Package ‘OhdsiReportGenerator’

February 26, 2025

Type Package

Title Observational Health Data Sciences and Informatics Report Generator

Version 1.0.1

Date 2025-2-19

Maintainer Jenna Reps <`jreps@its.jnj.com`>

Description Extract results into R from the Observational Health Data Sciences and Informatics result database (see <`https://ohdsi.github.io/Strategus/results-schema/index.html`>) and generate reports/presentations via 'quarto' that summarize results in HTML format. Learn more about 'OhdsiReportGenerator' at <`https://ohdsi.github.io/OhdsiReportGenerator/`>.

License Apache License 2.0

URL `https://ohdsi.github.io/OhdsiReportGenerator/`,
`https://github.com/OHDSI/OhdsiReportGenerator`

BugReports `https://github.com/OHDSI/OhdsiReportGenerator/issues`

VignetteBuilder knitr

Depends R (>= 3.3.0)

Imports DatabaseConnector, forestplot, dplyr, ggplot2, ggpubr, htmltools, kableExtra, ParallelLogger, quarto, reactable, rlang, rmarkdown, tibble, tidyr

Suggests knitr, markdown, ResultModelManager, RSQLite, testthat

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

Author Jenna Reps [aut, cre],
Anthony Sena [aut]

Repository CRAN

Date/Publication 2025-02-26 13:10:06 UTC

Contents

`generatePresentationMultiple`

generatePresentationMultiple

Description

Generates a presentation from a Strategus result

Usage

```
generatePresentationMultiple(
  server,
  username,
  password,
  dbms,
  resultsSchema = NULL,
  targetId = 1,
  targetName = "target cohort",
  cmSubsetId = 2,
  sccsSubsetId = NULL,
  indicationName = NULL,
  outcomeIds = 3,
  outcomeNames = "outcome cohort",
  comparatorIds = c(2, 4),
  comparatorNames = c("comparator cohort 1", "comparator cohort 2"),
  covariateIds = NULL,
  details = list(studyPeriod = "All Time", restrictions = "Age - None"),
  title = "ASSURE 001 ...",
  lead = "add name",
  date = Sys.Date(),
  backgroundText = "",
  evaluationText = "",
  outputLocation,
  outputName = paste0("presentation_", gsub(":", "_", gsub(" ", "_",
    as.character(date()))), ".html"),
  intermediateDir = tempdir()
)
```

Arguments

`server`	The server containing the result database
`username`	The username for an account that can access the result database
`password`	The password for an account that can access the result database
`dbms`	The dbms used to access the result database
`resultsSchema`	The result database schema

`targetId`	The cohort definition id for the target cohort
`targetName`	A friendly name for the target cohort
`cmSubsetId`	Optional a subset ID for the cohort method/prediction results
`sccsSubsetId`	Optional a subset ID for the SCCS and characterization results
`indicationName`	A name for the indication if used or NULL
`outcomeIds`	The cohort definition id for the outcome
`outcomeNames`	Friendly names for the outcomes
`comparatorIds`	The cohort method comparator cohort id
`comparatorNames`	Friendly names for the comparators
`covariateIds`	A vector of covariateIds to include in the characterization
`details`	a list with the studyPeriod and restrictions
`title`	A title for the presentation
`lead`	The name of the presentor
`date`	The date of the presentation
`backgroundText`	a character with any background text
`evaluationText`	a list of bullet points for the evaluation
`outputLocation`	The file location and name to save the protocol
`outputName`	The name of the html protocol that is created
`intermediateDir`	The work directory for quarto

Details

Specify the connection details to the result database and the schema name to generate a presentation.

Value

An named R list with the elements ’standard’ and ’source’

`getBinaryCaseSeries` *A function to extract case series characterization results*

Description

A function to extract case series characterization results

Usage

```
getBinaryCaseSeries(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetId = NULL,
  outcomeId = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cTablePrefix`	The prefix used for the characterization results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetId`	An integer corresponding to the target cohort ID
`outcomeId`	Am integer corresponding to the outcome cohort ID

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

A data.frame with the characterization case series results

See Also

Other Characterization: `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCharacterizationDemographics()`, `getContinuousCaseSeries()`, `getContinuousRiskFactors` `getDechallengeRechallenge()`, `getIncidenceRates()`, `getTargetBinaryFeatures()`, `getTargetContinuousFeature` `getTargetCounts()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

cs <- getBinaryCaseSeries(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetId = 1,
```

```
  outcomeId = 3
)
```

---

getBinaryRiskFactors *A function to extract non-case and case binary characterization results*

Description

A function to extract non-case and case binary characterization results

Usage

```
getBinaryRiskFactors(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetId = NULL,
  outcomeId = NULL,
  analysisIds = c(3)
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cTablePrefix`	The prefix used for the characterization results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetId`	An integer corresponding to the target cohort ID
`outcomeId`	Am integer corresponding to the outcome cohort ID
`analysisIds`	The feature extraction analysis ID of interest (e.g., 201 is condition)

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

A data.frame with the characterization results for the cases and non-cases

See Also

Other Characterization: `getBinaryCaseSeries()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCharacterizationDemographics()`, `getContinuousCaseSeries()`, `getContinuousRiskFactor` `getDechallengeRechallenge()`, `getIncidenceRates()`, `getTargetBinaryFeatures()`, `getTargetContinuousFeatur` `getTargetCounts()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

rf <- getBinaryRiskFactors(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetId = 1,
  outcomeId = 3
)
```

`getCaseBinaryFeatures` *Extract aggregate statistics of binary feature analysis IDs of interest for cases*

Description

This function extracts the feature extraction results for cases corresponding to specified target and outcome cohorts.

Usage

```
getCaseBinaryFeatures(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL,
  analysisIds = c(3)
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)

`cTablePrefix`	The prefix used for the characterization results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default 'database_meta_data')
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs
`analysisIds`	The feature extraction analysis ID of interest (e.g., 201 is condition)

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- targetName the target cohort name
- targetId the target cohort unique identifier
- outcomeName the outcome name
- outcomeId the outcome unique identifier
- minPriorObservation the minimum required observation days prior to index for an entry
- outcomeWashoutDays patients with the outcome occurring within this number of days prior to index are excluded (NA means no exclusion)
- riskWindowStart the number of days ofset the start anchor that is the start of the time-at-risk
- startAnchor the start anchor is either the target cohort start or cohort end date
- riskWindowEnd the number of days ofset the end anchor that is the end of the time-at-risk
- endAnchor the end anchor is either the target cohort start or cohort end date
- covariateName the name of the feature
- sumValue the number of cases who have the feature value of 1
- averageValue the mean feature value

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCharacterizationDemographics()`, `getContinuousCaseSeries()`, `getContinuousRiskFactor` `getDechallengeRechallenge()`, `getIncidenceRates()`, `getTargetBinaryFeatures()`, `getTargetContinuousFeatur` `getTargetCounts()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

cbf <- getCaseBinaryFeatures(
connectionHandler = connectionHandler,
schema = 'main'
)
```

`getCaseContinuousFeatures` *Extract aggregate statistics of continuous feature analysis IDs of interest for targets*

Description

This function extracts the continuous feature extraction results for cases corresponding to specified target and outcome cohorts.

