Package 'OhdsiReportGenerator'

November 13, 2026

Type Package

Title Observational Health Data Sciences and Informatics Report Generator

Version 2.3.2

Date 2026-11-13

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 (>= 4.1.0)

Imports CirceR, DatabaseConnector, forestploter, dplyr, ggplot2, grDevices, grid, gt, gtExtras, kableExtra, ParallelLogger, quarto, rlang, rmarkdown, SqlRender, tidyr

Suggests htmltools, knitr, markdown, ResultModelManager, RSQLite, testthat (>= 3.0.0)

Encoding UTF-8

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

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

Repository CRAN

Date/Publication 2026-11-13 22:20:06 UTC

Contents

.getCmVersion . . . 4
.getSccsVersion . . . 5
addTarColumn . . . 6
characterizationCompareBinary . . . 7
characterizationCompareContinuous . . . 8
createCharacterizationIndexes . . . 10
createCohortIndexes . . . 10
createIncidenceIndexes . . . 11
createPredictionReport . . . 12
createSccsIndexes . . . 13
formatBinaryCovariateName . . . 13
formatTargetPopulation . . . 14
generateFullReport . . . 15
generateSummaryPredictionReport . . . 17
getAggregateBinaryRiskFactors . . . 18
getAggregateContinuousRiskFactors . . . 20
getAnalysisCohorts . . . 22
getBinaryCaseSeries . . . 23
getBinaryRiskFactors . . . 25
getBinaryTargetBaseline . . . 26
getCaseCounts . . . 28
getCharacterizationCaseSettings . . . 31
getCharacterizationDemographics . . . 32
getCharacterizationTargetSettings . . . 34
getCmDiagnosticsData . . . 36
getCMEstimation . . . 38
getCmMetaEstimation . . . 40
getCmNegativeControlEstimates . . . 42
getCmOutcomes . . . 44
getCmPropensityModel . . . 45
getCmTable . . . 46
getCmTargets . . . 48
getCohortAttrition . . . 49
getCohortCounts . . . 50
getCohortDefinitions . . . 51
getCohortInclusionRules . . . 52
getCohortInclusionStats . . . 53
getCohortInclusionSummary . . . 54
getCohortMeta . . . 55
getCohortSubsetAttrition . . . 57
getCohortSubsetDefinitions . . . 58
getContinuousCaseSeries . . . 59
getContinuousRiskFactors . . . 61
getContinuousTargetBaseline . . . 62
getDatabaseDetails . . . 65
getDechallengeRechallenge . . . 66

getDechallengeRechallengeFails . . . 68
getEventDuration . . . 70
getExampleConnectionDetails . . . 72
getFullPredictionPerformances . . . 73
getIncidenceRates . . . 75
getIncidenceTargetSettings . . . 78
getNonCaseCounts . . . 79
getOutcomesUsedInCharacterization . . . 81
getOutcomesUsedInIncidence . . . 83
getOutcomeTable . . . 84
getPredictionAggregateTopPredictors . . . 87
getPredictionCohorts . . . 88
getPredictionCovariates . . . 90
getPredictionDiagnostics . . . 91
getPredictionDiagnosticTable . . . 93
getPredictionHyperParamSearch . . . 94
getPredictionIntercept . . . 95
getPredictionLift . . . 96
getPredictionModelDesigns . . . 98
getPredictionOutcomes . . . 100
getPredictionPerformances . . . 101
getPredictionPerformanceTable . . . 103
getPredictionTargets . . . 104
getPredictionTopPredictors . . . 106
getSccsDiagnosticsData . . . 108
getSccsEstimation . . . 110
getSccsMetaEstimation . . . 113
getSccsModel . . . 115
getSccsNegativeControlEstimates . . . 116
getSccsOutcomes . . . 118
getSccsTable . . . 119
getSccsTargets . . . 121
getSccsTimeToEvent . . . 122
getSubsetText . . . 123
getTargetCounts . . . 124
getTargetsUsedInCharacterization . . . 126
getTargetsUsedInIncidence . . . 127
getTargetTable . . . 128
getTarText . . . 131
getTimeToEvent . . . 132
getTreatmentPathways . . . 134
kableDark . . . 135
OhdsiReportGenerator . . . 136
plotAgeDistributions . . . 137
plotCmEstimates . . . 138
plotSccsEstimates . . . 139
plotSexDistributions . . . 141
processCohortDefinitionsForQuarto . . . 142

processCohorts 143
removeSpaces 144
restrictCohortDefinitionsForQuarto 144
viewIncidenceRate 146

Index **148**

`.getCmVersion` *An internal function to determine the version of CohortMethod used to store results*

Description

An internal function to determine the version of CohortMethod used to store results

Usage

```
.getCmVersion(connectionHandler, schema, cmTablePrefix = "cm_")
```

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

Details

Specify the connectionHandler, the schema and the prefixes. This query will attempt to identify if CohortMethod v6 was used by inspecing the migration table. When the migration_order is >= 3 then v6 of CohortMethod was used.

Value

A integer with the major version number of cohort method

See Also

Other Estimation: `.getSccsVersion()`, `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getCmNegativeControlEstimates()`, `getCmOutcomes()`, `getCmPropensityModel()`, `getCmTable()`, `getCmTargets()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `getSccsModel()`, `getSccsNegativeControlEstimates()`, `getSccsOutcomes()`, `getSccsTable()`, `getSccsTargets()`, `getSccsTimeToEvent()`, `plotCmEstimates()`, `plotSccsEstimates()`

`.getSccsVersion` *An internal function to determine the version of SCCS used to store results*

Description

An internal function to determine the version of SCCS used to store results

Usage

```
.getSccsVersion(connectionHandler, schema, sccsTablePrefix = "sccs_")
```

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

Details

Specify the connectionHandler, the schema and the prefixes. This query will attempt to identify if SelfControlledCaseSeries v6.1.4 was used by inspecing the migration table. When the migration_order is >= 3 then v6.1.4 of SelfControlledCaseSeries was used.

Value

A integer with the major version number of cohort method

See Also

Other Estimation: `.getCmVersion()`, `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getCmNegativeControlEstimates()`, `getCmOutcomes()`, `getCmPropensityModel()`, `getCmTable()`, `getCmTargets()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `getSccsModel()`, `getSccsNegativeControlEstimates()`, `getSccsOutcomes()`, `getSccsTable()`, `getSccsTargets()`, `getSccsTimeToEvent()`, `plotCmEstimates()`, `plotSccsEstimates()`

addTarColumn *addTarColumn*

Description

Finds the four TAR columns and creates a new column called tar that pastes the columns into a nice string

Usage

```
addTarColumn(data)
```

Arguments

data	The data.frame with the individual TAR columns that you want to combine into one column

Details

Create a friendly single tar column

Value

The data data.frame object with the tar column added if seperate TAR columns are found

See Also

Other helper: formatBinaryCovariateName(), formatTargetPopulation(), getExampleConnectionDetails(), getOutcomeTable(), getTarText(), getTargetTable(), kableDark(), removeSpaces()

Examples

```
addTarColumn(data.frame(
tarStartWith = 'cohort start',
tarStartOffset = 1,
tarEndWith = 'cohort start',
tarEndOffset = 0
))
```

`characterizationCompareBinary` *A function to extract cohort aggregate binary feature characterization results*

Description

A function to extract cohort aggregate binary feature characterization results

Usage

```
characterizationCompareBinary(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  characterizationTargetIds = NULL,
  databaseIds = NULL,
  minThreshold = 0
)
```

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')
`characterizationTargetIds`	(optional) The characterization target ids
`databaseIds`	(optional) One or more unique identifiers for the databases
`minThreshold`	The minimum fraction of the cohort that must have the feature for it to be reported

Details

Specify the connectionHandler, the schema and the target cohort ID and database id

Value

A data.frame with the characterization aggregate binary features for a specific cohort and database

See Also

Other Characterization: characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemog getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

binCohort <- characterizationCompareBinary(
  connectionHandler = connectionHandler,
  schema = 'main',
  characterizationTargetIds = 1,
  databaseIds = 'eunomia'
)
```

---

characterizationCompareContinuous

A function to extract cohort aggregate continuous feature characterization results

---

Description

A function to extract cohort aggregate continuous feature characterization results

Usage

```
characterizationCompareContinuous(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  characterizationTargetIds = NULL,
  databaseIds = NULL,
  minThreshold = 0
)
```

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')
characterizationTargetIds	(optional) The characterization target ids
databaseIds	(optional) One or more unique identifiers for the databases
minThreshold	The minimum fraction of the cohort that must have the feature for it to be reported

Details

Specify the connectionHandler, the schema and the target cohort ID and database id

Value

A data.frame with the characterization aggregate continuous features for a specific cohort and database

See Also

Other Characterization: characterizationCompareBinary(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemog getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

binCohort <- characterizationCompareContinuous(
  connectionHandler = connectionHandler,
  schema = 'main',
  characterizationTargetIds = 1,
  databaseIds = 'eunomia'
)
```

createCharacterizationIndexes *A function to add indexes to the Characterization results*

Description

A function to add indexes to the Characterization results

Usage

```
createCharacterizationIndexes(connectionHandler, schema, cTablePrefix = "c_")
```

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

Details

Specify the connectionHandler, the schema and the prefixes

See Also

Other Indexes: createCohortIndexes(), createIncidenceIndexes(), createSccsIndexes()

createCohortIndexes *A function to add indexes to the Cohort Generator results*

Description

A function to add indexes to the Cohort Generator results

Usage

```
createCohortIndexes(connectionHandler, schema, 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)
cgTablePrefix	The prefix used for the cohort generator results tables

Details

Specify the connectionHandler, the schema and the prefixes

See Also

Other Indexes: `createCharacterizationIndexes()`, `createIncidenceIndexes()`, `createSccsIndexes()`

`createIncidenceIndexes` *A function to add indexes to the incidence results*

Description

A function to add indexes to the incidence results

Usage

```
createIncidenceIndexes(connectionHandler, schema, ciTablePrefix = "ci_")
```

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

Details

Specify the connectionHandler, the schema and the prefixes

See Also

Other Indexes: `createCharacterizationIndexes()`, `createCohortIndexes()`, `createSccsIndexes()`

createPredictionReport *createPredictionReport*

Description

Generates a report for a given prediction model design

Usage

```
createPredictionReport(
  connectionHandler,
  schema,
  plpTablePrefix,
  databaseTablePrefix = plpTablePrefix,
  cgTablePrefix = plpTablePrefix,
  modelDesignId,
  output,
  intermediatesDir = file.path(tempdir(), "plp-prot"),
  outputFormat = "html_document"
)
```

Arguments

connectionHandler	The connection handler to the results database
schema	The result database schema
plpTablePrefix	The prediction table prefix
databaseTablePrefix	The database table name e.g., database_meta_data
cgTablePrefix	The cohort generator table prefix
modelDesignId	The model design ID of interest
output	The folder name where main.html will be save to
intermediatesDir	The work directory for rmarkdown
outputFormat	the type of outcome html_document or html_fragment

Details

Specify the connection handler to the result database, the schema name and the modelDesignId of interest to generate a html report summarizing the performance of models developed across databases.

Value

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

See Also

Other Reporting: `generateFullReport()`, `generateSummaryPredictionReport()`

`createSccsIndexes` *A function to add indexes to the SCCS results*

Description

A function to add indexes to the SCCS results

Usage

```
createSccsIndexes(connectionHandler, schema, sccsTablePrefix = "sccs_")
```

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

Details

Specify the connectionHandler, the schema and the prefixes

See Also

Other Indexes: `createCharacterizationIndexes()`, `createCohortIndexes()`, `createIncidenceIndexes()`

`formatBinaryCovariateName` *formatBinaryCovariateName*

Description

Removes the long part of the covariate name to make it friendly

Usage

```
formatBinaryCovariateName(data)
```

Arguments

`data`	The data.frame with the covariateName column

Details

Makes the covariateName more friendly and shorter

Value

The data data.frame object with the ovariateName column changed to be more friendly

See Also

Other helper: addTarColumn(), formatTargetPopulation(), getExampleConnectionDetails(), getOutcomeTable(), getTarText(), getTargetTable(), kableDark(), removeSpaces()

Examples

```
formatBinaryCovariateName(data.frame(
covariateName = c("fdfgfgf: dgdgff","made up test")
))
```

---

formatTargetPopulation

output a string summarizing the target population

Description

This returns a character that describes the target population

Usage

```
formatTargetPopulation(popSetting)
```

Arguments

popSetting	the target population setting

Details

Input the target population setting list

Value

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

See Also

Other helper: addTarColumn(), formatBinaryCovariateName(), getExampleConnectionDetails(), getOutcomeTable(), getTarText(), getTargetTable(), kableDark(), removeSpaces()

Examples

```
targetText <- formatTargetPopulation(
list(
limitToFirstInNDays = 365,
minPriorObservation = 365,
nestingName = 'Indication',
minAge = 12, maxAge = 9999,
studyEnd = 'Mar 2012', studyStart = NA,
genderConceptIds = NA
)
)
```

---

generateFullReport *generateFullReport*

Description

Generates a full report from a Strategus analysis

Usage

```
generateFullReport(
  server,
  username,
  password,
  dbms,
  resultsSchema = NULL,
  targetId = 1,
  outcomeIds = 3,
  comparatorIds = 2,
  indicationIds = NULL,
  restrictTargetToIndications = FALSE,
  cohortNames = c("target name", "outcome name", "comp name"),
  cohortIds = c(1, 3, 2),
  includeCI = TRUE,
  includeCharacterization = TRUE,
  includeCohortMethod = TRUE,
  includeSccs = TRUE,
  includePrediction = TRUE,
  webAPI = NULL,
  authMethod = NULL,
  webApiUsername = NULL,
  webApiPassword = NULL,
  outputLocation,
  outputName = paste0("full_report_", gsub(":", "_", gsub(" ", "_",
    as.character(date()))), ".html"),
```

```
  intermediateDir = tempdir(),
  pathToDriver = Sys.getenv("DATABASECONNECTOR_JAR_FOLDER")
)
```

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
outcomeIds	The cohort definition id for the outcome
comparatorIds	(optional) The cohort definition id for any comparator cohorts. If NULL the report will find and include all possible comparators in the results if includeCohortMethod is TRUE.
indicationIds	The cohort definition id for any indication cohorts to show in characterization. If no indication use NULL.
restrictTargetToIndications	If you only want the results for targets that are nested by the indicationIds set this to TRUE otherwise results for all children of the targetId will be generated.
cohortNames	Friendly names for any cohort used in the study
cohortIds	The corresponding Ids for the cohortNames
includeCI	Whether to include the cohort incidence slides
includeCharacterization	Whether to include the characterization slides
includeCohortMethod	Whether to include the cohort method slides
includeSccs	Whether to include the self controlled case series slides
includePrediction	Whether to include the patient level prediction slides
webAPI	The ATLAS web API to use for the characterization index breakdown (set to NULL to not include)
authMethod	The authorization method for the webAPI
webApiUsername	The username for the webAPI authorization
webApiPassword	The password for the webAPI authorization
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
pathToDriver	Path to a folder containing the JDBC driver JAR files.

Details

Specify the connection details to the result database and the schema name to generate the full report.

Value

An html document containing the full results for the target, comparators, indications and outcomes specified.

See Also

Other Reporting: `createPredictionReport()`, `generateSummaryPredictionReport()`

`generateSummaryPredictionReport` *generateSummaryPredictionReport*

Description

Generates a summary report for a given targets and outcomes

Usage

