify a temp folder on a drive that has sufficient space if the default system temp space is too limited.
`tablePrefix`	(Optional) string to insert before table names for database table names
`...`	See ResultModelManager::uploadResults

Index