Usage

```
getCaseContinuousFeatures(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL,
  analysisIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cTablePrefix`	The prefix used for the characterization results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs
`analysisIds`	The feature extraction analysis ID of interest (e.g., 201 is condition)

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- targetName the target cohort name
- targetId the target cohort unique identifier
- outcomeName the outcome name
- outcomeId the outcome unique identifier
- minPriorObservation the minimum required observation days prior to index for an entry
- outcomeWashoutDays patients with the outcome occurring within this number of days prior to index are excluded (NA means no exclusion)
- covariateName the name of the feature
- covariateId the id of the feature
- countValue the number of cases who have the feature
- minValue the minimum value observed for the feature
- maxValue the maximum value observed for the feature
- averageValue the mean value observed for the feature
- standardDeviation the standard deviation of the value observed for the feature
- medianValue the median value observed for the feature
- p10Value the 10th percentile of the value observed for the feature
- p25Value the 25th percentile of the value observed for the feature
- p75Value the 75th percentile of the value observed for the feature
- p90Value the 90th percentile of the value observed for the feature

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseCounts()`, `getCharacterizationDemographics()`, `getContinuousCaseSeries()`, `getContinuousRiskFactors` `getDechallengeRechallenge()`, `getIncidenceRates()`, `getTargetBinaryFeatures()`, `getTargetContinuousFeature` `getTargetCounts()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

ccf <- getCaseContinuousFeatures(
connectionHandler = connectionHandler,
schema = 'main'
)
```

`getCaseCounts` *Extract the outcome cohort counts result*

Description

This function extracts outcome cohort counts across databases in the results for specified target and outcome cohorts.

Usage

```
getCaseCounts(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cTablePrefix`	The prefix used for the characterization results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- targetName the target cohort name
- targetId the target cohort unique identifier
- outcomeName the outcome name
- outcomeId the outcome unique identifier

- rowCount the number of entries in the cohort
- personCount the number of people in the cohort
- minPriorObservation the minimum required observation days prior to index for an entry
- outcomeWashoutDays patients with the outcome occurring within this number of days prior to index are excluded (NA means no exclusion)
- riskWindowStart the number of days ofset the start anchor that is the start of the time-at-risk
- startAnchor the start anchor is either the target cohort start or cohort end date
- riskWindowEnd the number of days ofset the end anchor that is the end of the time-at-risk
- endAnchor the end anchor is either the target cohort start or cohort end date

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCharacterizationDemographics()`, `getContinuousCaseSeries()`, `getContinuousRiskFactors()`, `getDechallengeRechallenge()`, `getIncidenceRates()`, `getTargetBinaryFeatures()` `getTargetContinuousFeatures()`, `getTargetCounts()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

cc <- getCaseCounts(
connectionHandler = connectionHandler,
schema = 'main'
)
```

`getCharacterizationDemographics`

Extract the binary age groups for the cases and targets

Description

This function extracts the age group feature extraction results for cases and targets corresponding to specified target and outcome cohorts.

Usage

```
getCharacterizationDemographics(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
```

```
  databaseTable = "database_meta_data",
  targetId = NULL,
  outcomeId = NULL,
  type = "age"
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cTablePrefix`	The prefix used for the characterization results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetId`	An integer corresponding to the target cohort ID
`outcomeId`	Am integer corresponding to the outcome cohort ID
`type`	A character of ’age’ or ’sex’

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- targetName the target cohort name
- targetId the target cohort unique identifier
- outcomeName the outcome name
- outcomeId the outcome unique identifier
- minPriorObservation the minimum required observation days prior to index for an entry
- outcomeWashoutDays patients with the outcome occurring within this number of days prior to index are excluded (NA means no exclusion)
- riskWindowStart the number of days ofset the start anchor that is the start of the time-at-risk
- startAnchor the start anchor is either the target cohort start or cohort end date
- riskWindowEnd the number of days ofset the end anchor that is the end of the time-at-risk
- endAnchor the end anchor is either the target cohort start or cohort end date
- covariateName the name of the feature
- sumValue the number of cases who have the feature value of 1
- averageValue the mean feature value

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getContinuousCaseSeries()`, `getContinuousRiskFactors()`, `getDechallengeRechallenge()`, `getIncidenceRates()`, `getTargetBinaryFeatures()`, `getTargetContinuousFeatur`
`getTargetCounts()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
# example code

conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

ageData <- getCharacterizationDemographics(
connectionHandler = connectionHandler,
schema = 'main'
)
```

`getCmDiagnosticsData` *Extract the cohort method diagostic results*

Description

This function extracts the cohort method diagnostics that examine whether the analyses were sufficiently powered and checks for different types of bias.

Usage

```
getCmDiagnosticsData(
  connectionHandler,
  schema,
  cmTablePrefix = "cm_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL,
  comparatorIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)

`cmTablePrefix`	The prefix used for the cohort method results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default 'database_meta_data')
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs
`comparatorIds`	A vector of integers corresponding to the comparator cohort IDs

Details

Specify the connectionHandler, the schema and the target/comparator/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- analysisId the analysis unique identifier
- description a description of the analysis
- targetName the target cohort name
- targetId the target cohort unique identifier
- comparatorName the comparator cohort name
- comparatorId the comparator cohort unique identifier
- outcomeName the outcome name
- outcomeId the outcome cohort unique identifier
- maxSdm max allowed standardized difference of means when comparing the target to the comparator after PS adjustment for the ballance diagnostic diagnostic to pass.
- sharedMaxSdm max allowed standardized difference of means when comparing the target to the comparator after PS adjustment for the ballance diagnostic diagnostic to pass.
- equipoise the bounds on the preference score to determine whether a subject is in equipoise.
- mdrr the maximum passable minimum detectable relative risk (mdrr) value. If the mdrr is greater than this the diagnostics will fail.
- attritionFraction (depreciated) the minmum attrition before the diagnostics fails.
- ease The expected absolute systematic error (ease) measures residual bias.
- balanceDiagnostic whether the balance diagnostic passed or failed.
- sharedBalanceDiagnostic whether the shared balance diagnostic passed or failed.
- equipoiseDiagnostic whether the equipose diagnostic passed or failed.
- mdrrDiagnostic whether the mdrr (power) diagnostic passed or failed.
- attritionDiagnostic (depreciated) whether the attrition diagnostic passed or failed.
- easeDiagnostic whether the ease diagnostic passed or failed.
- unblind whether the results can be unblinded.
- summaryValue summary of diagnostics results. FAIL, PASS or number of warnings.

See Also

Other Estimation: `getCMEstimation()`, `getCmMetaEstimation()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `plotCmEstimates()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

cmDiag <- getCmDiagnosticsData(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetIds = 1,
  outcomeIds = 3
)
```

`getCMEstimation` *Extract the cohort method results*

Description

This function extracts the single database cohort method estimates for results that can be unblinded and have a calibrated RR