```
generateSummaryPredictionReport(
  connectionHandler,
  schema,
  targetIds = NULL,
  outcomeIds = NULL,
  plpTablePrefix = "plp_",
  databaseTablePrefix = "",
  cgTablePrefix = "cg_",
  outputFolder,
  outputFileName = "plp-summary.html",
  intermediatesDir = file.path(tempdir(), "plp-prot"),
  overwrite = FALSE
)
```

Arguments

`connectionHandler`	The connection handler to the results database
`schema`	The result database schema
`targetIds`	The target cohort IDs of interest
`outcomeIds`	The outcome cohort IDs of interest
`plpTablePrefix`	The prediction table prefix
`databaseTablePrefix`	The database table name e.g., database_meta_data

cgTablePrefix	The cohort generator table prefix
outputFolder	The folder name where file will be save to
outputFileName	The file name of the saved report
intermediatesDir	The work directory for rmarkdown
overwrite	whether to overwrite any existing file at the outputFolder/outputFileName

Details

Specify the connection handler to the result database, the schema name and the cohortId of interest to generate a html report summarizing the performance of prediction models in the database.

Value

A html file is created with the summary report

See Also

Other Reporting: `createPredictionReport()`, `generateFullReport()`

---

getAggregateBinaryRiskFactors

A function to extract binary risk factor across databases

Description

A function to extract binary risk factor across databases

Usage

```
getAggregateBinaryRiskFactors(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  characterizationTargetId = NULL,
  characterizationCaseId = NULL,
  outcomeId = NULL,
  outcomeWashout = NULL,
  analysisIds = c(3),
  riskWindowStart = NULL,
  riskWindowEnd = NULL,
  startAnchor = NULL,
  endAnchor = 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
characterizationTargetId	the characterization target id
characterizationCaseId	the characterization case id
outcomeId	Am integer corresponding to the outcome cohort ID
outcomeWashout	(optional) restrict to outcome washout
analysisIds	The feature extraction analysis ID of interest (e.g., 201 is condition)
riskWindowStart	(optional) A vector of time-at-risk risk window starts to restrict to
riskWindowEnd	(optional) A vector of time-at-risk risk window ends to restrict to
startAnchor	(optional) A vector of time-at-risk start anchors to restrict to
endAnchor	(optional) A vector of time-at-risk end anchors to restrict to

Details

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

Value

A data.frame with the aggregate binary risk factors

See Also

Other Characterization: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemog getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

```
rf <- getAggregateBinaryRiskFactors(
  connectionHandler = connectionHandler,
  schema = 'main',
  characterizationTargetId = 1,
  outcomeId = 3
)
```

`getAggregateContinuousRiskFactors` *A function to extract risk factors across add databases in results*

Description

A function to extract risk factors across add databases in results

Usage

```
getAggregateContinuousRiskFactors(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  characterizationTargetId = NULL,
  characterizationCaseId = NULL,
  outcomeId = NULL,
  outcomeWashout = NULL,
  analysisIds = NULL,
  riskWindowStart = NULL,
  riskWindowEnd = NULL,
  startAnchor = NULL,
  endAnchor = 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')
`characterizationTargetId`	The characterization target id

characterizationCaseId	The characterization case id
outcomeId	Am integer corresponding to the outcome cohort ID
outcomeWashout	(optional) the outcome washout to restrict to
analysisIds	The feature extraction analysis ID of interest (e.g., 201 is condition)
riskWindowStart	(optional) A vector of time-at-risk risk window starts to restrict to
riskWindowEnd	(optional) A vector of time-at-risk risk window ends to restrict to
startAnchor	(optional) A vector of time-at-risk start anchors to restrict to
endAnchor	(optional) A vector of time-at-risk end anchors to restrict to

Details

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

Value

A data.frame with the risk factors

See Also

Other Characterization: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

rf <- getAggregateContinuousRiskFactors(
  connectionHandler = connectionHandler,
  schema = 'main',
  characterizationTargetId = 1,
  outcomeId = 3
)
```

`getAnalysisCohorts` *Extracts the different analyses ran for each target and event cohorts*

Description

This function extracts analysis ids, events cohorts, and databases per target from treatment patterns

Usage

```
getAnalysisCohorts(connectionHandler, schema, tpTablePrefix = "tp_")
```

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)
`tpTablePrefix`	The prefix used for the cohort generator results tables

Details

Specify the connectionHandler and the schema

Value

Returns a data.frame with the columns:

- databaseName a concatinated string of all the database names ran for that analysis
- databaseId a concatinated string of all the database ids ran for that analysis
- analysisId the analysis ids for the treament patterns run
- targetCohortName the target cohort name
- targetCohortId the target cohort unique identifier
- eventCohortList a concatinated string of all the event cohort names ran for that target
- exitCohortList a concatinated string of all the exit cohort names ran for that target

See Also

Other TreatmentPatterns: `getEventDuration()`, `getTreatmentPathways()`

Examples

```
conDet <- getExampleConnectionDetails()

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

cohortAnalysis <- getAnalysisCohorts(
  connectionHandler = connectionHandler,
  schema = "main"
)
```

---

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",
  characterizationTargetId = NULL,
  characterizationCaseId = NULL,
  outcomeId = NULL,
  databaseIds = NULL,
  riskWindowStart = NULL,
  riskWindowEnd = NULL,
  startAnchor = NULL,
  endAnchor = NULL,
  conceptIds = NULL,
  minVal = NULL,
  outcomeWashout = 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')
characterizationTargetId	The characterization target id to restrict results to
characterizationCaseId	The characterization case id to restrict results to
outcomeId	Am integer corresponding to the outcome cohort ID
databaseIds	(optional) One or more unique identifiers for the databases
riskWindowStart	(optional) A riskWindowStart to restrict to
riskWindowEnd	(optional) A riskWindowEnd to restrict to
startAnchor	(optional) A startAnchor to restrict to
endAnchor	(optional) An endAnchor to restrict to
conceptIds	(optional) An conceptIds to restrict to
minVal	(optional) the minimum averageVal to extract
outcomeWashout	(optional) the outcomeWashout to restrict to

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: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemog getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

cs <- getBinaryCaseSeries(
  connectionHandler = connectionHandler,
  schema = 'main',
  characterizationTargetId = 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",
  characterizationTargetId = NULL,
  characterizationCaseId = NULL,
  outcomeId = NULL,
  outcomeWashout = NULL,
  databaseId = NULL,
  analysisIds = c(3),
  riskWindowStart = NULL,
  riskWindowEnd = NULL,
  startAnchor = NULL,
  endAnchor = 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')
`characterizationTargetId`	the characterization target id
`characterizationCaseId`	the characterization case id
`outcomeId`	Am integer corresponding to the outcome cohort ID
`outcomeWashout`	(optional) restrict to outcome washout
`databaseId`	The database ID to restrict results to
`analysisIds`	The feature extraction analysis ID of interest (e.g., 201 is condition)

riskWindowStart	(optional) A vector of time-at-risk risk window starts to restrict to
riskWindowEnd	(optional) A vector of time-at-risk risk window ends to restrict to
startAnchor	(optional) A vector of time-at-risk start anchors to restrict to
endAnchor	(optional) A vector of time-at-risk end anchors to restrict to

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: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemog getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

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

---

getBinaryTargetBaseline

Extract the aggregate covariates for the target ids of interest

---

Description

This function extracts the specified covariates for the specified targets

Usage

```
getBinaryTargetBaseline(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  characterizationTargetIds = NULL,
  analysisIds = NULL,
  covariateIds = NULL,
  conceptIds = NULL,
  databaseIds = NULL,
  includeNames = TRUE,
  minThreshold = 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')
characterizationTargetIds	The characterization target cohort ids of interest
analysisIds	The analysisIds of the covariate to restrict results to
covariateIds	The covariateIds to restict results to
conceptIds	The conceptIds of the covariate to restrict results to
databaseIds	The databaseIds of the covariate to restrict results to
includeNames	Whether to add database and cohort names (setting to FALSE will make extraction quicker)
minThreshold	(optional) The minimum average value for results to be returned

Details

Specify the connectionHandler, the schema and the target cohort IDs

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- databaseId the unique identifier of the database

- targetName the target cohort name
- targetId the target cohort unique identifier
- minPriorObservation the
- limitToFirstINDays the
- covariateName the
- covariateId the
- analysisId the
- sumValue the
- averageValue the

See Also

Other Characterization: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemograp getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

btb <- getBinaryTargetBaseline(
 connectionHandler = connectionHandler,
 schema = 'main',
 characterizationTargetIds = 1
)
```

---

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",
  characterizationTargetIds = NULL,
  characterizationCaseIds = NULL,
  outcomeIds = NULL,
  databaseIds = NULL,
  riskWindowStart = NULL,
  riskWindowEnd = NULL,
  startAnchor = NULL,
  endAnchor = NULL,
  includeNames = TRUE,
  outcomeWashout = 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')
characterizationTargetIds	The characterization target cohort ids of interest
characterizationCaseIds	The characterization case ids of interest
outcomeIds	A vector of integers corresponding to the outcome cohort IDs
databaseIds	(optional) A vector of database IDs to restrict to
riskWindowStart	(optional) A vector of time-at-risk risk window starts to restrict to
riskWindowEnd	(optional) A vector of time-at-risk risk window ends to restrict to
startAnchor	(optional) A vector of time-at-risk start anchors to restrict to
endAnchor	(optional) A vector of time-at-risk end anchors to restrict to
includeNames	whether to add the database names and cohort names
outcomeWashout	(option) whether to restrict to outcomeWashout

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
- databaseId the unique identifier of the database
- characterizationCaseId the unique identifier of target, outcome and TAR combination
- 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: `characterizationCompareBinary()`, `characterizationCompareContinuous()`, `getAggregateBinaryRiskFactors()`, `getAggregateContinuousRiskFactors()`, `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getBinaryTargetBaseline()`, `getCharacterizationCaseSettings()`, `getCharacterizationDemographics()`, `getCharacterizationTargetSettings()`, `getContinuousCaseSeries()`, `getContinuousRiskFactors()`, `getContinuousTargetBaseline()`, `getDechallengeRechallenge()`, `getDechallengeRechallengeFails()`, `getIncidenceRates()`, `getIncidenceTargetSettings()`, `getNonCaseCounts()`, `getOutcomesUsedInCharacterization()`, `getOutcomesUsedInIncidence()`, `getTargetCounts()`, `getTargetsUsedInCharacterization()`, `getTargetsUsedInIncidence()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`, `viewIncidenceRate()`

Examples

```
conDet <- getExampleConnectionDetails()

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

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

`getCharacterizationCaseSettings` *A function to extract the case settings found in characterization*

Description

A function to extract the case settings found in characterization

Usage

```
getCharacterizationCaseSettings(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  characterizationTargetIds = NULL,
  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
`characterizationTargetIds`	optional vector of characterizationTargetIds to restrict to
`targetIds`	optional vector of targetIds to restrict to
`outcomeIds`	optional vector of outcomeIds to restrict to

Details

Specify the connectionHandler, the schema and the prefixes

Value

A data.frame with the characterization target cohort ids, names and inclusion criteria.

See Also

Other Characterization: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

caseCohorts <- getCharacterizationCaseSettings(
  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",
  characterizationTargetId = 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')
characterizationTargetId	the characterization target 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
- databaseId the unique identifier 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: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
# example code

conDet <- getExampleConnectionDetails()

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

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

---

getCharacterizationTargetSettings

A function to extarct the targets settings found in characterization

---

Description

A function to extarct the targets settings found in characterization

Usage

```
getCharacterizationTargetSettings(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  characterizationTargetIds = NULL,
  targetIds = NULL,
  addDatabaseDetails = FALSE,
  databaseTable = "database_meta_data"
)
```

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
characterizationTargetIds	optional vector of characterizationTargetIds to restrict to
targetIds	optional vector of targetIds to restrict to
addDatabaseDetails	whether to add a databaseName and databaseId string
databaseTable	The name of the table with the database details (default 'database_meta_data')

Details

Specify the connectionHandler, the schema and the prefixes

Value

A data.frame with the characterization target cohort ids, names and inclusion criteria.

See Also

Other Characterization: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemographics(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

targetCohorts <- getCharacterizationTargetSettings(
  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,
  indicationIds = NULL,
  analysisIds = NULL,
  databaseIds = 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
`indicationIds`	A vector of cohort ids for the indication (nesting id) to restrict to
`analysisIds`	An optional vector of analysisIds to filter to
`databaseIds`	An optional vector of databaseIds to filter to

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
- databaseId the unqiue identifier 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.
- unblindForEvidenceSynthesis whether the results can be unblinded for the meta analysis.
- unblind whether the results can be unblinded.
- summaryValue summary of diagnostics results. FAIL, PASS or number of warnings.

See Also

Other Estimation: `.getCmVersion()`, `.getSccsVersion()`, `getCMEstimation()`, `getCmMetaEstimation()`, `getCmNegativeControlEstimates()`, `getCmOutcomes()`, `getCmPropensityModel()`, `getCmTable()`, `getCmTargets()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `getSccsModel()`, `getSccsNegativeControlEstimates()`, `getSccsOutcomes()`, `getSccsTable()`, `getSccsTargets()`, `getSccsTimeToEvent()`, `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,
  indicationIds = 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
`indicationIds`	A vector of cohort ids for the indication (nesting id) to restrict to
`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
- databaseId the unqiue identifier 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
- Unblind Whether the results passed diagnostics and were unblinded
- unblindForEvidenceSynthesis whether the results can be unblinded for the meta analysis.
- targetEstimator ...

See Also

Other Estimation: `.getCmVersion()`, `.getSccsVersion()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getCmNegativeControlEstimates()`, `getCmOutcomes()`, `getCmPropensityModel()`, `getCmTable()`, `getCmTargets()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `getSccsModel()`, `getSccsNegativeControlEstimates()`, `getSccsOutcomes()`, `getSccsTable()`, `getSccsTargets()`, `getSccsTimeToEvent()`, `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,
  indicationIds = NULL,
  comparatorIds = NULL,
  includeOneSidedP = TRUE
)
```

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

`indicationIds` A vector of cohort ids for the indication (nesting id) to restrict to

`comparatorIds` A vector of integers corresponding to the comparator cohort IDs

`includeOneSidedP`
This lets you extract from older results that do not have the one sided p by setting this to FALSE

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
- unblind whether the results can be unblinded.
- nDatabases the number of databases included

See Also

Other Estimation: .getCmVersion(), .getSccsVersion(), getCMEstimation(), getCmDiagnosticsData(), getCmNegativeControlEstimates(), getCmOutcomes(), getCmPropensityModel(), getCmTable(), getCmTargets(), getSccsDiagnosticsData(), getSccsEstimation(), getSccsMetaEstimation(), getSccsModel(), getSccsNegativeControlEstimates(), getSccsOutcomes(), getSccsTable(), getSccsTargets(), getSccsTimeToEvent(), plotCmEstimates(), plotSccsEstimates()

Examples

```
conDet <- getExampleConnectionDetails()

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

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

---

getCmNegativeControlEstimates

Extract the cohort method negative controls

Description

This function extracts the cohort method negative control table.