Usage

```
getCMEstimation(
  connectionHandler,
  schema,
  cmTablePrefix = "cm_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL,
  comparatorIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cmTablePrefix`	The prefix used for the cohort method results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables

`databaseTable`	The name of the table with the database details (default 'database_meta_data')
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs
`comparatorIds`	A vector of integers corresponding to the comparator cohort IDs

Details

Specify the connectionHandler, the schema and the target/comparator/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- analysisId the analysis design unique identifier
- description the analysis design description
- targetName the target cohort name
- targetId the target cohort unique identifier
- comparatorName the comparator cohort name
- comparatorId the comparator cohort unique identifier
- outcomeName the outcome name
- outcomeId the outcome unique identifier
- calibratedRr the calibrated relative risk
- calibratedRrCi95Lb the calibrated relative risk 95 percent confidence interval lower bound
- calibratedRrCi95Ub the calibrated relative risk 95 percent confidence interval upper bound
- calibratedP the two sided calibrated p value
- calibratedOneSidedP the one sided calibrated p value
- calibratedLogRr the calibrated relative risk logged
- calibratedSeLogRr the standard error of the calibrated relative risk logged
- targetSubjects the number of people in the target cohort
- comparatorSubjects the number of people in the comparator cohort
- targetDays the total number of days at risk across the target cohort people
- comparatorDays the total number of days at risk across the comparator cohort people
- targetOutcomes the total number of outcomes occuring during the time at risk for the target cohort people
- comparatorOutcomes the total number of outcomes occuring during the time at risk for the comparator cohort people
- targetEstimator ...

See Also

Other Estimation: `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `plotCmEstimates()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

cmEst <- getCMEstimation(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetIds = 1,
  outcomeIds = 3
)
```

`getCmMetaEstimation` *Extract the cohort method meta analysis results*

Description

This function extracts any meta analysis estimation results for cohort method.

Usage

```
getCmMetaEstimation(
  connectionHandler,
  schema,
  cmTablePrefix = "cm_",
  cgTablePrefix = "cg_",
  esTablePrefix = "es_",
  targetIds = NULL,
  outcomeIds = NULL,
  comparatorIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cmTablePrefix`	The prefix used for the cohort method results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`esTablePrefix`	The prefix used for the evidence synthesis results tables
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs
`comparatorIds`	A vector of integers corresponding to the comparator cohort IDs

Details

Specify the connectionHandler, the schema and the target/comparator/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- analysisId the analysis unique identifier
- description a description of the analysis
- targetName the target cohort name
- targetId the target cohort unique identifier
- comparatorName the comparator cohort name
- comparatorId the comparator cohort unique identifier
- outcomeName the outcome name
- outcomeId the outcome cohort unique identifier
- calibratedRr the calibrated relative risk
- calibratedRrCi95Lb the calibrated relative risk 95 percent confidence interval lower bound
- calibratedRrCi95Ub the calibrated relative risk 95 percent confidence interval upper bound
- calibratedP the two sided calibrated p value
- calibratedOneSidedP the one sided calibrated p value
- calibratedLogRr the calibrated relative risk logged
- calibratedSeLogRr the standard error of the calibrated relative risk logged
- targetSubjects the number of people in the target cohort across included database
- comparatorSubjects the number of people in the comparator cohort across included database
- targetDays the total number of days at risk across the target cohort people across included database
- comparatorDays the total number of days at risk across the comparator cohort people across included database
- targetOutcomes the total number of outcomes occuring during the time at risk for the target cohort people across included database
- comparatorOutcomes the total number of outcomes occuring during the time at risk for the comparator cohort people across included database
- nDatabases the number of databases included

See Also

Other Estimation: `getCMEstimation()`, `getCmDiagnosticsData()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `plotCmEstimates()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

cmMeta <- getCmMetaEstimation(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetIds = 1,
  outcomeIds = 3
)
```

`getCohortDefinitions` *Extract the cohort definition details*

Description

This function extracts all cohort definitions for the targets of interest.

Usage

```
getCohortDefinitions(
  connectionHandler,
  schema,
  cgTablePrefix = "cg_",
  targetIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cgTablePrefix`	The prefix used for the cohort generator results tables
`targetIds`	A vector of integers corresponding to the target cohort IDs

Details

Specify the connectionHandler, the schema and the target cohort IDs

Value

Returns a data.frame with the cohort details

See Also

Other Cohorts: `getCohortSubsetDefinitions()`, `processCohorts()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

cohortDef <- getCohortDefinitions(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`getCohortSubsetDefinitions`

Extract the cohort subset definition details

Description

This function extracts all cohort subset definitions for the subsets of interest.

Usage

```
getCohortSubsetDefinitions(
  connectionHandler,
  schema,
  cgTablePrefix = "cg_",
  subsetIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cgTablePrefix`	The prefix used for the cohort generator results tables
`subsetIds`	A vector of subset cohort ids or NULL

Details

Specify the connectionHandler, the schema and the subset IDs

Value

Returns a data.frame with the cohort subset details

See Also

Other Cohorts: `getCohortDefinitions()`, `processCohorts()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

subsetDef <- getCohortSubsetDefinitions(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getContinuousCaseSeries

A function to extract case series continuous feature characterization results

Description

A function to extract case series continuous feature characterization results

Usage

```
getContinuousCaseSeries(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetId = NULL,
  outcomeId = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cTablePrefix`	The prefix used for the characterization results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetId`	An integer corresponding to the target cohort ID
`outcomeId`	Am integer corresponding to the outcome cohort ID

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

A data.frame with the characterization case series results

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCharacterizationDemographics()`, `getContinuousRiskFacto` `getDechallengeRechallenge()`, `getIncidenceRates()`, `getTargetBinaryFeatures()`, `getTargetContinuousFeature` `getTargetCounts()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

cs <- getContinuousCaseSeries(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetId = 1,
  outcomeId = 3
)
```

`getContinuousRiskFactors` *A function to extract non-case and case continuous characterization results*

Description

A function to extract non-case and case continuous characterization results

Usage

```
getContinuousRiskFactors(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetId = NULL,
  outcomeId = NULL,
  analysisIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cTablePrefix`	The prefix used for the characterization results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetId`	An integer corresponding to the target cohort ID
`outcomeId`	Am integer corresponding to the outcome cohort ID
`analysisIds`	The feature extraction analysis ID of interest (e.g., 201 is condition)

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

A data.frame with the characterization results for the cases and non-cases

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCharacterizationDemographics()`, `getContinuousCaseSerie` `getDechallengeRechallenge()`, `getIncidenceRates()`, `getTargetBinaryFeatures()`, `getTargetContinuousFeature` `getTargetCounts()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

rf <- getContinuousRiskFactors(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetId = 1,
  outcomeId = 3
)
```

`getDechallengeRechallenge` *Extract the dechallenge rechallenge results*

Description

This function extracts all dechallenge rechallenge results across databases for specified target and outcome cohorts.

Usage

```
getDechallengeRechallenge(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cTablePrefix`	The prefix used for the characterization results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- targetName the target cohort name
- targetId the target cohort unique identifier
- outcomeName the outcome name

- outcomeId the outcome unique identifier
- dechallengeStopInterval An integer specifying the how much time to add to the cohort_end when determining whether the event starts during cohort and ends after
- dechallengeEvaluationWindow A period of time evaluated for outcome recurrence after discontinuation of exposure, among patients with challenge outcomes
- numExposureEras Distinct number of exposure events (i.e. drug eras) in a given target cohort
- numPersonsExposed Distinct number of people exposed in target cohort. A person must have at least 1 day exposure to be included
- numCases Distinct number of persons in outcome cohort. A person must have at least 1 day of observation time to be included
- dechallengeAttempt Distinct count of people with observable time after discontinuation of the exposure era during which the challenge outcome occurred
- dechallengeFail Among people with challenge outcomes, the distinct number of people with outcomes during dechallengeEvaluationWindow
- dechallengeSuccess Among people with challenge outcomes, the distinct number of people without outcomes during the dechallengeEvaluationWindow
- rechallengeAttempt Number of people with a new exposure era after the occurrence of an outcome during a prior exposure era
- rechallengeFail Number of people with a new exposure era during which an outcome occurred, after the occurrence of an outcome during a prior exposure era
- rechallengeSuccess Number of people with a new exposure era during which an outcome did not occur, after the occurrence of an outcome during a prior exposure era
- pctDechallengeAttempt Percent of people with observable time after discontinuation of the exposure era during which the challenge outcome occurred
- pctDechallengeFail Among people with challenge outcomes, the percent of people without outcomes during the dechallengeEvaluationWindow
- pctDechallengeSuccess Among people with challenge outcomes, the percent of people with outcomes during dechallengeEvaluationWindow
- pctRechallengeAttempt Percent of people with a new exposure era after the occurrence of an outcome during a prior exposure era
- pctRechallengeFail Percent of people with a new exposure era during which an outcome did not occur, after the occurrence of an outcome during a prior exposure era
- pctRechallengeSuccess Percent of people with a new exposure era during which an outcome occurred, after the occurrence of an outcome during a prior exposure era

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCharacterizationDemographics()`, `getContinuousCaseSerie` `getContinuousRiskFactors()`, `getIncidenceRates()`, `getTargetBinaryFeatures()`, `getTargetContinuousFeatures` `getTargetCounts()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

dcrc <- getDechallengeRechallenge(
connectionHandler = connectionHandler,
schema = 'main'
)
```

`getExampleConnectionDetails`

create a connection detail for an example OHDSI results database

---

Description

This returns an object of class ‘ConnectionDetails‘ that lets you connect via ‘DatabaseConnector::connect()‘ to the example result database.

Usage

```
getExampleConnectionDetails(exdir = tempdir())
```

Arguments

`exdir`	a directory to unzip the example result data into. Default is tempdir().

Details

Finds the location of the example result database in the package and calls ‘DatabaseConnector::createConnectionDetails‘ to create a ‘ConnectionDetails‘ object for connecting to the database.

Value

An object of class ‘ConnectionDetails‘ with the details to connect to the example OHDSI result database

See Also

Other helper: `kableDark()`, `printReactable()`, `removeSpaces()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)
```

`getIncidenceRates` *Extract the cohort incidence result*

Description

This function extracts all incidence rates across databases in the results for specified target and outcome cohorts.

Usage

```
getIncidenceRates(
  connectionHandler,
  schema,
  ciTablePrefix = "ci_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`ciTablePrefix`	The prefix used for the cohort incidence results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- targetName the target cohort name
- targetId the target cohort unique identifier
- outcomeName the outcome name
- outcomeId the outcome unique identifier

- cleanWindow clean windown around outcome
- subgroupName name for the result subgroup
- ageGroupName name for the result age group
- genderName name for the result gender group
- startYear name for the result start year
- tarStartWith time at risk start reference
- tarStartOffset time at risk start offset from reference
- tarEndWith time at risk end reference
- tarEndOffset time at risk end offset from reference
- personsAtRiskPe persons at risk per event
- personsAtRisk persons at risk
- personDaysPe person days per event
- personDays person days
- personOutcomesPe person outcome per event
- personOutcomes persons outcome
- outcomesPe number of outcome per event
- outcomes number of outcome
- incidenceProportionP100p incidence proportion per 100 persons
- incidenceRateP100py incidence rate per 100 person years

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCharacterizationDemographics()`, `getContinuousCaseSerie` `getContinuousRiskFactors()`, `getDechallengeRechallenge()`, `getTargetBinaryFeatures()`, `getTargetContinuousFeatures()`, `getTargetCounts()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

ir <- getIncidenceRates(
connectionHandler = connectionHandler,
schema = 'main'
)
```

`getPredictionCohorts` *Extract a complete set of cohorts used in the prediction results*

Description

This function extracts the target and outcome cohorts used to develop any model in the results

Usage

```
getPredictionCohorts(
  connectionHandler,
  schema,
  plpTablePrefix = "plp_",
  cgTablePrefix = "cg_"
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`plpTablePrefix`	The prefix used for the patient level prediction results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables

Details

Specify the connectionHandler, the resultDatabaseSettings and any targetIds or outcomeIds to restrict models to

Value

Returns a data.frame with the columns:

- cohortId the cohort definition ID
- cohortName the name of the cohort
- type whether the cohort was used as a target or outcome cohort

See Also

Other Prediction: `getPredictionDiagnosticTable()`, `getPredictionDiagnostics()`, `getPredictionHyperParamSear` `getPredictionIntercept()`, `getPredictionModelDesigns()`, `getPredictionPerformanceTable()`, `getPredictionPerformances()`, `getPredictionTopPredictors()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

predCohorts <- getPredictionCohorts(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`getPredictionDiagnostics` *Extract the model design diagnostics for a specific development database*

Description

This function extracts the PROBAST diagnostics

Usage

```
getPredictionDiagnostics(
  connectionHandler,
  schema,
  plpTablePrefix = "plp_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  databaseTablePrefix = "",
  modelDesignId = NULL,
  threshold1_2 = 0.9
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`plpTablePrefix`	The prefix used for the patient level prediction results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`databaseTablePrefix`	The prefix for the database table either ” or ’plp_’
`modelDesignId`	The identifier for a model design to restrict results to
`threshold1_2`	A threshold for probast 1.2

Details

Specify the connectionHandler, the resultDatabaseSettings and (optionally) a modelDesignId and threshold1_2 a threshold value to use for the PROBAST 1.2

Value

Returns a data.frame with the columns:

- modelDesignId the unique identifier for the model design
- diagnosticId the unique identifier for diagnostic result
- developmentDatabaseName the name for the database used to develop the model
- developmentTargetName the name for the development target population
- developmentOutcomeName the name for the development outcome
- probast1_1 Were appropriate data sources used, e.g., cohort, RCT, or nested case-control study data?
- probast1_2 Were all inclusions and exclusions of paticipants appropriate?
- probast2_1 Were predictors defined and assessed in a similar way for all participants?
- probast2_2 Were predictors assessments made without knowledge of outcome data?
- probast2_3 All all predictors available at the time the model is intended to be used?
- probast3_4 Was the outcome defined and determined in a similar way for all participants?
- probast3_6 Was the time interval between predictor assessment and outcome determination appropriate?
- probast4_1 Were there a reasonable number of participants with the outcome?

See Also

Other Prediction: `getPredictionCohorts()`, `getPredictionDiagnosticTable()`, `getPredictionHyperParamSearch()` `getPredictionIntercept()`, `getPredictionModelDesigns()`, `getPredictionPerformanceTable()`, `getPredictionPerformances()`, `getPredictionTopPredictors()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

diag <- getPredictionDiagnostics(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getPredictionDiagnosticTable

Extract specific diagnostic table

Description

This function extracts the specified diagnostic table

Usage

```
getPredictionDiagnosticTable(
  connectionHandler,
  schema,
  plpTablePrefix = "plp_",
  table = "diagnostic_participants",
  diagnosticId = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`plpTablePrefix`	The prefix used for the patient level prediction results tables
`table`	The table to extract
`diagnosticId`	(optional) restrict to the input diagnosticId

Details

Specify the connectionHandler, the resultDatabaseSettings, the table of interest and (optionally) a diagnosticId to filter to