Usage

```
getCmNegativeControlEstimates(
  connectionHandler,
  schema,
  cmTablePrefix = "cm_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  comparatorIds = NULL,
  indicationIds = NULL,
  analysisIds = NULL,
  databaseIds = NULL,
  excludePositiveControls = TRUE
)
```

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
comparatorIds	A vector of integers corresponding to the comparator cohort IDs
indicationIds	The indications that the target & comparator was nested to
analysisIds	the analysis IDs to restrict to
databaseIds	the database IDs to restrict to
excludePositiveControls	Whether to exclude the positive controls

Details

Specify the connectionHandler, the schema and optionally the target/comparator/outcome/analysis/database IDs

Value

Returns a data.frame with the cohort method negative controls

See Also

Other Estimation: `.getCmVersion()`, `.getSccsVersion()`, `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getCmOutcomes()`, `getCmPropensityModel()`, `getCmTable()`, `getCmTargets()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `getSccsModel()`, `getSccsNegativeControlEstimates()`, `getSccsOutcomes()`, `getSccsTable()`, `getSccsTargets()`, `getSccsTimeToEvent()`, `plotCmEstimates()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

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

cmNc <- getCmNegativeControlEstimates(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`getCmOutcomes` *A function to extract the outcomes found in cohort method*

Description

A function to extract the outcomes found in cohort method

Usage

```
getCmOutcomes(
  connectionHandler,
  schema,
  cmTablePrefix = "cm_",
  cgTablePrefix = "cg_",
  targetId = 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
`targetId`	An integer corresponding to the target cohort ID

Details

Specify the connectionHandler, the schema and the prefixes

Value

A data.frame with the cohort method outcome ids and names.

See Also

Other Estimation: `.getCmVersion()`, `.getSccsVersion()`, `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getCmNegativeControlEstimates()`, `getCmPropensityModel()`, `getCmTable()`, `getCmTargets()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `getSccsModel()`, `getSccsNegativeControlEstimates()`, `getSccsOutcomes()`, `getSccsTable()`, `getSccsTargets()`, `getSccsTimeToEvent()`, `plotCmEstimates()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

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

outcomes <- getCmOutcomes(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`getCmPropensityModel` *Extract the cohort method model*

Description

This function extracts the cohort method model.

Usage

```
getCmPropensityModel(
  connectionHandler,
  schema,
  cmTablePrefix = "cm_",
  targetId = NULL,
  comparatorId = NULL,
  indicationId = NULL,
  analysisId = 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)
`cmTablePrefix`	The prefix used for the cohort method results tables
`targetId`	An integer corresponding to the target cohort ID
`comparatorId`	the comparator ID of interest
`indicationId`	The indications that the target & comparator was nested to
`analysisId`	the analysis ID to restrict to
`databaseId`	the database ID to restrict to

Details

Specify the connectionHandler, the schema and optionally the target/comparator/analysis/database IDs

Value

Returns a data.frame with the cohort method model

See Also

Other Estimation: .getCmVersion(), .getSccsVersion(), getCMEstimation(), getCmDiagnosticsData(), getCmMetaEstimation(), getCmNegativeControlEstimates(), getCmOutcomes(), getCmTable(), getCmTargets(), getSccsDiagnosticsData(), getSccsEstimation(), getSccsMetaEstimation(), getSccsModel(), getSccsNegativeControlEstimates(), getSccsOutcomes(), getSccsTable(), getSccsTargets(), getSccsTimeToEvent(), plotCmEstimates(), plotSccsEstimates()

Examples

```
conDet <- getExampleConnectionDetails()

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

cmModel <- getCmPropensityModel(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getCmTable *Extract the cohort method table specified*

Description

This function extracts the specific cohort method table.

Usage

```
getCmTable(
  connectionHandler,
  schema,
  table = c("attrition", "follow_up_dist", "interaction_result", "covariate_balance",
    "kaplan_meier_dist", "likelihood_profile", "preference_score_dist",
    "propensity_model", "shared_covariate_balance")[1],
  cmTablePrefix = "cm_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  indicationIds = NULL,
```

```
  outcomeIds = NULL,
  comparatorIds = NULL,
  analysisIds = NULL,
  databaseIds = 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)
table	The result table to extract
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
indicationIds	The indications that the target & comparator was nested to
outcomeIds	A vector of integers corresponding to the outcome cohort IDs
comparatorIds	A vector of integers corresponding to the comparator cohort IDs
analysisIds	the analysis IDs to restrict to
databaseIds	the database IDs to restrict to

Details

Specify the connectionHandler, the schema and optionally the target/comparator/outcome/analysis/database IDs

Value

Returns a data.frame with the cohort method requested table

See Also

Other Estimation: `.getCmVersion()`, `.getSccsVersion()`, `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getCmNegativeControlEstimates()`, `getCmOutcomes()`, `getCmPropensityModel()`, `getCmTargets()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `getSccsModel()`, `getSccsNegativeControlEstimates()`, `getSccsOutcomes()`, `getSccsTable()`, `getSccsTargets()`, `getSccsTimeToEvent()`, `plotCmEstimates()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

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

cmTable <- getCmTable(
```

```
  connectionHandler = connectionHandler,
  schema = 'main',
  table = 'attrition'
)
```

`getCmTargets` *A function to extract the targets found in cohort method*

Description

A function to extract the targets found in cohort method

Usage

```
getCmTargets(
  connectionHandler,
  schema,
  cmTablePrefix = "cm_",
  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)
`cmTablePrefix`	The prefix used for the cohort method results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables

Details

Specify the connectionHandler, the schema and the prefixes

Value

A data.frame with the cohort method target cohort ids and names.

See Also

Other Estimation: `.getCmVersion()`, `.getSccsVersion()`, `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getCmNegativeControlEstimates()`, `getCmOutcomes()`, `getCmPropensityModel()`, `getCmTable()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `getSccsModel()`, `getSccsNegativeControlEstimates()`, `getSccsOutcomes()`, `getSccsTable()`, `getSccsTargets()`, `getSccsTimeToEvent()`, `plotCmEstimates()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

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

cohorts <- getCmTargets(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`getCohortAttrition` *Get cohort attrition*

Description

Retrieves attrition information for specified cohorts from the database.

Usage

```
getCohortAttrition(
  connectionHandler,
  schema,
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  cohortIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler object.
`schema`	The database schema name.
`cgTablePrefix`	Prefix for cohort generator tables. Default is 'cg_'.
`databaseTable`	Name of the database metadata table. Default is 'database_meta_data'.
`cohortIds`	Optional vector of cohort IDs to filter.

Value

A data.frame with attrition details for each cohort.

See Also

Other Cohorts: `getCohortCounts()`, `getCohortDefinitions()`, `getCohortInclusionRules()`, `getCohortInclusionStats()`, `getCohortInclusionSummary()`, `getCohortMeta()`, `getCohortSubsetAttrition()`, `getCohortSubsetDefinitions()`, `getSubsetText()`, `processCohortDefinitionsForQuarto()`, `processCohorts()`, `restrictCohortDefinitionsForQuarto()`

getCohortCounts *Extract the cohort counts*

Description

This function extracts all cohort counts for the cohorts of interest.

Usage

```
getCohortCounts(
  connectionHandler,
  schema,
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  cohortIds = 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
databaseTable	The name of the table with the database details (default 'database_meta_data')
cohortIds	Optionally a list of cohortIds to restrict to

Details

Specify the connectionHandler, the schema and the cohort IDs

Value

Returns a data.frame with the cohort inclusion rules

See Also

Other Cohorts: getCohortAttrition(), getCohortDefinitions(), getCohortInclusionRules(), getCohortInclusionStats(), getCohortInclusionSummary(), getCohortMeta(), getCohortSubsetAttrition(), getCohortSubsetDefinitions(), getSubsetText(), processCohortDefinitionsForQuarto(), processCohorts(), restrictCohortDefinitionsForQuarto()

Examples

```
conDet <- getExampleConnectionDetails()

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

cohortMeta <- getCohortCounts(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`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: `getCohortAttrition()`, `getCohortCounts()`, `getCohortInclusionRules()`, `getCohortInclusionStats()`, `getCohortInclusionSummary()`, `getCohortMeta()`, `getCohortSubsetAttrition()`, `getCohortSubsetDefinitions()`, `getSubsetText()`, `processCohortDefinitionsForQuarto()`, `processCohorts()`, `restrictCohortDefinitionsForQuarto()`

Examples

```
conDet <- getExampleConnectionDetails()

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

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

---

getCohortInclusionRules

Extract the cohort inclusion rules

Description

This function extracts all cohort inclusion rules for the cohorts of interest.

Usage

```
getCohortInclusionRules(
  connectionHandler,
  schema,
  cgTablePrefix = "cg_",
  cohortIds = 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
`cohortIds`	Optionally a list of cohortIds to restrict to

Details

Specify the connectionHandler, the schema and the cohort IDs

Value

Returns a data.frame with the cohort inclusion rules

See Also

Other Cohorts: `getCohortAttrition()`, `getCohortCounts()`, `getCohortDefinitions()`, `getCohortInclusionStats()`, `getCohortInclusionSummary()`, `getCohortMeta()`, `getCohortSubsetAttrition()`, `getCohortSubsetDefinitions()`, `getSubsetText()`, `processCohortDefinitionsForQuarto()`, `processCohorts()`, `restrictCohortDefinitionsForQua`

Examples

```
conDet <- getExampleConnectionDetails()

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

cohortInclsuionsRules <- getCohortInclusionRules(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getCohortInclusionStats

Extract the cohort inclusion stats

Description

This function extracts all cohort inclusion stats for the cohorts of interest.

Usage

```
getCohortInclusionStats(
  connectionHandler,
  schema,
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  cohortIds = 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
`databaseTable`	The name of the table with the database details (default 'database_meta_data')
`cohortIds`	Optionally a list of cohortIds to restrict to

Details

Specify the connectionHandler, the schema and the cohort IDs

Value

Returns a data.frame with the cohort inclusion stats

See Also

Other Cohorts: getCohortAttrition(), getCohortCounts(), getCohortDefinitions(), getCohortInclusionRules()
getCohortInclusionSummary(), getCohortMeta(), getCohortSubsetAttrition(), getCohortSubsetDefinitions(),
getSubsetText(), processCohortDefinitionsForQuarto(), processCohorts(), restrictCohortDefinitionsForQua

Examples

```
conDet <- getExampleConnectionDetails()

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

cohortInclsuionsStats <- getCohortInclusionStats(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`getCohortInclusionSummary`

Extract the cohort inclusion summary

Description

This function extracts all cohort inclusion summary for the cohorts of interest.

Usage

```
getCohortInclusionSummary(
  connectionHandler,
  schema,
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  cohortIds = 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
databaseTable	The name of the table with the database details (default 'database_meta_data')
cohortIds	Optionally a list of cohortIds to restrict to

Details

Specify the connectionHandler, the schema and the cohort IDs

Value

Returns a data.frame with the cohort inclusion rules

See Also

Other Cohorts: getCohortAttrition(), getCohortCounts(), getCohortDefinitions(), getCohortInclusionRules()
getCohortInclusionStats(), getCohortMeta(), getCohortSubsetAttrition(), getCohortSubsetDefinitions(),
getSubsetText(), processCohortDefinitionsForQuarto(), processCohorts(), restrictCohortDefinitionsForQua

Examples

```
conDet <- getExampleConnectionDetails()

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

cohortInclsuionsSummary <- getCohortInclusionSummary(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getCohortMeta *Extract the cohort meta*

Description

This function extracts all cohort meta for the cohorts of interest.

Usage

```
getCohortMeta(
  connectionHandler,
  schema,
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  cohortIds = 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
databaseTable	The name of the table with the database details (default 'database_meta_data')
cohortIds	Optionally a list of cohortIds to restrict to

Details

Specify the connectionHandler, the schema and the cohort IDs

Value

Returns a data.frame with the cohort inclusion rules

See Also

Other Cohorts: getCohortAttrition(), getCohortCounts(), getCohortDefinitions(), getCohortInclusionRules() getCohortInclusionStats(), getCohortInclusionSummary(), getCohortSubsetAttrition(), getCohortSubsetDefinitions(), getSubsetText(), processCohortDefinitionsForQuarto(), processCohorts(), restrictCohortDefinitionsForQuarto()

Examples

```
conDet <- getExampleConnectionDetails()

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

cohortMeta <- getCohortMeta(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`getCohortSubsetAttrition` *Get cohort subset attrition*

Description

Retrieves attrition information for specified cohort subsets from the database.

Usage

```
getCohortSubsetAttrition(
  connectionHandler,
  schema,
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  cohortIds = NULL
)
```

Arguments

`connectionHandler`	A connection handler object.
`schema`	The database schema name.
`cgTablePrefix`	Prefix for cohort generator tables. Default is 'cg_'.
`databaseTable`	Name of the database metadata table. Default is 'database_meta_data'.
`cohortIds`	Optional vector of cohort IDs to filter.

Value

A data.frame with attrition details for each cohort subset.

See Also

Other Cohorts: `getCohortAttrition()`, `getCohortCounts()`, `getCohortDefinitions()`, `getCohortInclusionRules()`, `getCohortInclusionStats()`, `getCohortInclusionSummary()`, `getCohortMeta()`, `getCohortSubsetDefinitions()`, `getSubsetText()`, `processCohortDefinitionsForQuarto()`, `processCohorts()`, `restrictCohortDefinitionsForQua`

`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: `getCohortAttrition()`, `getCohortCounts()`, `getCohortDefinitions()`, `getCohortInclusionRules()`, `getCohortInclusionStats()`, `getCohortInclusionSummary()`, `getCohortMeta()`, `getCohortSubsetAttrition()`, `getSubsetText()`, `processCohortDefinitionsForQuarto()`, `processCohorts()`, `restrictCohortDefinitionsForQua`

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",
  characterizationTargetId = NULL,
  characterizationCaseId = NULL,
  outcomeId = NULL,
  databaseIds = NULL,
  riskWindowStart = NULL,
  riskWindowEnd = NULL,
  startAnchor = NULL,
  endAnchor = NULL,
  outcomeWashout = 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')
`characterizationTargetId`	The characterization target id to restrict results to
`characterizationCaseId`	The characterization case id to restrict results to

outcomeId	Am integer corresponding to the outcome cohort ID
databaseIds	(optional) One or more unique identifiers for the databases
riskWindowStart	(optional) A riskWindowStart to restrict to
riskWindowEnd	(optional) A riskWindowEnd to restrict to
startAnchor	(optional) A startAnchor to restrict to
endAnchor	(optional) An endAnchor to restrict to
outcomeWashout	(optional) the outcomeWashout to restrict to

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: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

cs <- getContinuousCaseSeries(
  connectionHandler = connectionHandler,
  schema = 'main',
  characterizationTargetId = 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",
  characterizationTargetId = NULL,
  characterizationCaseId = NULL,
  outcomeId = NULL,
  outcomeWashout = NULL,
  analysisIds = NULL,
  databaseIds = NULL,
  riskWindowStart = NULL,
  riskWindowEnd = NULL,
  startAnchor = NULL,
  endAnchor = 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')
`characterizationTargetId`	The characterization target id
`characterizationCaseId`	The characterization case id
`outcomeId`	Am integer corresponding to the outcome cohort ID
`outcomeWashout`	(optional) the outcome washout to restrict to
`analysisIds`	The feature extraction analysis ID of interest (e.g., 201 is condition)

databaseIds	(optional) A vector of database IDs to restrict to
riskWindowStart	(optional) A vector of time-at-risk risk window starts to restrict to
riskWindowEnd	(optional) A vector of time-at-risk risk window ends to restrict to
startAnchor	(optional) A vector of time-at-risk start anchors to restrict to
endAnchor	(optional) A vector of time-at-risk end anchors to restrict to

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: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

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

---

getContinuousTargetBaseline

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

```
getContinuousTargetBaseline(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  characterizationTargetIds = NULL,
  analysisIds = NULL,
  databaseIds = NULL,
  includeNames = TRUE,
  minThreshold = 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')
characterizationTargetIds	The characterization target ids
analysisIds	The feature extraction analysis ID of interest (e.g., 201 is condition)
databaseIds	(Optional) A vector of database IDs to restrict to
includeNames	Whether to add database and cohort names (setting to FALSE will make extraction quicker)
minThreshold	(optional) The minimum average value for results to be returned

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
- databaseId the unique identifier 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: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

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

`getDatabaseDetails` *Extract the database used in the analyses*

Description

This function extracts the databases and their information.

Usage

```
getDatabaseDetails(
  connectionHandler,
  schema,
  databaseTable = "database_meta_data"
)
```

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)
`databaseTable`	The name of the table with the database details (default 'database_meta_data')

Details

Specify the connectionHandler, the schema and the database table name

Value

Returns a data.frame with the columns:

- databaseFullName the full name of the database
- databaseName the friendly name of the database
- cdmHolder the license holder of the database
- sourceDescription a description of the database
- sourceDocumentationReference a link to the database information document
- cdmEtlReference a link to the ETL document
- sourceReleaseDatethe release date for the source database
- (cdmReleaseDate the release date for the database mapped to the OMOP CDM)
- cdmVersion the OMOP CDM version of the database
- cdmVersionConceptId the CDM version concept ID
- vocabularyVersion the database's vocabulary version
- databaseId a unique identifier for the database
- maxObsPeriodEndDate the last observational period end date in the database

Examples