Value

Returns a data.frame with the specified table

See Also

Other Prediction: `getPredictionCohorts()`, `getPredictionDiagnostics()`, `getPredictionHyperParamSearch()`, `getPredictionIntercept()`, `getPredictionModelDesigns()`, `getPredictionPerformanceTable()`, `getPredictionPerformances()`, `getPredictionTopPredictors()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

diagPred <- getPredictionDiagnosticTable(
  connectionHandler = connectionHandler,
  schema = 'main',
  table = 'diagnostic_predictors'
)
```

`getPredictionHyperParamSearch`

Extract hyper parameters details

Description

This function extracts the hyper parameters details

Usage

```
getPredictionHyperParamSearch(
  connectionHandler,
  schema,
  plpTablePrefix = "plp_",
  modelDesignId = NULL,
  databaseId = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`plpTablePrefix`	The prefix used for the patient level prediction results tables
`modelDesignId`	The identifier for a model design to restrict to
`databaseId`	The identifier for the development database to restrict to

Details

Specify the connectionHandler, the resultDatabaseSettings, the modelDesignId and the databaseId

Value

Returns a data.frame with the columns:

- metric the hyperparameter optimization metric
- fold the fold in cross validation
- value the metric value for the fold with the specified hyperparameter combination

plus columns for all the hyperparameters and their values

See Also

Other Prediction: `getPredictionCohorts()`, `getPredictionDiagnosticTable()`, `getPredictionDiagnostics()`, `getPredictionIntercept()`, `getPredictionModelDesigns()`, `getPredictionPerformanceTable()`, `getPredictionPerformances()`, `getPredictionTopPredictors()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

hyperParams <- getPredictionHyperParamSearch(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`getPredictionIntercept` *Extract model interception (for logistic regression)*

Description

This function extracts the interception value

Usage

```
getPredictionIntercept(
  connectionHandler,
  schema,
  plpTablePrefix = "plp_",
  modelDesignId = NULL,
  databaseId = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`plpTablePrefix`	The prefix used for the patient level prediction results tables
`modelDesignId`	The identifier for a model design to restrict to
`databaseId`	The identifier for the development database to restrict to

Details

Specify the connectionHandler, the resultDatabaseSettings, the modelDesignId and the databaseId

Value

Returns a single value corresponding to the model intercept or NULL if not a logistic regression model

See Also

Other Prediction: `getPredictionCohorts()`, `getPredictionDiagnosticTable()`, `getPredictionDiagnostics()`, `getPredictionHyperParamSearch()`, `getPredictionModelDesigns()`, `getPredictionPerformanceTable()`, `getPredictionPerformances()`, `getPredictionTopPredictors()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

intercepts <- getPredictionIntercept(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`getPredictionModelDesigns` *Extract the model designs and aggregate performances for the prediction results*

Description

This function extracts the model design settings and min/max/mean AUROC values of the models developed using the model design across databases

Usage

```
getPredictionModelDesigns(
  connectionHandler,
  schema,
  plpTablePrefix = "plp_",
  cgTablePrefix = "cg_",
  targetIds = NULL,
  outcomeIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`plpTablePrefix`	The prefix used for the patient level prediction results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs

Details

Specify the connectionHandler, the resultDatabaseSettings and (optionally) any targetIds or outcomeIds to restrict model designs to

Value

Returns a data.frame with the columns:

- modelDesignId a unique identifier in the database for the model design
- modelType the type of classifier or surival model
- developmentTargetId a unique identifier for the development target ID
- developmentTargetName the name of the development target cohort
- developmentTargetJson the json of the target cohort
- developmentOutcomeId a unique identifier for the development outcome ID
- developmentOutcomeName the name of the development outcome cohort
- timeAtRisk the time at risk string
- developmentOutcomeJson the json of the outcome cohort
- covariateSettingsJson the covariate settings json
- populationSettingsJson the population settings json
- tidyCovariatesSettingsJson the tidy covariate settings json
- plpDataSettingsJson the plp data extraction settings json

- featureEngineeringSettingsJson the feature engineering settings json
- splitSettingsJson the split settings json
- sampleSettingsJson the sample settings json
- minAuroc the min AUROC value of models developed using the model design across databases
- meanAuroc the mean AUROC value of models developed using the model design across databases
- maxAuroc the max AUROC value of models developed using the model design across databases
- noDiagnosticDatabases the number of databases where the model design diagnostics were generated
- noDevelopmentDatabases the number of databases where the model design was used to develop models
- noValidationDatabases the number of databases where the models developed using the model design was externally validated

See Also

Other Prediction: `getPredictionCohorts()`, `getPredictionDiagnosticTable()`, `getPredictionDiagnostics()`, `getPredictionHyperParamSearch()`, `getPredictionIntercept()`, `getPredictionPerformanceTable()`, `getPredictionPerformances()`, `getPredictionTopPredictors()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

modDesign <- getPredictionModelDesigns(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getPredictionPerformances

Extract the model performances

Description

This function extracts the model performances

Usage

```
getPredictionPerformances(
  connectionHandler,
  schema,
  plpTablePrefix = "plp_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  databaseTablePrefix = "",
  modelDesignId = NULL,
  developmentDatabaseId = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`plpTablePrefix`	The prefix used for the patient level prediction results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`databaseTablePrefix`	A prefix to the database table, either ” or ’plp_’
`modelDesignId`	The identifier for a model design to restrict results to
`developmentDatabaseId`	The identifier for the development database to restrict results to

Details

Specify the connectionHandler, the resultDatabaseSettings and (optionally) a modelDesignId and/or developmentDatabaseId to restrict models to

Value

Returns a data.frame with the columns:

- performanceId the unique identifier for the performance
- modelDesignId the unique identifier for the model design
- developmentDatabaseId the unique identifier for the database used to develop the model
- validationDatabaseId the unique identifier for the database used to validate the model
- developmentTargetId the unique cohort id for the development target population
- developmentTargetName the name for the development target population
- developmentOutcomeId the unique cohort id for the development outcome
- developmentOutcomeName the name for the development outcome
- developmentDatabase the name for the database used to develop the model

- validationDatabase the name for the database used to validate the model
- validationTargetName the name for the validation target population
- validationOutcomeName the name for the validation outcome
- timeStamp the date/time when the analysis occurred
- auroc the test/validation AUROC value for the model
- auroc95lb the test/validation lower bound of the 95 percent CI AUROC value for the model
- auroc95ub the test/validation upper bound of the 95 percent CI AUROC value for the model
- calibrationInLarge the test/validation calibration in the large value for the model
- eStatistic the test/validation calibration e-statistic value for the model
- brierScore the test/validation brier value for the model
- auprc the test/validation discrimination AUPRC value for the model
- populationSize the test/validation population size used to develop the model
- outcomeCount the test/validation outcome count used to develop the model
- evalPercent the percentage of the development data used as the test set
- outcomePercent the outcome percent in the development data
- validationTimeAtRisk time at risk for the validation
- predictionResultType development or validation

See Also

Other Prediction: `getPredictionCohorts()`, `getPredictionDiagnosticTable()`, `getPredictionDiagnostics()`, `getPredictionHyperParamSearch()`, `getPredictionIntercept()`, `getPredictionModelDesigns()`, `getPredictionPerformanceTable()`, `getPredictionTopPredictors()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

perf <- getPredictionPerformances(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getPredictionPerformanceTable

Extract specific results table

Description

This function extracts the specified table

Usage

```
getPredictionPerformanceTable(
  connectionHandler,
  schema,
  plpTablePrefix = "plp_",
  table = "attrition",
  performanceId = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., 'main' for sqlite)
`plpTablePrefix`	The prefix used for the patient level prediction results tables
`table`	The table to extract
`performanceId`	(optional) restrict to the input performanceId

Details

Specify the connectionHandler, the resultDatabaseSettings, the table of interest and (optionally) a performanceId to filter to

Value

Returns a data.frame with the specified table

See Also

Other Prediction: `getPredictionCohorts()`, `getPredictionDiagnosticTable()`, `getPredictionDiagnostics()`, `getPredictionHyperParamSearch()`, `getPredictionIntercept()`, `getPredictionModelDesigns()`, `getPredictionPerformances()`, `getPredictionTopPredictors()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

attrition <- getPredictionPerformanceTable(
  connectionHandler = connectionHandler,
  schema = 'main',
  table = 'attrition'
)
```

`getPredictionTopPredictors`

Extract the top N predictors per model

Description

This function extracts the top N predictors per model from the prediction results tables

Usage

```
getPredictionTopPredictors(
  connectionHandler,
  schema,
  plpTablePrefix = "plp_",
  cgTablePrefix = "cg_",
  targetIds = NULL,
  outcomeIds = NULL,
  numberPredictors = 100
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`plpTablePrefix`	The prefix used for the patient level prediction results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs
`numberPredictors`	the number of predictors per model to return

Details

Specify the connectionHandler, the resultDatabaseSettings and (optionally) any targetIds or outcomeIds to restrict models to

Value

Returns a data.frame with the columns:

- databaseName the name of the database the model was developed on
- tarStartDay the time-at-risk start day
- tarStartAnchor whether the time-at-risk start is relative to cohort start or end
- tarEndDay the time-at-risk end day
- tarEndAnchor whether the time-at-risk end is relative to cohort start or end
- performanceId a unique identifier for the performance
- covariateId the FeatureExtraction covariate identifier
- covariateName the name of the covariate
- conceptId the covariates corresponding concept or 0
- covariateValue the feature importance or coefficient value
- covariateCount how many people had the covariate
- covariateMean the fraction of the target population with the covariate
- covariateStDev the standard deviation
- withNoOutcomeCovariateCount the number of the target population without the outcome with the covariate
- withNoOutcomeCovariateMean the fraction of the target population without the outcome with the covariate
- withNoOutcomeCovariateStDev the covariate standard deviation of the target population without the outcome
- withOutcomeCovariateCount the number of the target population with the outcome with the covariate
- withOutcomeCovariateMean the fraction of the target population with the outcome with the covariate
- withOutcomeCovariateStDev the covariate standard deviation of the target population with the outcome
- standardizedMeanDiff the standardized mean difference comparing the target population with outcome and without the outcome
- rn the row number showing the covariate rank

See Also

Other Prediction: `getPredictionCohorts()`, `getPredictionDiagnosticTable()`, `getPredictionDiagnostics()`, `getPredictionHyperParamSearch()`, `getPredictionIntercept()`, `getPredictionModelDesigns()`, `getPredictionPerformanceTable()`, `getPredictionPerformances()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

topPreds <- getPredictionTopPredictors(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetIds = 1,
  outcomeIds = 3
)
```

---

getSccsDiagnosticsData

Extract the self controlled case series (sccs) diagostic results

---

Description

This function extracts the sccs diagnostics that examine whether the analyses were sufficiently powered and checks for different types of bias.

Usage

```
getSccsDiagnosticsData(
  connectionHandler,
  schema,
  sccsTablePrefix = "sccs_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`sccsTablePrefix`	The prefix used for the cohort generator results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the database name
- analysisId the analysis unique identifier
- description an analysis description
- targetName the target name
- targetId the target cohort id
- outcomeName the outcome name
- outcomeId the outcome cohort id
- covariateName whether main or secondary analysis
- mdrr the maximum passable minimum detectable relative risk (mdrr) value. If the mdrr is greater than this the diagnostics will fail.
- ease The expected absolute systematic error (ease) measures residual bias.
- timeTrendP The p for whether the mean monthly ratio between observed and expected is no greater than 1.25.
- preExposureP One-sided p-value for whether the rate before expore is higher than after, against the null of no difference.
- mdrrDiagnostic whether the mdrr (power) diagnostic passed or failed.
- easeDiagnostic whether the ease diagnostic passed or failed.
- timeTrendDiagnostic Pass / warning / fail / not evaluated classification of the time trend (unstalbe months) diagnostic.
- preExposureDiagnostic Pass / warning / fail / not evaluated classification of the time trend (unstalbe months) diagnostic.
- unblind whether the results can be unblinded.
- unblindForEvidenceSynthesis whether the results can be unblinded for the meta analysis.
- summaryValue summary of diagnostics results. FAIL, PASS or number of warnings.

See Also

Other Estimation: `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `plotCmEstimates()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

sccsDiag <- getSccsDiagnosticsData(
```

```
    connectionHandler = connectionHandler,
    schema = 'main',
    targetIds = 1,
    outcomeIds = 3
)
```

`getSccsEstimation` *Extract the self controlled case series (sccs) results*

Description

This function extracts the single database sccs estimates

Usage

```
getSccsEstimation(
  connectionHandler,
  schema,
  sccsTablePrefix = "sccs_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`sccsTablePrefix`	The prefix used for the cohort generator results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the database name
- analysisId the analysis unique identifier
- description an analysis description
- targetName the target name
- targetId the target cohort id
- outcomeName the outcome name
- outcomeId the outcome cohort id
- covariateName whether main or secondary analysis
- outcomeSubjects The number of subjects with at least one outcome.
- outcomeEvents The number of outcome events.
- outcomeObservationPeriods The number of observation periods containing at least one outcome.
- covariateSubjects The number of subjects having the covariate.
- covariateDays The total covariate time in days.
- covariateEras The number of continuous eras of the covariate.
- covariateOutcomes The number of outcomes observed during the covariate time.
- observedDays The number of days subjects were observed.
- rr the relative risk
- ci95Lb the lower bound of the 95 percent confidence interval for the relative risk
- ci95Ub the upper bound of the 95 percent confidence interval for the relative risk
- p the p-value for the relative risk
- logRr the log of the relative risk
- seLogRr the standard error or the log of the relative risk
- calibratedRr the calibrated relative risk
- calibratedCi95Lb the lower bound of the 95 percent confidence interval for the calibrated relative risk
- calibratedCi95Ub the upper bound of the 95 percent confidence interval for the calibrated relative risk
- calibratedP the calibrated p-value
- calibratedLogRr the calibrated log of the relative risk
- calibratedSeLogRr the calibrated log of the relative risk standard error
- llr The log of the likelihood ratio (of the MLE vs the null hypothesis of no effect).
- mdrr The minimum detectable relative risk.
- unblind Whether the results can be unblinded

See Also

Other Estimation: `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getSccsDiagnosticsData()`, `getSccsMetaEstimation()`, `plotCmEstimates()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

sccsEst <- getSccsEstimation(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetIds = 1,
  outcomeIds = 3
)
```

`getSccsMetaEstimation` *Extract the self controlled case series (sccs) meta analysis results*

Description

This function extracts any meta analysis estimation results for sccs.

Usage