```
conDet <- getExampleConnectionDetails()

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

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

`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",
  characterizationTargetIds = 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')
`characterizationTargetIds`	The characterization target cohort ids of interest
`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
- databaseId the unique identifier 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: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

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

`getDechallengeRechallengeFails` *A function to extract the failed dechallenge-rechallenge cases*

Description

A function to extract the failed dechallenge-rechallenge cases

Usage

```
getDechallengeRechallengeFails(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  characterizationTargetId = NULL,
  outcomeId = NULL,
  databaseId = NULL,
  dechallengeStopInterval = NULL,
  dechallengeEvaluationWindow = 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
characterizationTargetId	The characterization target cohort id of interest
outcomeId	Am integer corresponding to the outcome cohort ID
databaseId	The unique identifier for the database of interest
dechallengeStopInterval	(optional) The maximum number of days between the outcome start and target end for an outcome to be flagged
dechallengeEvaluationWindow	(optional) The maximum number of days after the target restarts to see whether the outcome restarts

Details

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

Value

A data.frame each failed dechallenge rechallenge exposures and outcomes

See Also

Other Characterization: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterizatio getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

conCohort <- getDechallengeRechallengeFails(
  connectionHandler = connectionHandler,
  schema = 'main',
  characterizationTargetId = 1,
  outcomeId = 3,
```

```
  databaseId = 'eunomia'
)
```

`getEventDuration` *Extracts summary of event duration*

Description

This function extracts results summary stats of event duration for specified analysis ids and target cohorts

Usage

```
getEventDuration(
  connectionHandler,
  schema,
  analysisIds,
  tpTablePrefix = "tp_",
  databaseTable = "database_meta_data",
  databaseIds = NULL,
  databaseNames = NULL,
  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)
`analysisIds`	A vector of analysis ids to restrict to
`tpTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default 'database_meta_data')
`databaseIds`	(optional) A vector of database ids to restrict to
`databaseNames`	(optional) A vector of database Names to restrict to
`targetIds`	(optional) A vector of target cohort ids to restrict to

Details

Specify the connectionHandler, the schema, and the analysisIds

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- databaseId the unique identifier of the database
- analysisId the unique identifier of a treament patterns run
- targetCohortId the target cohort unique identifier
- targetCohortName the target cohort name
- eventName a string representing an events for a target. Uses '+' for combination of event cohorts
- rank the step number of event occurance
- eventCount the count of event occurance at rank
- durationAverage the average duration of event
- durationMax the maximum duration of event
- durationMin the minimum duration of event
- durationMedian the median duration of event
- p25Value the 25th percentile for duration of event
- p75Value the 75th percentile for duration of event
- standardDeviation the standard deviation for duration of event

See Also

Other TreatmentPatterns: `getAnalysisCohorts()`, `getTreatmentPathways()`

Examples

```
conDet <- getExampleConnectionDetails()

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

ed <- getEventDuration(
  connectionHandler = connectionHandler,
  schema = "main",
  analysisIds = c(1)
)
```

`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: `addTarColumn()`, `formatBinaryCovariateName()`, `formatTargetPopulation()`, `getOutcomeTable()`, `getTarText()`, `getTargetTable()`, `kableDark()`, `removeSpaces()`

Examples

```
conDet <- getExampleConnectionDetails()

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

`getFullPredictionPerformances` *Extract the model performances per evaluation*

Description

This function extracts the model performances per evaluation

Usage

```
getFullPredictionPerformances(
  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:

- timeStamp the date/time when the analysis occurred
- performanceId the unique identifier for the performance
- modelDesignId the unique identifier for the model design
- modelType the type of classifier
- algorithmName the model algorithm name
- covariateName a summary name for the candidate covariates
- 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
- validationTargetId the id for the validation target population
- validationTargetName the name for the validation target population if different from development
- developmentOutcomeId the unique cohort id for the development outcome
- developmentOutcomeName the name for the development outcome
- validationOutcomeId the id for the validation outcome
- validationOutcomeName the name for the validation outcome if different from development
- developmentDatabase the name for the database used to develop the model
- validationDatabase the name for the database used to validate the model if different from development
- validationTarId the validation time at risk id
- validationTimeAtRisk the time at risk used when evaluating the model if different from development
- developmentTarId the development time at risk id
- developmentTimeAtRisk the time at risk used when developing the model
- evaluation The type of evaluation: Test/Train/CV/Validation
- populationSize the test/validation population size used to develop the model
- outcomeCount the test/validation outcome count used to develop the model
- AUROC the AUROC value for the model
- 95 lower AUROC: the lower bound of the 95 percent CI AUROC value for the model
- 95 upper AUROC: the upper bound of the 95 percent CI AUROC value for the model
- AUPRC the discrimination AUPRC value for the model
- brier score: the brier value for the model
- brier score scaled: the scaled brier value for the model
- Average Precision: the average precision value for the model

- Eavg the calibration average error e-statistic value for the model
- E90 the calibration 90 percent upper bound e-statistic value for the model
- Emax the calibration max error e-statistic value for the model
- calibrationInLarge mean prediction: the calibration in the large mean predicted risk value for the model
- calibrationInLarge observed risk: the calibration in the large mean observed risk value for the model
- calibrationInLarge intercept: the calibration in the large value intercept for the model
- weak calibration intercept: the weak calibration intercept for the model
- weak calibration gradient: the weak calibration gradient for the model
- Hosmer Lemeshow calibration intercept: the Hosmer Lemeshow calibration intercept for the model
- Hosmer Lemeshow calibration gradient: the Hosmer Lemeshow calibration gradient for the model
- ... Additional metrics that are added to PLP

See Also

Other Prediction: `getPredictionAggregateTopPredictors()`, `getPredictionCohorts()`, `getPredictionCovariates()`, `getPredictionDiagnosticTable()`, `getPredictionDiagnostics()`, `getPredictionHyperParamSearch()`, `getPredictionIntercept()`, `getPredictionLift()`, `getPredictionModelDesigns()`, `getPredictionOutcomes()`, `getPredictionPerformanceTable()`, `getPredictionPerformances()`, `getPredictionTargets()`, `getPredictionTopPredictors()`

Examples

```
conDet <- getExampleConnectionDetails()

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

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

`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",
  characterizationTargetIds = NULL,
  targetIds = NULL,
  parentIds = 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')
characterizationTargetIds	The characterization identifier for the target
targetIds	A vector of integers corresponding to the target cohort IDs
parentIds	The parent cohort ids to restrict to
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
- databaseId the unique id of the database
- targetName the target cohort name
- targetId the target cohort unique identifier
- outcomeName the outcome name
- outcomeId the outcome unique identifier
- (tar the friendly time-at-risk string)
- 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: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterization(), getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

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

`getIncidenceTargetSettings` *A function to extarct the cohort incidence targets*

Description

A function to extarct the cohort incidence targets

Usage

```
getIncidenceTargetSettings(
  connectionHandler,
  schema,
  ciTablePrefix = "ci_",
  cgTablePrefix = "cg_",
  characterizationTargetIds = NULL,
  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)
`ciTablePrefix`	The prefix used for the cohort incidence results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables
`characterizationTargetIds`	optional vector of characterizationTargetIds to restrict to
`targetIds`	optional vector of targetIds to restrict to

Details

Specify the connectionHandler, the schema and the prefixes

Value

A data.frame with the characterization target cohort ids, names and inclusion criteria.

See Also

Other Characterization: `characterizationCompareBinary()`, `characterizationCompareContinuous()`, `getAggregateBinaryRiskFactors()`, `getAggregateContinuousRiskFactors()`, `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getBinaryTargetBaseline()`, `getCaseCounts()`, `getCharacterizationCaseSettings()`, `getCharacterizationDemographics()`, `getCharacterizationTargetSettings()`, `getContinuousCaseSeries()`, `getContinuousRiskFactors()`, `getContinuousTargetBaseline()`, `getDechallengeRechallenge()`,

getDechallengeRechallengeFails(), getIncidenceRates(), getNonCaseCounts(), getOutcomesUsedInCharacteriz
getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(),
getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(),
viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

targetCohorts <- getIncidenceTargetSettings(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getNonCaseCounts *Extract the non-cases counts result*

Description

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

Usage

```
getNonCaseCounts(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  characterizationTargetIds = NULL,
  characterizationCaseIds = NULL,
  startAnchor = NULL,
  endAnchor = NULL,
  riskWindowStart = NULL,
  riskWindowEnd = NULL,
  outcomeIds = NULL,
  outcomeWashouts = NULL,
  databaseIds = NULL,
  includeNames = TRUE
)
```

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')
characterizationTargetIds	The characterization target cohort ids of interest
characterizationCaseIds	The characterization case ids of interest
startAnchor	optional filter by startAchor
endAnchor	optional filter by endAchor
riskWindowStart	optional filter by riskWindowStart
riskWindowEnd	optional filter by riskWindowEnd
outcomeIds	A vector of integers corresponding to the outcome cohort IDs
outcomeWashouts	optional filter by washout
databaseIds	A vector of database IDs to restrict to
includeNames	whether to add the database names and cohort names

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
- databaseId the unique identifier of the database
- characterizationCaseId the unique identifier of target, outcome and TAR combination
- targetName the target cohort name
- targetId the target cohort unique identifier
- limitToFirstInNDays target index is limited to first in N days
- minPriorObservation the minimum required observation days prior to index for an entry
- nestingCohortId the cohort id a person must be in at index
- nestingName the cohort name a person must be in at index
- minAge min age to be included at index
- maxAge max age to be included at index

- studyStart index must be on or after this date to be included
- studyEnd index must be on or before this date to be included
- genderConceptIds the gender concept ids a subject must have to be included
- outcomeName the outcome name
- outcomeId the outcome unique identifier
- outcomeWashoutDays patients with the outcome occurring within this number of days prior to index are excluded (NA means no exclusion)
- rowCount the number of entries in the cohort
- personCount the number of people in the cohort
- withoutExcludedPersonCount the number of people in the target ignoring exclusions

See Also

Other Characterization: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getOutcomesUsedInCharacterization(), getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

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

---

getOutcomesUsedInCharacterization

A function to extract the outcomes found in characterization

Description

A function to extract the outcomes found in characterization

Usage

```
getOutcomesUsedInCharacterization(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  characterizationTargetId = NULL,
  targetId = NULL,
  printTimes = FALSE,
  useDcrc = TRUE,
  useTte = TRUE,
  useRf = TRUE,
  useCs = TRUE
)
```

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
characterizationTargetId	A vector of characterizationTargetIds to restrict to
targetId	An integer corresponding to the target cohort ID
printTimes	Print the time it takes to run each query
useDcrc	look for outcome in dechal-rechal results
useTte	look for outcome in time-to-event results
useRf	look for outcome in risk-factor results
useCs	look for outcome in case series results

Details

Specify the connectionHandler, the schema and the prefixes

Value

A data.frame with the characterization outcome cohort ids, names and which characterization analyses the cohorts are used in.

See Also

Other Characterization: `characterizationCompareBinary()`, `characterizationCompareContinuous()`, `getAggregateBinaryRiskFactors()`, `getAggregateContinuousRiskFactors()`, `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getBinaryTargetBaseline()`, `getCaseCounts()`, `getCharacterizationCaseSettings()`,

getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

cohorts <- getOutcomesUsedInCharacterization(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`getOutcomesUsedInIncidence` *A function to extract the outcomes found in incidence*

Description

A function to extract the outcomes found in incidence

Usage

```
getOutcomesUsedInIncidence(
  connectionHandler,
  schema,
  ciTablePrefix = "ci_",
  cgTablePrefix = "cg_",
  characterizationTargetId = NULL,
  targetId = NULL,
  parentId = 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

characterizationTargetId
: The characterization identifier for the target

targetId
: An integer corresponding to the target cohort ID

parentId
: the parent target cohort Id to extract outcomes for

Details

Specify the connectionHandler, the schema and the prefixes

Value

A data.frame with the incidence outcome cohort ids and names

See Also

Other Characterization: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterization(), getTargetCounts(), getTargetsUsedInCharacteri
getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

outcomes <- getOutcomesUsedInIncidence(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getOutcomeTable *Extract the outcome cohorts and where they are used in the analyses.*

Description

This function extracts the outcome cohorts, the number of subjects/entries and where the cohort was used.

Usage

```
getOutcomeTable(
  connectionHandler,
  schema,
  cgTablePrefix = "cg_",
  cTablePrefix = "c_",
  ciTablePrefix = "ci_",
  cmTablePrefix = "cm_",
  sccsTablePrefix = "sccs_",
  plpTablePrefix = "plp_",
  databaseTable = "database_meta_data",
  targetId = NULL,
  getIncidenceInclusion = TRUE,
  getCharacterizationInclusion = TRUE,
  getPredictionInclusion = TRUE,
  getCohortMethodInclusion = TRUE,
  getSccsInclusion = TRUE,
  printTimes = FALSE
)
```

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
cTablePrefix	The prefix used for the characterization results tables
ciTablePrefix	The prefix used for the cohort incidence results tables
cmTablePrefix	The prefix used for the cohort method results tables
sccsTablePrefix	The prefix used for the cohort generator results tables
plpTablePrefix	The prefix used for the patient level prediction 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
getIncidenceInclusion	Whether to check usage of the cohort in incidence
getCharacterizationInclusion	Whether to check usage of the cohort in characterization
getPredictionInclusion	Whether to check usage of the cohort in prediction
getCohortMethodInclusion	Whether to check usage of the cohort in cohort method
getSccsInclusion	Whether to check usage of the cohort in SCCS
printTimes	whether to print the time it takes to run each SQL query

Details

Specify the connectionHandler, the schema and the table prefixes

Value

Returns a data.frame with the columns:

- cohortId the number id for the target cohort
- cohortName the name of the cohort
- subsetParent the number id of the parent cohort
- subsetDefinitionId the number id of the subset
- numDatabase number of databases with the cohort
- databaseString all the names of the databases with the cohort
- databaseIdString all the ids of the databases with the cohort
- databaseStringCount all the names of the databases with the cohort plus their counts
- databaseCount all the names of the databases with the cohort and their sizes
- minSubjectCount number of subjects in databases with lowest count
- maxSubjectCount number of subjects in databases with highest count
- minEntryCount number of entries in databases with lowest count
- maxEntryCount number of entries in databases with highest count
- cohortIncidence whether the cohort was used in cohort incidence
- dechalRechal whether the cohort was used in dechallenge rechallenge
- riskFactors whether the cohort was used in risk factors
- caseSeries whether the cohort was used in case series analysis
- timeToEvent whether the cohort was used in time to event
- prediction whether the cohort was used in prediction
- cohortMethod whether the cohort was used in cohort method
- selfControlledCaseSeries whether the cohort was used in self controlled case series

See Also

Other helper: `addTarColumn()`, `formatBinaryCovariateName()`, `formatTargetPopulation()`, `getExampleConnectionDetails()`, `getTarText()`, `getTargetTable()`, `kableDark()`, `removeSpaces()`

Examples

```
conDet <- getExampleConnectionDetails()

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

outcomeTable <- getOutcomeTable(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`getPredictionAggregateTopPredictors` *Extract the top N predictors across a set of models*

Description

This function extracts the top N predictors across models by finding the sum of the absolute coefficient value across models.

Usage

```
getPredictionAggregateTopPredictors(
  connectionHandler,
  schema,
  plpTablePrefix = "plp_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  modelDesignIds = 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')
`modelDesignIds`	One or more model design IDs to restrict to

Details

Specify the connectionHandler, the resultDatabaseSettings and (optionally) any modelDesignIds to restrict 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

- covariateId the FeatureExtraction covariate identifier
- covariateName the name of the covariate
- conceptId the covariates corresponding concept or 0
- sumCovariateValue the total absolute feature importance or coefficient value
- numberOfTimesPredictive number of models that contained the covariate