```
getSccsMetaEstimation(
  connectionHandler,
  schema,
  sccsTablePrefix = "sccs_",
  cgTablePrefix = "cg_",
  esTablePrefix = "es_",
  targetIds = NULL,
  outcomeIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`sccsTablePrefix`	The prefix used for the cohort generator results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`esTablePrefix`	The prefix used for the evidence synthesis results tables

`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs

Details

Specify the connectionHandler, the schema and the targetoutcome cohort IDs

Value

Returns a data.frame with the columns:

#'

- databaseName the database name
- analysisId the analysis unique identifier
- description an analysis description
- targetName the target name
- targetId the target cohort id
- outcomeName the outcome name
- outcomeId the outcome cohort id
- covariateName whether main or secondary analysis
- outcomeSubjects The number of subjects with at least one outcome.
- outcomeEvents The number of outcome events.
- outcomeObservationPeriods The number of observation periods containing at least one outcome.
- covariateSubjects The number of subjects having the covariate.
- covariateDays The total covariate time in days.
- covariateEras The number of continuous eras of the covariate.
- covariateOutcomes The number of outcomes observed during the covariate time.
- observedDays The number of days subjects were observed.
- calibratedRr the calibrated relative risk
- calibratedCi95Lb the lower bound of the 95 percent confidence interval for the calibrated relative risk
- calibratedCi95Ub the upper bound of the 95 percent confidence interval for the calibrated relative risk
- calibratedP the calibrated p-value
- calibratedOneSidedP the calibrated one sided p-value
- calibratedLogRr the calibrated log of the relative risk
- calibratedSeLogRr the calibrated log of the relative risk standard error
- nDatabases The number of databases included in the estimate.

See Also

Other Estimation: `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `plotCmEstimates()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

sccsMeta <- getSccsMetaEstimation(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetIds = 1,
  outcomeIds = 3
)
```

`getTargetBinaryFeatures` *Extract aggregate statistics of binary feature analysis IDs of interest for targets*

Description

This function extracts the feature extraction results for targets corresponding to specified target and outcome cohorts.

Usage

```
getTargetBinaryFeatures(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL,
  analysisIds = c(3)
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cTablePrefix`	The prefix used for the characterization results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs
`analysisIds`	The feature extraction analysis ID of interest (e.g., 201 is condition)

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- targetName the target cohort name
- targetId the target cohort unique identifier
- outcomeName the outcome name
- outcomeId the outcome unique identifier
- minPriorObservation the minimum required observation days prior to index for an entry
- outcomeWashoutDays patients with the outcome occurring within this number of days prior to index are excluded (NA means no exclusion)
- covariateName the name of the feature
- sumValue the number of cases who have the feature value of 1

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCharacterizationDemographics()`, `getContinuousCaseSerie` `getContinuousRiskFactors()`, `getDechallengeRechallenge()`, `getIncidenceRates()`, `getTargetContinuousFeatu` `getTargetCounts()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

tbf <- getTargetBinaryFeatures (
connectionHandler = connectionHandler,
schema = 'main'
)
```

`getTargetContinuousFeatures`

Extract aggregate statistics of continuous feature analysis IDs of interest for targets

---

Description

This function extracts the continuous feature extraction results for targets corresponding to specified target cohorts.

Usage

```
getTargetContinuousFeatures(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  analysisIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cTablePrefix`	The prefix used for the characterization results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetIds`	A vector of integers corresponding to the target cohort IDs
`analysisIds`	The feature extraction analysis ID of interest (e.g., 201 is condition)

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- targetName the target cohort name
- targetId the target cohort unique identifier
- minPriorObservation the minimum required observation days prior to index for an entry
- covariateName the name of the feature
- covariateId the id of the feature
- countValue the number of cases who have the feature
- minValue the minimum value observed for the feature
- maxValue the maximum value observed for the feature
- averageValue the mean value observed for the feature
- standardDeviation the standard deviation of the value observed for the feature
- medianValue the median value observed for the feature

- p10Value the 10th percentile of the value observed for the feature
- p25Value the 25th percentile of the value observed for the feature
- p75Value the 75th percentile of the value observed for the feature
- p90Value the 90th percentile of the value observed for the feature

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCharacterizationDemographics()`, `getContinuousCaseSeri` `getContinuousRiskFactors()`, `getDechallengeRechallenge()`, `getIncidenceRates()`, `getTargetBinaryFeatures(` `getTargetCounts()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

tcf <- getTargetContinuousFeatures(
connectionHandler = connectionHandler,
schema = 'main'
)
```

`getTargetCounts` *Extract the target cohort counts result*

Description

This function extracts target cohort counts across databases in the results for specified target and outcome cohorts.

Usage

```
getTargetCounts(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cTablePrefix`	The prefix used for the characterization results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- targetName the target cohort name
- targetId the target cohort unique identifier
- outcomeName the outcome name
- outcomeId the outcome unique identifier
- rowCount the number of entries in the cohort
- personCount the number of people in the cohort
- minPriorObservation the minimum required observation days prior to index for an entry
- outcomeWashoutDays patients with the outcome occurring within this number of days prior to index are excluded (NA means no exclusion)

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCharacterizationDemographics()`, `getContinuousCaseSerie` `getContinuousRiskFactors()`, `getDechallengeRechallenge()`, `getIncidenceRates()`, `getTargetBinaryFeatures(` `getTargetContinuousFeatures()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

tc <- getTargetCounts(
connectionHandler = connectionHandler,
schema = 'main'
)
```

`getTimeToEvent` *Extract the time to event result*

Description

This function extracts all time to event results across databases for specified target and outcome cohorts.

Usage

```
getTimeToEvent(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler that connects to the database and extracts sql queries. Create a connection handler via ‘ResultModelManager::ConnectionHandler$new()‘.
`schema`	The result database schema (e.g., ’main’ for sqlite)
`cTablePrefix`	The prefix used for the characterization results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default ’database_meta_data’)
`targetIds`	A vector of integers corresponding to the target cohort IDs
`outcomeIds`	A vector of integers corresponding to the outcome cohort IDs

Details

Specify the connectionHandler, the schema and the target/outcome cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- targetName the target cohort name
- targetId the target cohort unique identifier
- outcomeName the outcome name
- outcomeId the outcome unique identifier

- outcomeType Whether the outcome is the first or subsequent
- targetOutcomeType The interval that the outcome occurs
- timeToEvent The number of days from index
- numEvents The number of target cohort entries
- timeScale The correspondin time-scale

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCharacterizationDemographics()`, `getContinuousCaseSerie` `getContinuousRiskFactors()`, `getDechallengeRechallenge()`, `getIncidenceRates()`, `getTargetBinaryFeatures(` `getTargetContinuousFeatures()`, `getTargetCounts()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

tte <- getTimeToEvent(
connectionHandler = connectionHandler,
schema = 'main'
)
```

`kableDark` *output a nicely formatted html table*

Description

This returns a html table with the input data

Usage

```
kableDark(data, caption = NULL, position = NULL)
```

Arguments

`data`	A data.frame containing data of interest to show via a table
`caption`	A caption for the table
`position`	The position for the table if used within a quarto document. This is the "real" or say floating position for the latex table environment. The kable only puts tables in a table environment when a caption is provided. That is also the reason why your tables will be floating around if you specify captions for your table. Possible choices are h (here), t (top, default), b (bottom) and p (on a dedicated page).

Details

Input the data that you want to be shown via a dark html table

Value

An object of class ‘knitr_kable‘ that will show the data via a nice html table

See Also

Other helper: `getExampleConnectionDetails()`, `printReactable()`, `removeSpaces()`

Examples

```
kableDark(
data = data.frame(a=1,b=4),
caption = 'A made up table to demonstrate this function',
position = 'h'
)
```

`OhdsiReportGenerator` *OhdsiReportGenerator*

Description

A package for extracting analyses results and creating reports.

Author(s)

Maintainer: Jenna Reps `<jreps@its.jnj.com>`

Authors:

- Anthony Sena `<sena@ohdsi.org>`

See Also

Useful links:

- `https://ohdsi.github.io/OhdsiReportGenerator/`
- `https://github.com/OHDSI/OhdsiReportGenerator`
- Report bugs at `https://github.com/OHDSI/OhdsiReportGenerator/issues`

`plotAgeDistributions` *Plots the age distributions using the binary age groups*

Description

Creates bar charts for the target and case age groups.

Usage

```
plotAgeDistributions(
  ageData,
  riskWindowStart = "1",
  riskWindowEnd = "365",
  startAnchor = "cohort start",
  endAnchor = "cohort start"
)
```

Arguments

`ageData`	The age data extracted using 'getCharacterizationDemographics(type = 'age')'
`riskWindowStart`	The time at risk window start
`riskWindowEnd`	The time at risk window end
`startAnchor`	The anchor for the time at risk start
`endAnchor`	The anchor for the time at risk end

Details

Input the data returned from 'getCharacterizationDemographics(type = 'age')' and the time-at-risk

Value

Returns a ggplot with the distributions

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCharacterizationDemographics()`, `getContinuousCaseSeri`
`getContinuousRiskFactors()`, `getDechallengeRechallenge()`, `getIncidenceRates()`, `getTargetBinaryFeatures(`
`getTargetContinuousFeatures()`, `getTargetCounts()`, `getTimeToEvent()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

ageData <- getCharacterizationDemographics(
connectionHandler = connectionHandler,
schema = 'main',
targetId = 1,
outcomeId = 3,
type = 'age'
)

plotAgeDistributions(ageData = ageData)
```

`plotCmEstimates` *Plots the cohort method results for one analysis*

Description

Creates nice cohort method plots

Usage

```
plotCmEstimates(
  cmData,
  cmMeta = NULL,
  targetName,
  comparatorName,
  selectedAnalysisId
)
```

Arguments

`cmData`	The cohort method data
`cmMeta`	(optional) The cohort method evidence synthesis data
`targetName`	A friendly name for the target cohort
`comparatorName`	A friendly name for the comparator cohort
`selectedAnalysisId`	The analysis ID of interest to plot

Details

Input the cohort method data

Value

Returns a ggplot with the estimates

See Also

Other Estimation: `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

cmEst <- getCMEstimation(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetIds = 1,
  outcomeIds = 3
)
plotCmEstimates(
  cmData = cmEst,
  cmMeta = NULL,
  targetName = 'target',
  comparatorName = 'comp',
  selectedAnalysisId = 1
)
```

`plotSccsEstimates` *Plots the self controlled case series results for one analysis*

Description

Creates nice self controlled case series plots

Usage

```
plotSccsEstimates(sccsData, sccsMeta = NULL, targetName, selectedAnalysisId)
```

Arguments

`sccsData`	The self controlled case series data
`sccsMeta`	(optional) The self controlled case seriesd evidence synthesis data
`targetName`	A friendly name for the target cohort
`selectedAnalysisId`	The analysis ID of interest to plot

Details

Input the self controlled case series data

Value

Returns a ggplot with the estimates

See Also

Other Estimation: `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `plotCmEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

sccsEst <- getSccsEstimation(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetIds = 1,
  outcomeIds = 3
)
plotSccsEstimates(
  sccsData = sccsEst,
  sccsMeta = NULL,
  targetName = 'target',
  selectedAnalysisId = 1
)
```

`plotSexDistributions` *Plots the sex distributions using the sex features*

Description

Creates bar charts for the target and case sex.

Usage

```
plotSexDistributions(
  sexData,
  riskWindowStart = "1",
  riskWindowEnd = "365",
  startAnchor = "cohort start",
  endAnchor = "cohort start"
)
```

Arguments

`sexData`	The sex data extracted using 'getCharacterizationDemographics(type = 'sex')'
`riskWindowStart`	The time at risk window start
`riskWindowEnd`	The time at risk window end
`startAnchor`	The anchor for the time at risk start
`endAnchor`	The anchor for the time at risk end

Details

Input the data returned from 'getCharacterizationDemographics(type = 'sex')' and the time-at-risk

Value

Returns a ggplot with the distributions

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCharacterizationDemographics()`, `getContinuousCaseSerie` `getContinuousRiskFactors()`, `getDechallengeRechallenge()`, `getIncidenceRates()`, `getTargetBinaryFeatures(` `getTargetContinuousFeatures()`, `getTargetCounts()`, `getTimeToEvent()`, `plotAgeDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

sexData <- getCharacterizationDemographics(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetId = 1,
  outcomeId = 3,
  type = 'sex'
)
plotSexDistributions(sexData = sexData)
```

`printReactable` *prints a reactable in a quarto document*

Description

This function lets you print a reactable in a quarto document

Usage

```
printReactable(
  data,
  columns = NULL,
  groupBy = NULL,
  defaultPageSize = 20,
  highlight = TRUE,
  striped = TRUE
)
```

Arguments

`data`	The data for the table
`columns`	The formating for the columns
`groupBy`	A column or columns to group the table by
`defaultPageSize`	The number of rows in the table
`highlight`	whether to highlight the row of interest
`striped`	whether the rows change color to give a striped appearance

Details

Input the values for reactable::reactable

Value

Nothing but the html code for the table is printed (to be used in a quarto document)

See Also

Other helper: `getExampleConnectionDetails()`, `kableDark()`, `removeSpaces()`

Examples

```
printReactable(
data = data.frame(a=1,b=4)
)
```

`processCohorts` *Extract the cohort parents and children cohorts (cohorts derieved from the parent cohort)*

Description

This function lets you split the cohort data.frame into the parents and the children per parent.

Usage

```
processCohorts(cohort)
```

Arguments

`cohort`	The data.frame extracted using ‘getCohortDefinitions()‘

Details

Finds the parent cohorts and children cohorts

Value

Returns a list containing parents: a named vector of all the parent cohorts and cohortList: a list the same length as the parent vector with the first element containing all the children of the first parent cohort, the second element containing the children of the second parent, etc.

See Also

Other Cohorts: `getCohortDefinitions()`, `getCohortSubsetDefinitions()`

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)

cohortDef <- getCohortDefinitions(
  connectionHandler = connectionHandler,
  schema = 'main'
)

parents <- processCohorts(cohortDef)
```

`removeSpaces` *removeSpaces*

Description

Removes spaces and replaces with under scroll

Usage

```
removeSpaces(x)
```

Arguments

`x`	A string

Details

Removes spaces and replaces with under scroll

Value

A string without spaces

See Also

Other helper: `getExampleConnectionDetails()`, `kableDark()`, `printReactable()`

Examples

```
removeSpaces(' made up.   string')
```

Index