See Also

Other Prediction: getFullPredictionPerformances(), getPredictionCohorts(), getPredictionCovariates(), getPredictionDiagnosticTable(), getPredictionDiagnostics(), getPredictionHyperParamSearch(), getPredictionIntercept(), getPredictionLift(), getPredictionModelDesigns(), getPredictionOutcomes(), getPredictionPerformanceTable(), getPredictionPerformances(), getPredictionTargets(), getPredictionTopPredictors()

Examples

```
conDet <- getExampleConnectionDetails()

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

topPreds <- getPredictionAggregateTopPredictors(
  connectionHandler = connectionHandler,
  schema = 'main',
  modelDesignIds = c(1,2,5)
)
```

---

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: getFullPredictionPerformances(), getPredictionAggregateTopPredictors(), getPredictionCovariates(), getPredictionDiagnosticTable(), getPredictionDiagnostics(), getPredictionHyperParamSearch(), getPredictionIntercept(), getPredictionLift(), getPredictionModelDesig getPredictionOutcomes(), getPredictionPerformanceTable(), getPredictionPerformances(), getPredictionTargets(), getPredictionTopPredictors()

Examples

```
conDet <- getExampleConnectionDetails()

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

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

`getPredictionCovariates` *Extract covariate summary details*

Description

This function extracts the covariate summary details

Usage

```
getPredictionCovariates(
  connectionHandler,
  schema,
  plpTablePrefix = "plp_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  performanceIds = NULL,
  modelDesignIds = 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')
`performanceIds`	(optional) restrict to the input performanceIds
`modelDesignIds`	(optional) restrict to the input modelDesignIds

Details

Specify the connectionHandler, the resultDatabaseSettings, the table of interest and (optionally) modelDesignIds/performanceIds to filter to

Value

Returns a data.frame with the specified table

See Also

Other Prediction: getFullPredictionPerformances(), getPredictionAggregateTopPredictors(), getPredictionCohorts(), getPredictionDiagnosticTable(), getPredictionDiagnostics(), getPredictionHyperParamSearch(), getPredictionIntercept(), getPredictionLift(), getPredictionModelDesig getPredictionOutcomes(), getPredictionPerformanceTable(), getPredictionPerformances(), getPredictionTargets(), getPredictionTopPredictors()

Examples

```
conDet <- getExampleConnectionDetails()

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

covs <- getPredictionCovariates(
  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",
  modelDesignIds = 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')
modelDesignIds	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: `getFullPredictionPerformances()`, `getPredictionAggregateTopPredictors()`, `getPredictionCohorts()`, `getPredictionCovariates()`, `getPredictionDiagnosticTable()`, `getPredictionHyperParamSearch()`, `getPredictionIntercept()`, `getPredictionLift()`, `getPredictionModelDesig`
`getPredictionOutcomes()`, `getPredictionPerformanceTable()`, `getPredictionPerformances()`, `getPredictionTargets()`, `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: `getFullPredictionPerformances()`, `getPredictionAggregateTopPredictors()`, `getPredictionCohorts()`, `getPredictionCovariates()`, `getPredictionDiagnostics()`, `getPredictionHyperParam`
`getPredictionIntercept()`, `getPredictionLift()`, `getPredictionModelDesigns()`, `getPredictionOutcomes()`, `getPredictionPerformanceTable()`, `getPredictionPerformances()`, `getPredictionTargets()`, `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: getFullPredictionPerformances(), getPredictionAggregateTopPredictors(), getPredictionCohorts(), getPredictionCovariates(), getPredictionDiagnosticTable(), getPredictionDiagnostics(), getPredictionIntercept(), getPredictionLift(), getPredictionModelDesigns(), getPredictionOutcomes(), getPredictionPerformanceTable(), getPredictionPerformances(), getPredictionTargets(), 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: getFullPredictionPerformances(), getPredictionAggregateTopPredictors(), getPredictionCohorts(), getPredictionCovariates(), getPredictionDiagnosticTable(), getPredictionDiagnostics(), getPredictionHyperParamSearch(), getPredictionLift(), getPredictionModelDesigns(), getPredictionOutcomes(), getPredictionPerformanceTable(), getPredictionPerformances(), getPredictionTargets(), getPredictionTopPredictors()

Examples

```
conDet <- getExampleConnectionDetails()

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

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

---

getPredictionLift *Extract model lift at given model sensitivity*

Description

This function extracts the model lift (PPV/outcomeRate)

Usage

```
getPredictionLift(
  connectionHandler,
  schema,
  plpTablePrefix = "plp_",
  modelDesignIds = NULL,
  performanceIds = NULL,
  sensitivity = 0.1
)
```

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
modelDesignIds	(optional) restrict to the input modelDesignIds
performanceIds	(optional) restrict to the input performanceIds
sensitivity	(default 0.1) the lift at the threshold with the sensitivity closest to this value is return

Details

Specify the connectionHandler, the resultDatabaseSettings and (optionally) modelDesignIds or performanceIds to filter to

Value

Returns a data.frame with the columns: modelDesignId, performanceId, evaluation, sensitivity, outcomeCount, positivePredictiveValue, outcomeRate and lift.

See Also

Other Prediction: getFullPredictionPerformances(), getPredictionAggregateTopPredictors(), getPredictionCohorts(), getPredictionCovariates(), getPredictionDiagnosticTable(), getPredictionDiagnostics(), getPredictionHyperParamSearch(), getPredictionIntercept(), getPredictionModelDesigns(), getPredictionOutcomes(), getPredictionPerformanceTable(), getPredictionPerformances(), getPredictionTargets(), getPredictionTopPredictors()

Examples

```
conDet <- getExampleConnectionDetails()

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

liftsAt0p15 <- getPredictionLift(
  connectionHandler = connectionHandler,
```

```
  schema = 'main',
  sensitivity = 0.15
)
```

`getPredictionModelDesigns` *Extract the model designs from the prediction results*

Description

This function extracts the model design settings

Usage

```
getPredictionModelDesigns(
  connectionHandler,
  schema,
  plpTablePrefix = "plp_",
  cgTablePrefix = "cg_",
  targetIds = NULL,
  outcomeIds = NULL,
  modelDesignIds = 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
`modelDesignIds`	(Optional) A set of model design ids to restrict to

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
- modelSettingsJson the model settings json

See Also

Other Prediction: `getFullPredictionPerformances()`, `getPredictionAggregateTopPredictors()`, `getPredictionCohorts()`, `getPredictionCovariates()`, `getPredictionDiagnosticTable()`, `getPredictionDiagnostics()`, `getPredictionHyperParamSearch()`, `getPredictionIntercept()`, `getPredictionLift()`, `getPredictionOutcomes()`, `getPredictionPerformanceTable()`, `getPredictionPerformance`
`getPredictionTargets()`, `getPredictionTopPredictors()`

Examples

```
conDet <- getExampleConnectionDetails()

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

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

`getPredictionOutcomes` *A function to extract the outcomes found in prediction*

Description

A function to extract the outcomes found in prediction

Usage

```
getPredictionOutcomes(
  connectionHandler,
  schema,
  plpTablePrefix = "plp_",
  cgTablePrefix = "cg_",
  targetId = 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
`targetId`	An integer corresponding to the target cohort ID

Details

Specify the connectionHandler, the schema and the prefixes

Value

A data.frame with the prediction outcome cohort ids and names.

See Also

Other Prediction: `getFullPredictionPerformances()`, `getPredictionAggregateTopPredictors()`, `getPredictionCohorts()`, `getPredictionCovariates()`, `getPredictionDiagnosticTable()`, `getPredictionDiagnostics()`, `getPredictionHyperParamSearch()`, `getPredictionIntercept()`, `getPredictionLift()`, `getPredictionModelDesigns()`, `getPredictionPerformanceTable()`, `getPredictionPerformances()`, `getPredictionTargets()`, `getPredictionTopPredictors()`

Examples

```
conDet <- getExampleConnectionDetails()

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

outcomes <- getPredictionOutcomes(
  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
- modelType the type of classifier
- algorithmName the model algorithm name
- 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: getFullPredictionPerformances(), getPredictionAggregateTopPredictors(), getPredictionCohorts(), getPredictionCovariates(), getPredictionDiagnosticTable(), getPredictionDiagnostics(), getPredictionHyperParamSearch(), getPredictionIntercept(), getPredictionLift(), getPredictionModelDesigns(), getPredictionOutcomes(), getPredictionPerformanceTabl getPredictionTargets(), 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_",
  databaseTable = "database_meta_data",
  table = "attrition",
  modelDesignIds = NULL,
  performanceIds = NULL,
  evaluations = 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

databaseTable	The name of the table with the database details (default 'database_meta_data')
table	The table to extract (covariate_summary, attrition, prediction_distribution, threshold_summary, calibration_summary, evaluation_statistics or demographic_summary)
modelDesignIds	(optional) restrict to the input modelDesignIds
performanceIds	(optional) restrict to the input performanceIds
evaluations	(optional) restrict to the type of evaluation (e.g., 'Test'/'Train'/'CV'/'Validation')

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: getFullPredictionPerformances(), getPredictionAggregateTopPredictors(), getPredictionCohorts(), getPredictionCovariates(), getPredictionDiagnosticTable(), getPredictionDiagnostics(), getPredictionHyperParamSearch(), getPredictionIntercept(), getPredictionLift(), getPredictionModelDesigns(), getPredictionOutcomes(), getPredictionPerformances(), getPredictionTargets(), getPredictionTopPredictors()

Examples

```
conDet <- getExampleConnectionDetails()

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

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

---

getPredictionTargets *A function to extarct the targets found in prediction*

Description

A function to extarct the targets found in prediction

Usage

```
getPredictionTargets(
  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 schema and the prefixes

Value

A data.frame with the prediction target cohort ids and names.

See Also

Other Prediction: getFullPredictionPerformances(), getPredictionAggregateTopPredictors(), getPredictionCohorts(), getPredictionCovariates(), getPredictionDiagnosticTable(), getPredictionDiagnostics(), getPredictionHyperParamSearch(), getPredictionIntercept(), getPredictionLift(), getPredictionModelDesigns(), getPredictionOutcomes(), getPredictionPerformanceTabl getPredictionPerformances(), getPredictionTopPredictors()

Examples

```
conDet <- getExampleConnectionDetails()

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

cohorts <- getPredictionTargets(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`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_",
  databaseTable = "database_meta_data",
  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
`databaseTable`	The database table name
`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: `getFullPredictionPerformances()`, `getPredictionAggregateTopPredictors()`, `getPredictionCohorts()`, `getPredictionCovariates()`, `getPredictionDiagnosticTable()`, `getPredictionDiagnostics()`, `getPredictionHyperParamSearch()`, `getPredictionIntercept()`, `getPredictionLift()`, `getPredictionModelDesigns()`, `getPredictionOutcomes()`, `getPredictionPerformanceTabl` `getPredictionPerformances()`, `getPredictionTargets()`

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
- databaseId the unique identifier for the database
- 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
- indicationName the indication name
- indicatonId the indication 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 (Depreciated to timeStabilityP) The p for whether the mean monthly ratio between observed and expected is no greater than 1.25.
- preExposureP (Depreciated) 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.
- timeStabilityP (New) The p for whether the mean monthly ratio between observed and expected exceeds the specified threshold.
- eventExposureLb (New) Lower bound of the 95% CI for the pre-expososure estimate.
- eventExposureUb (New) Upper bound of the 95% CI for the pre-expososure estimate.
- eventObservationLb (New) Lower bound of the 95% CI for the end of observation probe estimate.
- eventObservationUb (New) Upper bound of the 95% CI for the end of observation probe estimate.
- rareOutcomePrevalence (New) The proportion of people in the underlying population who have the outcome at least once.
- timeTrendDiagnostic (Depreciated) Pass / warning / fail / not evaluated classification of the time trend (unstalbe months) diagnostic.
- preExposureDiagnostic (Depreciated) Pass / warning / fail / not evaluated classification of the time trend (unstalbe months) diagnostic.
- timeStabilityDiagnostic (New) Pass / fail / not evaluated classification of the time stability diagnostic.
- eventExposureDiagnostic (New) Pass / fail / not evaluated classification of the event-exposure independence diagnostic.

- eventObservationDiagnostic (New) Pass / fail / not evaluated classification of the event-observation period dependence diagnostic.
- rareOutcomeDiagnostic (New) Pass / fail / not evaluated classification of the rare outcome 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: `.getCmVersion()`, `.getSccsVersion()`, `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getCmNegativeControlEstimates()`, `getCmOutcomes()`, `getCmPropensityModel()`, `getCmTable()`, `getCmTargets()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `getSccsModel()`, `getSccsNegativeControlEstimates()`, `getSccsOutcomes()`, `getSccsTable()`, `getSccsTargets()`, `getSccsTimeToEvent()`, `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
- databaseId the database id
- exposuresOutcomeSetId the exposure outcome set identifier
- 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
- indicationName the indication name
- indicatonId the indication cohort id
- covariateName whether main or secondary analysis
- covariateId the analysis id
- 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
- 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
- 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
- unblindForEvidenceSynthesis whether the results can be unblinded for the meta analysis.

See Also

Other Estimation: `.getCmVersion()`, `.getSccsVersion()`, `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getCmNegativeControlEstimates()`, `getCmOutcomes()`, `getCmPropensityModel()`, `getCmTable()`, `getCmTargets()`, `getSccsDiagnosticsData()`, `getSccsMetaEstimation()`, `getSccsModel()`, `getSccsNegativeControlEstimates()`, `getSccsOutcomes()`, `getSccsTable()`, `getSccsTargets()`, `getSccsTimeToEvent()`, `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,
  includeOneSidedP = TRUE
)
```

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
`includeOneSidedP`	This lets you extract from older results that do not have the one sided p by setting this to FALSE

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
- indicationName the indicationname
- indicationId the indication 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: `.getCmVersion()`, `.getSccsVersion()`, `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getCmNegativeControlEstimates()`, `getCmOutcomes()`, `getCmPropensityModel()`, `getCmTable()`, `getCmTargets()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsModel()`, `getSccsNegativeControlEstimates()`, `getSccsOutcomes()`, `getSccsTable()`, `getSccsTargets()`, `getSccsTimeToEvent()`, `plotCmEstimates()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

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

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

`getSccsModel` *Extract the SCCS model table*

Description

This function extracts the sccs model table.

Usage

```
getSccsModel(
  connectionHandler,
  schema,
  sccsTablePrefix = "sccs_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  exposureOutcomeSetIds = NULL,
  indicationIds = NULL,
  outcomeIds = NULL,
  databaseIds = NULL,
  analysisIds = NULL,
  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)
`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')

exposureOutcomeSetIds	the exposureOutcomeIds to restrict to
indicationIds	The indications that the target was nested to
outcomeIds	A vector of integers corresponding to the outcome cohort IDs
databaseIds	the database IDs to restrict to
analysisIds	the analysis IDs to restrict to
targetIds	A vector of integers corresponding to the target cohort IDs

Details

Specify the connectionHandler, the schema and optionally the target/outcome/analysis/database IDs

Value

Returns a data.frame with the SCCS model table

See Also

Other Estimation: `.getCmVersion()`, `.getSccsVersion()`, `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getCmNegativeControlEstimates()`, `getCmOutcomes()`, `getCmPropensityModel()`, `getCmTable()`, `getCmTargets()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `getSccsNegativeControlEstimates()`, `getSccsOutcomes()`, `getSccsTable()`, `getSccsTargets()`, `getSccsTimeToEvent()`, `plotCmEstimates()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

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

sccsModels <- getSccsModel(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getSccsNegativeControlEstimates

Extract the SCCS negative controls

Description

This function extracts the sccs negative controls.

Usage

```
getSccsNegativeControlEstimates(
  connectionHandler,
  schema,
  sccsTablePrefix = "sccs_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  databaseIds = NULL,
  exposuresOutcomeSetIds = NULL,
  indicationIds = NULL,
  outcomeIds = NULL,
  targetIds = NULL,
  analysisIds = NULL,
  covariateIds = NULL,
  covariateAnalysisIds = 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')
databaseIds	the database IDs to restrict to
exposuresOutcomeSetIds	the exposureOutcomeIds to restrict to
indicationIds	The indications that the target was nested to
outcomeIds	A vector of integers corresponding to the outcome cohort IDs
targetIds	A vector of integers corresponding to the target cohort IDs
analysisIds	the analysis IDs to restrict to
covariateIds	the covariate IDs to restrict to
covariateAnalysisIds	the covariate analysis IDs to restrict to

Details

Specify the connectionHandler, the schema and optionally the target/outcome/analysis/database IDs

Value

Returns a data.frame with the SCCS negative controls

See Also

Other Estimation: .getCmVersion(), .getSccsVersion(), getCMEstimation(), getCmDiagnosticsData(), getCmMetaEstimation(), getCmNegativeControlEstimates(), getCmOutcomes(), getCmPropensityModel(), getCmTable(), getCmTargets(), getSccsDiagnosticsData(), getSccsEstimation(), getSccsMetaEstimation(), getSccsModel(), getSccsOutcomes(), getSccsTable(), getSccsTargets(), getSccsTimeToEvent(), plotCmEstimates(), plotSccsEstimates()

Examples

```
conDet <- getExampleConnectionDetails()

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

sccsNcs <- getSccsNegativeControlEstimates(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getSccsOutcomes *A function to extract the outcomes found in self controlled case series*

Description

A function to extract the outcomes found in self controlled case series

Usage

```
getSccsOutcomes(
  connectionHandler,
  schema,
  sccsTablePrefix = "sccs_",
  cgTablePrefix = "cg_",
  targetId = 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
targetId	An integer corresponding to the target cohort ID

Details

Specify the connectionHandler, the schema and the prefixes

Value

A data.frame with the self controlled case series outcome ids and names.

See Also

Other Estimation: .getCmVersion(), .getSccsVersion(), getCMEstimation(), getCmDiagnosticsData(), getCmMetaEstimation(), getCmNegativeControlEstimates(), getCmOutcomes(), getCmPropensityModel(), getCmTable(), getCmTargets(), getSccsDiagnosticsData(), getSccsEstimation(), getSccsMetaEstimation(), getSccsModel(), getSccsNegativeControlEstimates(), getSccsTable(), getSccsTargets(), getSccsTimeToEvent(), plotCmEstimates(), plotSccsEstimates()

Examples

```
conDet <- getExampleConnectionDetails()

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

outcomes <- getSccsOutcomes(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getSccsTable *Extract the SCCS table specified*

Description

This function extracts the specific cohort method table.

Usage

```
getSccsTable(
  connectionHandler,
  schema,
  table = c("attrition", "time_trend", "event_dep_observation", "age_spanning",
    "calendar_time_spanning", "spline")[1],
  sccsTablePrefix = "sccs_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  indicationIds = NULL,
  outcomeIds = NULL,
  analysisIds = NULL,
  databaseIds = NULL,
```

```
  exposureOutcomeIds = NULL,
  covariateIds = 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)
table	The result table to extract
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')
indicationIds	The indications that the target was nested to
outcomeIds	A vector of integers corresponding to the outcome cohort IDs
analysisIds	the analysis IDs to restrict to
databaseIds	the database IDs to restrict to
exposureOutcomeIds	the exposureOutcomeIds to restrict to
covariateIds	the covariateIds to restrict to

Details

Specify the connectionHandler, the schema and optionally the target/outcome/analysis/database IDs

Value

Returns a data.frame with the cohort method requested table

See Also

Other Estimation: `.getCmVersion()`, `.getSccsVersion()`, `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getCmNegativeControlEstimates()`, `getCmOutcomes()`, `getCmPropensityModel()`, `getCmTable()`, `getCmTargets()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `getSccsModel()`, `getSccsNegativeControlEstimates()`, `getSccsOutcomes()`, `getSccsTargets()`, `getSccsTimeToEvent()`, `plotCmEstimates()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

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

sccsTable <- getSccsTable(
  connectionHandler = connectionHandler,
```

```
  schema = 'main',
  table = 'attrition'
)
```

`getSccsTargets` *A function to extract the targets found in self controlled case series*

Description

A function to extract the targets found in self controlled case series

Usage

```
getSccsTargets(
  connectionHandler,
  schema,
  sccsTablePrefix = "sccs_",
  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)
`sccsTablePrefix`	The prefix used for the cohort generator results tables
`cgTablePrefix`	The prefix used for the cohort generator results tables

Details

Specify the connectionHandler, the schema and the prefixes

Value

A data.frame with the self controlled case series target cohort ids and names.

See Also

Other Estimation: `.getCmVersion()`, `.getSccsVersion()`, `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `getCmNegativeControlEstimates()`, `getCmOutcomes()`, `getCmPropensityModel()`, `getCmTable()`, `getCmTargets()`, `getSccsDiagnosticsData()`, `getSccsEstimation()`, `getSccsMetaEstimation()`, `getSccsModel()`, `getSccsNegativeControlEstimates()`, `getSccsOutcomes()`, `getSccsTable()`, `getSccsTimeToEvent()`, `plotCmEstimates()`, `plotSccsEstimates()`

Examples

```
conDet <- getExampleConnectionDetails()

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

cohorts <- getSccsTargets(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`getSccsTimeToEvent` *Extract the SCCS time-to-event*

Description

This function extracts the SCCS time-to-event.

Usage

```
getSccsTimeToEvent(
  connectionHandler,
  schema,
  sccsTablePrefix = "sccs_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  databaseIds = NULL,
  exposuresOutcomeSetIds = NULL,
  indicationIds = NULL,
  outcomeIds = NULL,
  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)
`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')
`databaseIds`	the database IDs to restrict to

exposuresOutcomeSetIds	the exposureOutcomeIds to restrict to
indicationIds	The indications that the target was nested to
outcomeIds	A vector of integers corresponding to the outcome cohort IDs
targetIds	A vector of integers corresponding to the target cohort IDs
analysisIds	the analysis IDs to restrict to

Details

Specify the connectionHandler, the schema and optionally the target/outcome/analysis/database IDs

Value

Returns a data.frame with the SCCS time-to-event

See Also

Other Estimation: .getCmVersion(), .getSccsVersion(), getCMEstimation(), getCmDiagnosticsData(), getCmMetaEstimation(), getCmNegativeControlEstimates(), getCmOutcomes(), getCmPropensityModel(), getCmTable(), getCmTargets(), getSccsDiagnosticsData(), getSccsEstimation(), getSccsMetaEstimation(), getSccsModel(), getSccsNegativeControlEstimates(), getSccsOutcomes(), getSccsTable(), getSccsTargets(), plotCmEstimates(), plotSccsEstimates()

Examples

```
conDet <- getExampleConnectionDetails()

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

getSccsTimeToEvent <- getSccsNegativeControlEstimates(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getSubsetText *Function that converts a subsetDefinitionJson into text description*

Description

Function that converts a subsetDefinitionJson into text description

Usage

```
getSubsetText(subsetDefinitionJson, cohortDefinitions)
```

Arguments

`subsetDefinitionJson`
: The subset logic json

`cohortDefinitions`
: A data.frame with the columns cohortDefinitionId, cohortName and optionally friendlyName that will be used to know the friendly cohort name for any subsetting that nests in other cohorts

Details

The function takes a subsetDefinitionJson and converts it into friendly text describing the subset logic

Value

A text string describing the subsetting

See Also

Other Cohorts: `getCohortAttrition()`, `getCohortCounts()`, `getCohortDefinitions()`, `getCohortInclusionRules()`, `getCohortInclusionStats()`, `getCohortInclusionSummary()`, `getCohortMeta()`, `getCohortSubsetAttrition()`, `getCohortSubsetDefinitions()`, `processCohortDefinitionsForQuarto()`, `processCohorts()`, `restrictCohortDefinitionsForQuarto()`

`getTargetCounts` *Extract the target cohort counts result*

Description

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

Usage

```
getTargetCounts(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  characterizationTargetIds = NULL,
  databaseIds = NULL,
  includeNames = TRUE
)
```

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')
characterizationTargetIds	The characterization target cohort ids of interest
databaseIds	(optional) A vector of database IDs to restrict to
includeNames	whether to add the database names and cohort names

Details

Specify the connectionHandler, the schema and the characterization target IDs

Value

Returns a data.frame with the columns:

- settingId the setting id
- databaseId the unique identifier of the database
- characterizationTargetId the target cohort unique identifier
- N the number of people in the cohort

See Also

Other Characterization: `characterizationCompareBinary()`, `characterizationCompareContinuous()`, `getAggregateBinaryRiskFactors()`, `getAggregateContinuousRiskFactors()`, `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getBinaryTargetBaseline()`, `getCaseCounts()`, `getCharacterizationCaseSettings()`, `getCharacterizationDemographics()`, `getCharacterizationTargetSettings()`, `getContinuousCaseSeries()`, `getContinuousRiskFactors()`, `getContinuousTargetBaseline()`, `getDechallengeRechallenge()`, `getDechallengeRechallengeFails()`, `getIncidenceRates()`, `getIncidenceTargetSettings()`, `getNonCaseCounts()`, `getOutcomesUsedInCharacterization()`, `getOutcomesUsedInIncidence()`, `getTargetsUsedInCharacterization()`, `getTargetsUsedInIncidence()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`, `viewIncidenceRate()`

Examples

```
conDet <- getExampleConnectionDetails()

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

cc <- getTargetCounts(
connectionHandler = connectionHandler,
schema = 'main'
```

```
)
```

`getTargetsUsedInCharacterization` *A function to extarct the targets found in characterization*

Description

A function to extarct the targets found in characterization

Usage

```
getTargetsUsedInCharacterization(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  printTimes = FALSE,
  useTte = TRUE,
  useDcrc = TRUE,
  useRf = TRUE,
  useTb = TRUE,
  useCs = TRUE
)
```

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
`printTimes`	Print the time it takes to run each query
`useTte`	whether to determine what cohorts are used in time to event
`useDcrc`	whether to determine what cohorts are used in dechal-rechal
`useRf`	whether to determine what cohorts are used in risk factor
`useTb`	whether to determine what cohorts are used in target baseline
`useCs`	whether to determine what cohorts are used in case-series

Details

Specify the connectionHandler, the schema and the prefixes

Value

A data.frame with the characterization target cohort ids, names and which characterization analyses the cohorts are used in.

See Also

Other Characterization: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterization(), getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

cohorts <- getTargetsUsedInCharacterization(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getTargetsUsedInIncidence

A function to extract the targets found in incidence

Description

A function to extract the targets found in incidence

Usage

```
getTargetsUsedInIncidence(
  connectionHandler,
  schema,
  ciTablePrefix = "ci_",
  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)
ciTablePrefix	The prefix used for the cohort incidence results tables
cgTablePrefix	The prefix used for the cohort generator results tables

Details

Specify the connectionHandler, the schema and the prefixes

Value

A data.frame with the incidence target cohort ids and names

See Also

Other Characterization: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterization(), getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

cohorts <- getTargetsUsedInIncidence(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

---

getTargetTable *Extract the target cohorts and where they are used in the analyses.*

Description

This function extracts the target cohorts, the number of subjects/entries and where the cohort was used.

Usage

```
getTargetTable(
  connectionHandler,
  schema,
  cgTablePrefix = "cg_",
  cTablePrefix = "c_",
  ciTablePrefix = "ci_",
  cmTablePrefix = "cm_",
  sccsTablePrefix = "sccs_",
  plpTablePrefix = "plp_",
  databaseTable = "database_meta_data",
  getIncidenceInclusion = TRUE,
  getCharacterizationInclusion = TRUE,
  getPredictionInclusion = TRUE,
  getCohortMethodInclusion = TRUE,
  getSccsInclusion = TRUE,
  printTimes = FALSE
)
```

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
cTablePrefix	The prefix used for the characterization results tables
ciTablePrefix	The prefix used for the cohort incidence results tables
cmTablePrefix	The prefix used for the cohort method results tables
sccsTablePrefix	The prefix used for the cohort generator results tables
plpTablePrefix	The prefix used for the patient level prediction results tables
databaseTable	The name of the table with the database details (default 'database_meta_data')
getIncidenceInclusion	Whether to check useage of the cohort in incidence
getCharacterizationInclusion	Whether to check useage of the cohort in characterization
getPredictionInclusion	Whether to check useage of the cohort in prediction
getCohortMethodInclusion	Whether to check useage of the cohort in cohort method
getSccsInclusion	Whether to check useage of the cohort in SCCS
printTimes	Whether to print how long each query took

Details

Specify the connectionHandler, the schema and the table prefixes

Value

Returns a data.frame with the columns:

- cohortId the number id for the target cohort
- cohortName the name of the cohort
- subsetParent the number id of the parent cohort
- subsetDefinitionId the number id of the subset
- subsetDefinitionJson the json of the subset
- subsetCohortIds the ids of any cohorts that are restricted to by the subset logic
- numDatabase number of databases with the cohort
- databaseString all the names of the databases with the cohort
- databaseCount all the names of the databases with the cohort and their sizes
- minSubjectCount number of subjects in databases with lowest count
- maxSubjectCount number of subjects in databases with highest count
- minEntryCount number of entries in databases with lowest count
- maxEntryCount number of entries in databases with highest count
- cohortIncidence whether the cohort was used in cohort incidence
- databaseComparator whether the cohort was used in database comparator
- cohortComparator whether the cohort was used in cohort comparator
- dechalRechal whether the cohort was used in dechallenge rechallenge
- riskFactors whether the cohort was used in risk factors
- caseSeries whether the cohort was used in case series analysis
- timeToEvent whether the cohort was used in time to event
- prediction whether the cohort was used in prediction
- cohortMethod whether the cohort was used in cohort method
- selfControlledCaseSeries whether the cohort was used in self controlled case series

See Also

Other helper: `addTarColumn()`, `formatBinaryCovariateName()`, `formatTargetPopulation()`, `getExampleConnectionDetails()`, `getOutcomeTable()`, `getTarText()`, `kableDark()`, `removeSpaces()`

Examples

```
conDet <- getExampleConnectionDetails()

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

targetTable <- getTargetTable(
  connectionHandler = connectionHandler,
  schema = 'main'
)
```

`getTarText` *output a string summarizing the time at risk*

Description

This returns a character that describes the time at risk

Usage

```
getTarText(tar)
```

Arguments

`tar`	a list containing the four tar setting

Details

Input the tar settings

Value

An character string describing the tar

See Also

Other helper: `addTarColumn()`, `formatBinaryCovariateName()`, `formatTargetPopulation()`, `getExampleConnectionDetails()`, `getOutcomeTable()`, `getTargetTable()`, `kableDark()`, `removeSpaces()`

Examples

```
targetText <- getTarText(
list(
startAnchor = 'cohort_start',
riskWindowStart = 1,
endAnchor = 'cohort_end',
riskWindowEnd = 0
)
)
```

`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",
  characterizationTargetIds = 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')
`characterizationTargetIds`	The characterization target cohort ids of interest
`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
- databaseId the unique identifier of the database
- targetName the target cohort name
- targetId the target cohort unique identifier

- limitToFirstInNDays the target cohort was restrict to first in N days
- minPriorObservation the target cohort was restrict to require minPriorObservation before index
- nestingCohortId the nesting cohort id that a target cohort subject must also be in at index
- nestingName the nesting cohort name that a target cohort subject must also be in at index
- minAge the min age of the target cohort
- maxAge the max age of the target cohort
- studyStart the earliest date of the target cohort
- studyEnd the latlest date of the target cohort
- genderConceptIds the gender concept ids restricted to
- 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: `characterizationCompareBinary()`, `characterizationCompareContinuous()`, `getAggregateBinaryRiskFactors()`, `getAggregateContinuousRiskFactors()`, `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getBinaryTargetBaseline()`, `getCaseCounts()`, `getCharacterizationCaseSettings()`, `getCharacterizationDemographics()`, `getCharacterizationTargetSettings()`, `getContinuousCaseSeries()`, `getContinuousRiskFactors()`, `getContinuousTargetBaseline()`, `getDechallengeRechallenge()`, `getDechallengeRechallengeFails()`, `getIncidenceRates()`, `getIncidenceTargetSettings()`, `getNonCaseCounts()`, `getOutcomesUsedInCharacterization()`, `getOutcomesUsedInIncidence()`, `getTargetCounts()`, `getTargetsUsedInCharacterization()`, `getTargetsUsedInIncidence()`, `plotAgeDistributions()`, `plotSexDistributions()`, `viewIncidenceRate()`

Examples

```
conDet <- getExampleConnectionDetails()

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

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

`getTreatmentPathways` *Extracts treatment pathways*

Description

This function extracts results pathways for specified analysis ids and target cohorts

Usage

```
getTreatmentPathways(
  connectionHandler,
  schema,
  tpTablePrefix = "tp_",
  databaseTable = "database_meta_data",
  age = "all",
  sex = "all",
  indexYear = "all",
  analysisIds = NULL,
  databaseIds = NULL,
  databaseNames = NULL,
  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)
`tpTablePrefix`	The prefix used for the cohort generator results tables
`databaseTable`	The name of the table with the database details (default 'database_meta_data')
`age`	(optional) a string representing an age bucket to restrict (e.g., "0-17", "18-34", "65+")
`sex`	(optional) A string "male" or "female" to restrict
`indexYear`	(optional) A string with a four-digit year to restrict
`analysisIds`	(optional) A vector of analysis ids to restrict to
`databaseIds`	(optional) A vector of database ids to restrict to
`databaseNames`	(optional) A vector of database Names to restrict to
`targetIds`	(optional) A vector of target cohort ids to restrict to

Details

Specify the connectionHandler and the schema

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- databaseId the unique identifier of the database
- analysisId the unique identifier of a treament patterns run
- targetCohortName the target cohort name
- targetCohortId the target cohort unique identifier
- pathway a string representing the progression of events for a target. Use '-' to separate sequential steps and '+' for combination of events at that step
- freq the count of pathway occurance
- age the stratifyed pathways for age
- indexYear the stratifyed pathways for index year
- sex the stratifyed pathways for sex

See Also

Other TreatmentPatterns: `getAnalysisCohorts()`, `getEventDuration()`

Examples

```
conDet <- getExampleConnectionDetails()

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

tp <- getTreatmentPathways(
  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: addTarColumn(), formatBinaryCovariateName(), formatTargetPopulation(), getExampleConnectionDetails(), getOutcomeTable(), getTarText(), getTargetTable(), 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:

- Jenna Reps <jreps@its.jnj.com>
- 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.

Usage

```
plotAgeDistributions(ageData)
```

Arguments

`ageData`	The age data extracted using 'getCharacterizationDemographics(type = 'age')'

Details

Input the data returned from 'getCharacterizationDemographics(type = 'age')'

Value

Returns a ggplot with the distributions

See Also

Other Characterization: `characterizationCompareBinary()`, `characterizationCompareContinuous()`, `getAggregateBinaryRiskFactors()`, `getAggregateContinuousRiskFactors()`, `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getBinaryTargetBaseline()`, `getCaseCounts()`, `getCharacterizationCaseSettings()`, `getCharacterizationDemographics()`, `getCharacterizationTargetSettings()`, `getContinuousCaseSeries()`, `getContinuousRiskFactors()`, `getContinuousTargetBaseline()`, `getDechallengeRechallenge()`, `getDechallengeRechallengeFails()`, `getIncidenceRates()`, `getIncidenceTargetSettings()`, `getNonCaseCounts()`, `getOutcomesUsedInCharacterization()`, `getOutcomesUsedInIncidence()`, `getTargetCounts()`, `getTargetsUsedInCharacterization()`, `getTargetsUsedInIncidence()`, `getTimeToEvent()`, `plotSexDistributions()`, `viewIncidenceRate()`

Examples

```
conDet <- getExampleConnectionDetails()

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

ageData <- getCharacterizationDemographics(
connectionHandler = connectionHandler,
schema = 'main',
characterizationTargetId = 10,
type = 'age'
)

plotAgeDistributions(ageData = ageData)
```

---

plotCmEstimates *Plots the cohort method results for one analysis*

Description

Creates nice cohort method plots

Usage

```
plotCmEstimates(
  cmData,
  cmDiagnostics = NULL,
  cmMeta = NULL,
  cohortNames = NULL,
  includeCounts = TRUE,
  selectedAnalysisId = NULL
)
```

Arguments

cmData	The cohort method data
cmDiagnostics	(optional) The cohort method diagnostic data
cmMeta	(optional) The cohort method evidence synthesis data
cohortNames	A data.frame with columns cohortId and cohortName
includeCounts	Whether to include the target/comp size and event counts
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: .getCmVersion(), .getSccsVersion(), getCMEstimation(), getCmDiagnosticsData(), getCmMetaEstimation(), getCmNegativeControlEstimates(), getCmOutcomes(), getCmPropensityModel(), getCmTable(), getCmTargets(), getSccsDiagnosticsData(), getSccsEstimation(), getSccsMetaEstimation(), getSccsModel(), getSccsNegativeControlEstimates(), getSccsOutcomes(), getSccsTable(), getSccsTargets(), getSccsTimeToEvent(), plotSccsEstimates()

Examples

```
conDet <- getExampleConnectionDetails()

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

cmEst <- getCMEstimation(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetIds = 1002,
  outcomeIds = 3
)
plotCmEstimates(
  cmData = cmEst,
  cmMeta = NULL,
  selectedAnalysisId = 1
)
```

---

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

Description

Creates nice self controlled case series plots

Usage

```
plotSccsEstimates(
  sccsData,
  sccsDiagnostics = NULL,
  sccsMeta = NULL,
  includeCounts = TRUE,
  selectedAnalysisId
)
```

Arguments

sccsData	The self controlled case series data
sccsDiagnostics	(optional) The self controlled case series diagnostic data
sccsMeta	(optional) The self controlled case series evidence synthesis data
includeCounts	Whether to include count on the plot
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: .getCmVersion(), .getSccsVersion(), getCMEstimation(), getCmDiagnosticsData(), getCmMetaEstimation(), getCmNegativeControlEstimates(), getCmOutcomes(), getCmPropensityModel(), getCmTable(), getCmTargets(), getSccsDiagnosticsData(), getSccsEstimation(), getSccsMetaEstimation(), getSccsModel(), getSccsNegativeControlEstimates(), getSccsOutcomes(), getSccsTable(), getSccsTargets(), getSccsTimeToEvent(), plotCmEstimates()

Examples

```
conDet <- getExampleConnectionDetails()

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

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

plotSexDistributions *Plots the sex distributions using the sex features*

Description

Creates bar charts for the target.

Usage

```
plotSexDistributions(sexData)
```

Arguments

sexData	The sex data extracted using 'getCharacterizationDemographics(type = 'sex')'

Details

Input the data returned from 'getCharacterizationDemographics(type = 'sex')'

Value

Returns a ggplot with the distributions

See Also

Other Characterization: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterization(), getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), viewIncidenceRate()

Examples

```
conDet <- getExampleConnectionDetails()

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

sexData <- getCharacterizationDemographics(
  connectionHandler = connectionHandler,
  schema = 'main',
  characterizationTargetId = 10,
  outcomeId = 3,
  type = 'sex'
)
plotSexDistributions(sexData = sexData)
```

`processCohortDefinitionsForQuarto` *Function processes the cohortDefinitions object ready for use in the main quarto report.*

Description

Function processes the cohortDefinitions object ready for use in the main quarto report.

Usage

```
processCohortDefinitionsForQuarto(
  cohortDefinitions,
  friendlyCohortIds,
  friendlyCohortNames,
  restrictTargetToIndications,
  indicationIds
)
```

Arguments

`cohortDefinitions`	The output of 'getCohortDefinitions()'
`friendlyCohortIds`	a vector of parent cohort ids that you want to rename
`friendlyCohortNames`	a vector of new names for the friendlyCohortIds
`restrictTargetToIndications`	whether to restrict the results to cohorts nested in certain indications
`indicationIds`	The indication ids of interest for restrictTargetToIndications

Details

Function processes the cohortDefinitions object by adding friendly names for specified parent cohorts, determines which cohorts are nested in the specified indication cohort ids and ...

Value

A cohortDefinitions object with extra columns: friendlyName,

See Also

Other Cohorts: `getCohortAttrition()`, `getCohortCounts()`, `getCohortDefinitions()`, `getCohortInclusionRules()` `getCohortInclusionStats()`, `getCohortInclusionSummary()`, `getCohortMeta()`, `getCohortSubsetAttrition()`, `getCohortSubsetDefinitions()`, `getSubsetText()`, `processCohorts()`, `restrictCohortDefinitionsForQuarto()`

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: getCohortAttrition(), getCohortCounts(), getCohortDefinitions(), getCohortInclusionRules() getCohortInclusionStats(), getCohortInclusionSummary(), getCohortMeta(), getCohortSubsetAttrition(), getCohortSubsetDefinitions(), getSubsetText(), processCohortDefinitionsForQuarto(), restrictCohortDefinitionsForQuarto()

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: `addTarColumn()`, `formatBinaryCovariateName()`, `formatTargetPopulation()`, `getExampleConnectionDetails()`, `getOutcomeTable()`, `getTarText()`, `getTargetTable()`, `kableDark()`

Examples

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

`restrictCohortDefinitionsForQuarto` *Function to figure out the target, comparator, outcome and indication ids of interest for the quarto report based on the user inputs*

Description

Function to figure out the target, comparator, outcome and indication ids of interest for the quarto report based on the user inputs

Usage

```
restrictCohortDefinitionsForQuarto(
  connectionHandler,
  schema,
  cohortDefinitions,
  targetId,
  outcomeIds,
  comparatorIds,
  restrictTargetToIndications,
  indicationIds,
  includeCohortMethod
)
```

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)
cohortDefinitions	The output of 'processCohortDefinitionsForQuarto()'
targetId	a parent target id
outcomeIds	a vector of outcome ids
comparatorIds	NULL or a vector of comparator parent ids
restrictTargetToIndications	whether to restrict the target ids to cohorts nested in indicationIds
indicationIds	The indication ids of interest for restrictTargetToIndications
includeCohortMethod	If TRUE, when comparatorIds is NULL all comparators included in CohortMethod are included in the report

Details

This function finds the targets, comparators, indications and outcomes of interest based on the user inputs for quarto report generation.

Value

A list of targetIdsOfInterest that is a vector of cohortIds that are targets of interest to include in the report, comparatorIds a vector of cohortIds that are comparators, comparatorIdsOfInterest a vector of cohortIds that are comparators of interest, outcomeIdsOfInterest a vector of cohortIds that are outcomes of interest and indicationIdsOfInterest a vector of cohortIds that are indications of interest.

See Also

Other Cohorts: getCohortAttrition(), getCohortCounts(), getCohortDefinitions(), getCohortInclusionRules() getCohortInclusionStats(), getCohortInclusionSummary(), getCohortMeta(), getCohortSubsetAttrition(), getCohortSubsetDefinitions(), getSubsetText(), processCohortDefinitionsForQuarto(), processCohorts()

`viewIncidenceRate` *View the Incidence Rates*

Description

Creates a table with the incidence rates and optionally demographics

Usage

```
viewIncidenceRate(
  incidenceData,
  ageData = NULL,
  genderData = NULL,
  stratification = "overall",
  maxAgeSampleSize = 5000
)
```

Arguments

`incidenceData`	The data extracted using 'getIncidenceRates'
`ageData`	The data extracted using 'getBinaryTargetBaseline' with analysisIds = 3
`genderData`	The data extracted using 'getBinaryTargetBaseline' with analysisIds = 1
`stratification`	Pick either overall/age/sex/year to specify whether to view the overall rates or stratified by age/sex/year
`maxAgeSampleSize`	When creating the age distributions this is the max age vector length to create

Details

Input the incidence rate data (and optionally demographic data)

Value

Returns a gt table that displays the incidence rates

See Also

Other Characterization: characterizationCompareBinary(), characterizationCompareContinuous(), getAggregateBinaryRiskFactors(), getAggregateContinuousRiskFactors(), getBinaryCaseSeries(), getBinaryRiskFactors(), getBinaryTargetBaseline(), getCaseCounts(), getCharacterizationCaseSettings(), getCharacterizationDemographics(), getCharacterizationTargetSettings(), getContinuousCaseSeries(), getContinuousRiskFactors(), getContinuousTargetBaseline(), getDechallengeRechallenge(), getDechallengeRechallengeFails(), getIncidenceRates(), getIncidenceTargetSettings(), getNonCaseCounts(), getOutcomesUsedInCharacterization(), getOutcomesUsedInIncidence(), getTargetCounts(), getTargetsUsedInCharacterization(), getTargetsUsedInIncidence(), getTimeToEvent(), plotAgeDistributions(), plotSexDistributions()

Examples

```
conDet <- getExampleConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)
schema <- 'main'

incidenceData <- getIncidenceRates(
  connectionHandler = connectionHandler ,
  schema = schema
  )

  # incidence data does not have rate values to imputing them
  incidenceData$incidenceRateP100py <- 1 +
    sample(c(-1,1),replace = TRUE)*runif(nrow(incidenceData))
  incidenceData$incidenceProportionP100p <- 0.5 +
    sample(c(-1,1),replace = TRUE)*runif(nrow(incidenceData))

 ageData <- getBinaryTargetBaseline(
  connectionHandler = connectionHandler,
  schema = schema,
  analysisIds = 3
 )

 genderData <- getBinaryTargetBaseline(
  connectionHandler = connectionHandler,
  schema = schema,
  analysisIds = 1
 )

viewIncidenceRate(
  incidenceData = incidenceData,
  ageData = ageData,
  genderData = genderData
  )
```

Index

∗ **Characterization**
characterizationCompareBinary, 7
characterizationCompareContinuous, 8
getAggregateBinaryRiskFactors, 18
getAggregateContinuousRiskFactors, 20
getBinaryCaseSeries, 23
getBinaryRiskFactors, 25
getBinaryTargetBaseline, 26
getCaseCounts, 28
getCharacterizationCaseSettings, 31
getCharacterizationDemographics, 32
getCharacterizationTargetSettings, 34
getContinuousCaseSeries, 59
getContinuousRiskFactors, 61
getContinuousTargetBaseline, 62
getDechallengeRechallenge, 66
getDechallengeRechallengeFails, 68
getIncidenceRates, 75
getIncidenceTargetSettings, 78
getNonCaseCounts, 79
getOutcomesUsedInCharacterization, 81
getOutcomesUsedInIncidence, 83
getTargetCounts, 124
getTargetsUsedInCharacterization, 126
getTargetsUsedInIncidence, 127
getTimeToEvent, 132
plotAgeDistributions, 137
plotSexDistributions, 141
viewIncidenceRate, 146
∗ **Cohorts**
getCohortAttrition, 49
getCohortCounts, 50
getCohortDefinitions, 51
getCohortInclusionRules, 52
getCohortInclusionStats, 53
getCohortInclusionSummary, 54
getCohortMeta, 55
getCohortSubsetAttrition, 57
getCohortSubsetDefinitions, 58
getSubsetText, 123
processCohortDefinitionsForQuarto, 142
processCohorts, 143
restrictCohortDefinitionsForQuarto, 144
∗ **Database**
getDatabaseDetails, 65
∗ **Estimation**
.getCmVersion, 4
.getSccsVersion, 5
getCmDiagnosticsData, 36
getCMEstimation, 38
getCmMetaEstimation, 40
getCmNegativeControlEstimates, 42
getCmOutcomes, 44
getCmPropensityModel, 45
getCmTable, 46
getCmTargets, 48
getSccsDiagnosticsData, 108
getSccsEstimation, 110
getSccsMetaEstimation, 113
getSccsModel, 115
getSccsNegativeControlEstimates, 116
getSccsOutcomes, 118
getSccsTable, 119
getSccsTargets, 121
getSccsTimeToEvent, 122
plotCmEstimates, 138
plotSccsEstimates, 139
∗ **Indexes**

createCharacterizationIndexes, 10
createCohortIndexes, 10
createIncidenceIndexes, 11
createSccsIndexes, 13
∗ **Prediction**
getFullPredictionPerformances, 73
getPredictionAggregateTopPredictors, 87
getPredictionCohorts, 88
getPredictionCovariates, 90
getPredictionDiagnostics, 91
getPredictionDiagnosticTable, 93
getPredictionHyperParamSearch, 94
getPredictionIntercept, 95
getPredictionLift, 96
getPredictionModelDesigns, 98
getPredictionOutcomes, 100
getPredictionPerformances, 101
getPredictionPerformanceTable, 103
getPredictionTargets, 104
getPredictionTopPredictors, 106
∗ **Reporting**
createPredictionReport, 12
generateFullReport, 15
generateSummaryPredictionReport, 17
∗ **TreatmentPatterns**
getAnalysisCohorts, 22
getEventDuration, 70
getTreatmentPathways, 134
∗ **helper**
addTarColumn, 6
formatBinaryCovariateName, 13
formatTargetPopulation, 14
getExampleConnectionDetails, 72
getOutcomeTable, 84
getTargetTable, 128
getTarText, 131
kableDark, 135
removeSpaces, 144
.getCmVersion, 4
.getCmVersion(), *5*, *37*, *39*, *42–44*, *46–48*, *110*, *112*, *114*, *116*, *118–121*, *123*, *139, 140*
.getSccsVersion, 5
.getSccsVersion(), *4*, *37*, *39*, *42–44*, *46–48*, *110*, *112*, *114*, *116*, *118–121*, *123*, *139, 140*

addTarColumn, 6
addTarColumn(), *14*, *72*, *86*, *130, 131*, *136*, *144*

characterizationCompareBinary, 7
characterizationCompareBinary(), *9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77, 78*, *81, 82*, *84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
characterizationCompareContinuous, 8
characterizationCompareContinuous(), *8*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77, 78*, *81, 82*, *84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
createCharacterizationIndexes, 10
createCharacterizationIndexes(), *11*, *13*
createCohortIndexes, 10
createCohortIndexes(), *10, 11*, *13*
createIncidenceIndexes, 11
createIncidenceIndexes(), *10, 11*, *13*
createPredictionReport, 12
createPredictionReport(), *17, 18*
createSccsIndexes, 13
createSccsIndexes(), *10, 11*

formatBinaryCovariateName, 13
formatBinaryCovariateName(), *6*, *14*, *72*, *86*, *130, 131*, *136*, *144*
formatTargetPopulation, 14
formatTargetPopulation(), *6*, *14*, *72*, *86*, *130, 131*, *136*, *144*

generateFullReport, 15
generateFullReport(), *13*, *18*
generateSummaryPredictionReport, 17
generateSummaryPredictionReport(), *13*, *17*
getAggregateBinaryRiskFactors, 18
getAggregateBinaryRiskFactors(), *8, 9*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77, 78*, *81, 82*, *84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getAggregateContinuousRiskFactors, 20
getAggregateContinuousRiskFactors(), *8*, *9*, *19*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77, 78*, *81, 82*, *84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getAnalysisCohorts, 22
getAnalysisCohorts(), *71*, *135*

getBinaryCaseSeries, 23
getBinaryCaseSeries(), *8, 9*, *19*, *21*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77, 78*, *81, 82*, *84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getBinaryRiskFactors, 25
getBinaryRiskFactors(), *8, 9*, *19*, *21*, *24*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77, 78*, *81, 82*, *84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getBinaryTargetBaseline, 26
getBinaryTargetBaseline(), *8, 9*, *19*, *21*, *24*, *26*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77, 78*, *81, 82*, *84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getCaseCounts, 28
getCaseCounts(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77, 78*, *81, 82*, *84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getCharacterizationCaseSettings, 31
getCharacterizationCaseSettings(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77, 78*, *81, 82*, *84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getCharacterizationDemographics, 32
getCharacterizationDemographics(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *35*, *60*, *62*, *64*, *68, 69*, *77, 78*, *81*, *83, 84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getCharacterizationTargetSettings, 34
getCharacterizationTargetSettings(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34*, *60*, *62*, *64*, *68, 69*, *77, 78*, *81*, *83, 84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getCmDiagnosticsData, 36
getCmDiagnosticsData(), *4, 5*, *39*, *42–44*, *46–48*, *110*, *112*, *114*, *116*, *118–121*, *123*, *139, 140*
getCMEstimation, 38
getCMEstimation(), *4, 5*, *37*, *42–44*, *46–48*, *110*, *112*, *114*, *116*, *118–121*, *123*, *139, 140*
getCmMetaEstimation, 40
getCmMetaEstimation(), *4, 5*, *37*, *39*, *43, 44*, *46–48*, *110*, *112*, *114*, *116*, *118–121*, *123*, *139, 140*
getCmNegativeControlEstimates, 42
getCmNegativeControlEstimates(), *4, 5*, *37*, *39*, *42*, *44*, *46–48*, *110*, *112*, *114*, *116*, *118–121*, *123*, *139, 140*
getCmOutcomes, 44
getCmOutcomes(), *4, 5*, *37*, *39*, *42, 43*, *46–48*, *110*, *112*, *114*, *116*, *118–121*, *123*, *139, 140*
getCmPropensityModel, 45
getCmPropensityModel(), *4, 5*, *37*, *39*, *42–44*, *47, 48*, *110*, *112*, *114*, *116*, *118–121*, *123*, *139, 140*
getCmTable, 46
getCmTable(), *4, 5*, *37*, *39*, *42–44*, *46*, *48*, *110*, *112*, *114*, *116*, *118–121*, *123*, *139, 140*
getCmTargets, 48
getCmTargets(), *4, 5*, *37*, *39*, *42–44*, *46, 47*, *110*, *112*, *114*, *116*, *118–121*, *123*, *139, 140*
getCohortAttrition, 49
getCohortAttrition(), *50*, *52–58*, *124*, *142, 143*, *146*
getCohortCounts, 50
getCohortCounts(), *49*, *52–58*, *124*, *142, 143*, *146*
getCohortDefinitions, 51
getCohortDefinitions(), *49, 50*, *53–58*, *124*, *142, 143*, *146*
getCohortInclusionRules, 52
getCohortInclusionRules(), *49, 50*, *52*, *54–58*, *124*, *142, 143*, *146*
getCohortInclusionStats, 53
getCohortInclusionStats(), *49, 50*, *52, 53*, *55–58*, *124*, *142, 143*, *146*
getCohortInclusionSummary, 54
getCohortInclusionSummary(), *49, 50*, *52–54*, *56–58*, *124*, *142, 143*, *146*
getCohortMeta, 55
getCohortMeta(), *49, 50*, *52–55*, *57, 58*, *124*, *142, 143*, *146*
getCohortSubsetAttrition, 57
getCohortSubsetAttrition(), *49, 50*, *52–56*, *58*, *124*, *142, 143*, *146*
getCohortSubsetDefinitions, 58
getCohortSubsetDefinitions(), *49, 50*, *52–57*, *124*, *142, 143*, *146*
getContinuousCaseSeries, 59
getContinuousCaseSeries(), *8, 9*, *19*, *21*,

24, *26*, *28*, *30*, *32*, *34, 35*, *62*, *64*, *68, 69*, *77, 78*, *81*, *83, 84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getContinuousRiskFactors, 61
getContinuousRiskFactors(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *64*, *68, 69*, *77, 78*, *81*, *83, 84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getContinuousTargetBaseline, 62
getContinuousTargetBaseline(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *68, 69*, *77, 78*, *81*, *83, 84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getDatabaseDetails, 65
getDechallengeRechallenge, 66
getDechallengeRechallenge(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *69*, *77, 78*, *81*, *83, 84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getDechallengeRechallengeFails, 68
getDechallengeRechallengeFails(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68*, *77*, *79*, *81*, *83, 84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getEventDuration, 70
getEventDuration(), *22*, *135*
getExampleConnectionDetails, 72
getExampleConnectionDetails(), *6*, *14*, *86*, *130, 131*, *136*, *144*
getFullPredictionPerformances, 73
getFullPredictionPerformances(), *88, 89*, *91–93*, *95–97*, *99, 100*, *103–105*, *107*
getIncidenceRates, 75
getIncidenceRates(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *79*, *81*, *83, 84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getIncidenceTargetSettings, 78
getIncidenceTargetSettings(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77*, *81*, *83, 84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getNonCaseCounts, 79
getNonCaseCounts(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77*, *79*, *83, 84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getOutcomesUsedInCharacterization, 81
getOutcomesUsedInCharacterization(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77*, *79*, *81*, *84*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getOutcomesUsedInIncidence, 83
getOutcomesUsedInIncidence(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77*, *79*, *81*, *83*, *125*, *127, 128*, *133*, *137*, *141*, *147*
getOutcomeTable, 84
getOutcomeTable(), *6*, *14*, *72*, *130, 131*, *136*, *144*
getPredictionAggregateTopPredictors, 87
getPredictionAggregateTopPredictors(), *75*, *89*, *91–93*, *95–97*, *99, 100*, *103–105*, *107*
getPredictionCohorts, 88
getPredictionCohorts(), *75*, *88*, *91–93*, *95–97*, *99, 100*, *103–105*, *107*
getPredictionCovariates, 90
getPredictionCovariates(), *75*, *88, 89*, *92, 93*, *95–97*, *99, 100*, *103–105*, *107*
getPredictionDiagnostics, 91
getPredictionDiagnostics(), *75*, *88, 89*, *91*, *93*, *95–97*, *99, 100*, *103–105*, *107*
getPredictionDiagnosticTable, 93
getPredictionDiagnosticTable(), *75*, *88, 89*, *91, 92*, *95–97*, *99, 100*, *103–105*, *107*
getPredictionHyperParamSearch, 94
getPredictionHyperParamSearch(), *75*, *88, 89*, *91–93*, *96, 97*, *99, 100*, *103–105*, *107*
getPredictionIntercept, 95
getPredictionIntercept(), *75*, *88, 89*, *91–93*, *95*, *97*, *99, 100*, *103–105*, *107*
getPredictionLift, 96
getPredictionLift(), *75*, *88, 89*, *91–93*, *95, 96*, *99, 100*, *103–105*, *107*
getPredictionModelDesigns, 98
getPredictionModelDesigns(), *75*, *88, 89*, *91–93*, *95–97*, *100*, *103–105*, *107*
getPredictionOutcomes, 100
getPredictionOutcomes(), *75*, *88, 89*, *91–93*, *95–97*, *99*, *103–105*, *107*
getPredictionPerformances, 101

getPredictionPerformances(), *75*, *88, 89*, *91–93*, *95–97*, *99, 100*, *104, 105*, *107*
getPredictionPerformanceTable, 103
getPredictionPerformanceTable(), *75*, *88, 89*, *91–93*, *95–97*, *99, 100*, *103*, *105*, *107*
getPredictionTargets, 104
getPredictionTargets(), *75*, *88, 89*, *91–93*, *95–97*, *99, 100*, *103, 104*, *107*
getPredictionTopPredictors, 106
getPredictionTopPredictors(), *75*, *88, 89*, *91–93*, *95–97*, *99, 100*, *103–105*
getSccsDiagnosticsData, 108
getSccsDiagnosticsData(), *4, 5*, *37*, *39*, *42–44*, *46–48*, *112*, *114*, *116*, *118–121*, *123*, *139, 140*
getSccsEstimation, 110
getSccsEstimation(), *4, 5*, *37*, *39*, *42–44*, *46–48*, *110*, *114*, *116*, *118–121*, *123*, *139, 140*
getSccsMetaEstimation, 113
getSccsMetaEstimation(), *4, 5*, *37*, *39*, *42–44*, *46–48*, *110*, *112*, *116*, *118–121*, *123*, *139, 140*
getSccsModel, 115
getSccsModel(), *4, 5*, *37*, *39*, *42–44*, *46–48*, *110*, *112*, *114*, *118–121*, *123*, *139, 140*
getSccsNegativeControlEstimates, 116
getSccsNegativeControlEstimates(), *4, 5*, *37*, *39*, *42–44*, *46–48*, *110*, *112*, *114*, *116*, *119–121*, *123*, *139, 140*
getSccsOutcomes, 118
getSccsOutcomes(), *4, 5*, *37*, *39*, *42–44*, *46–48*, *110*, *112*, *114*, *116*, *118*, *120, 121*, *123*, *139, 140*
getSccsTable, 119
getSccsTable(), *4, 5*, *37*, *39*, *42–44*, *46–48*, *110*, *112*, *114*, *116*, *118, 119*, *121*, *123*, *139, 140*
getSccsTargets, 121
getSccsTargets(), *4, 5*, *37*, *39*, *42–44*, *46–48*, *110*, *112*, *114*, *116*, *118–120*, *123*, *139, 140*
getSccsTimeToEvent, 122
getSccsTimeToEvent(), *4, 5*, *37*, *39*, *42–44*, *46–48*, *110*, *112*, *114*, *116*, *118–121*, *139, 140*
getSubsetText, 123
getSubsetText(), *49, 50*, *52–58*, *142, 143*, *146*
getTargetCounts, 124
getTargetCounts(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77*, *79*, *81*, *83, 84*, *127, 128*, *133*, *137*, *141*, *147*
getTargetsUsedInCharacterization, 126
getTargetsUsedInCharacterization(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77*, *79*, *81*, *83, 84*, *125*, *128*, *133*, *137*, *141*, *147*
getTargetsUsedInIncidence, 127
getTargetsUsedInIncidence(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77*, *79*, *81*, *83, 84*, *125*, *127*, *133*, *137*, *141*, *147*
getTargetTable, 128
getTargetTable(), *6*, *14*, *72*, *86*, *131*, *136*, *144*
getTarText, 131
getTarText(), *6*, *14*, *72*, *86*, *130*, *136*, *144*
getTimeToEvent, 132
getTimeToEvent(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77*, *79*, *81*, *83, 84*, *125*, *127, 128*, *137*, *141*, *147*
getTreatmentPathways, 134
getTreatmentPathways(), *22*, *71*

kableDark, 135
kableDark(), *6*, *14*, *72*, *86*, *130, 131*, *144*

OhdsiReportGenerator, 136
OhdsiReportGenerator-package *(*OhdsiReportGenerator*)*, 136

plotAgeDistributions, 137
plotAgeDistributions(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77*, *79*, *81*, *83, 84*, *125*, *127, 128*, *133*, *141*, *147*
plotCmEstimates, 138
plotCmEstimates(), *4, 5*, *37*, *39*, *42–44*, *46–48*, *110*, *112*, *114*, *116*, *118–121*, *123*, *140*
plotSccsEstimates, 139

plotSccsEstimates(), *4, 5*, *37*, *39*, *42–44*, *46–48*, *110*, *112*, *114*, *116*, *118–121*, *123*, *139*
plotSexDistributions, 141
plotSexDistributions(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77*, *79*, *81*, *83, 84*, *125*, *127, 128*, *133*, *137*, *147*
processCohortDefinitionsForQuarto, 142
processCohortDefinitionsForQuarto(), *49, 50*, *52–58*, *124*, *143*, *146*
processCohorts, 143
processCohorts(), *49, 50*, *52–58*, *124*, *142*, *146*

removeSpaces, 144
removeSpaces(), *6*, *14*, *72*, *86*, *130, 131*, *136*
restrictCohortDefinitionsForQuarto, 144
restrictCohortDefinitionsForQuarto(), *49, 50*, *52–58*, *124*, *142, 143*

viewIncidenceRate, 146
viewIncidenceRate(), *8, 9*, *19*, *21*, *24*, *26*, *28*, *30*, *32*, *34, 35*, *60*, *62*, *64*, *68, 69*, *77*, *79*, *81*, *83, 84*, *125*, *127, 128*, *133*, *137*, *141*