

Package ‘OhdsiReportGenerator’

December 18, 2025

Type Package

Title Observational Health Data Sciences and Informatics Report Generator

Version 2.0.2

Date 2025-12-17

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

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

License Apache License 2.0

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

BugReports <https://github.com/OHDSI/OhdsiReportGenerator/issues>

VignetteBuilder knitr

Depends R (>= 3.3.0)

Imports CirceR, DatabaseConnector, forestplot, dplyr, fs, ggplot2, ggpubr, gt, htmltools, kableExtra, ParallelLogger, quarto, reactable (>= 0.4.4), rlang, rmarkdown, tibble, tidyr

Suggests knitr, markdown, ResultModelManager, RSQLite, testthat

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

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

Repository CRAN

Date/Publication 2025-12-17 23:00:19 UTC

Contents

addTarColumn	3
createPredictionReport	4
formatBinaryCovariateName	5
generateFullReport	6
generatePresentation	8
generatePresentationMultiple	10
generateSummaryPredictionReport	12
getBinaryCaseSeries	13
getBinaryRiskFactors	15
getCaseBinaryFeatures	17
getCaseContinuousFeatures	19
getCaseCounts	21
getCaseTargetBinaryFeatures	23
getCaseTargetCounts	25
getCharacterizationCohortBinary	27
getCharacterizationCohortContinuous	28
getCharacterizationDemographics	30
getCharacterizationOutcomes	32
getCharacterizationTargets	33
getCmDiagnosticsData	34
getCMEstimation	37
getCmMetaEstimation	39
getCmNegativeControlEstimates	41
getCmOutcomes	42
getCmPropensityModel	44
getCmTable	45
getCmTargets	46
getCohortCounts	48
getCohortDefinitions	49
getCohortInclusionRules	50
getCohortInclusionStats	51
getCohortInclusionSummary	52
getCohortMeta	53
getCohortSubsetDefinitions	54
getContinuousCaseSeries	55
getContinuousRiskFactors	57
getDatabaseDetails	59
getDechallengeRechallenge	60
getDechallengeRechallengeFails	62
getExampleConnectionDetails	64
getFullPredictionPerformances	65
getIncidenceOutcomes	67
getIncidenceRates	69
getIncidenceTargets	71
getOutcomeTable	72
getPredictionAggregateTopPredictors	74

getPredictionCohorts	76
getPredictionCovariates	77
getPredictionDiagnostics	78
getPredictionDiagnosticTable	80
getPredictionHyperParamSearch	81
getPredictionIntercept	83
getPredictionLift	84
getPredictionModelDesigns	85
getPredictionOutcomes	87
getPredictionPerformances	88
getPredictionPerformanceTable	90
getPredictionTargets	91
getPredictionTopPredictors	93
getSccsDiagnosticsData	95
getSccsEstimation	97
getSccsMetaEstimation	100
getSccsModel	102
getSccsNegativeControlEstimates	103
getSccsOutcomes	105
getSccsTable	106
getSccsTargets	108
getSccsTimeToEvent	109
getTargetBinaryFeatures	110
getTargetContinuousFeatures	112
getTargetTable	114
getTimeToEvent	116
kableDark	118
OhdsiReportGenerator	119
plotAgeDistributions	120
plotCmEstimates	121
plotSccsEstimates	122
plotSexDistributions	123
printReactable	125
processCohorts	126
removeSpaces	127
Index	128

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 separate TAR columns are found

See Also

Other helper: [formatBinaryCovariateName\(\)](#), [getExampleConnectionDetails\(\)](#), [getOutcomeTable\(\)](#), [getTargetTable\(\)](#), [kableDark\(\)](#), [printReactable\(\)](#), [removeSpaces\(\)](#)

Examples

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

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\(\)](#), [generatePresentation\(\)](#), [generatePresentationMultiple\(\)](#), [generateSummaryPredictionReport\(\)](#)

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 covariateName column changed to be more friendly

See Also

Other helper: [addTarColumn\(\)](#), [getExampleConnectionDetails\(\)](#), [getOutcomeTable\(\)](#), [getTargetTable\(\)](#), [kableDark\(\)](#), [printReactable\(\)](#), [removeSpaces\(\)](#)

Examples

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

generateFullReport	<i>generateFullReport</i>
--------------------	---------------------------

Description

Generates a full report from a Strategus analysis

Usage

```
generateFullReport(  
  server,  
  username,  
  password,  
  dbms,  
  resultsSchema = NULL,  
  targetId = 1,  
  outcomeIds = 3,  
  comparatorIds = 2,  
  indicationIds = "",  
  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. If no indication use "" and if you want some indications plus no indication use c("", indicationId1, indicationId2). Use 'Any' to include all children of targetId.
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\(\)](#), [generatePresentation\(\)](#), [generatePresentationMultiple\(\)](#), [generateSummaryPredictionReport\(\)](#)

`generatePresentation` *generatePresentation*

Description

Generates a presentation from a Strategus result

Usage

```
generatePresentation(
  server,
  username,
  password,
  dbms,
  resultsSchema = NULL,
  dbDetails = NULL,
  lead = "add name",
  team = "name 1 name 2",
  trigger = "A signal was found in spontaneous reports",
  safetyQuestion = "",
  objective = "",
  topline1 =
    "Very brief executive summary. You can copy-paste language from the conclusion.",
  topline2 =
    "If an estimation was requested but not feasible, this should be mentioned here.",
  topline3 =
    "If no estimation study was requested, this high-level summary might be skipped.",
  date = as.character(Sys.Date()),
```

```

targetId = 1,
outcomeIds = 3,
cohortNames = c("target name", "outcome name"),
cohortIds = c(1, 3),
covariateIds = NULL,
details = list(studyPeriod = "All Time", restrictions = "Age - None"),
evaluationText = "",
includeCI = TRUE,
includeCharacterization = TRUE,
includeCM = TRUE,
includeSCCS = TRUE,
includePLP = TRUE,
outputLocation,
outputName = paste0("presentation_", gsub(":", "_", gsub(" ", "_",
  as.character(date()))), ".html"),
intermediateDir = fs::path_real(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
dbDetails	(Optional) a data.frame with the columns:
lead	The name of the presenter
team	A vector or all the team members
trigger	What triggered the request
safetyQuestion	What is the general safety question
objective	What is the request/objective of the work.
topline1	add a very brief executive summary for the topline slide
topline2	add estimation summary here for the topline slide
topline3	add any other statement summary here for the topline slide
date	The date of the presentation
targetId	The cohort definition id for the target cohort
outcomeIds	The cohort definition id for the outcome
cohortNames	Friendly names for any cohort used in the study
cohortIds	The corresponding Ids for the cohortNames
covariateIds	A vector of covariateIds to include in the characterization
details	a list with the studyPeriod and restrictions

evaluationText a list of bullet points for the evaluation
 includeCI Whether to include the cohort incidence slides
 includeCharacterization Whether to include the characterization slides
 includeCM Whether to include the cohort method slides
 includeSCCS Whether to include the self controlled case series slides
 includePLP Whether to include the patient level prediction slides
 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 a presentation.

Value

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

See Also

Other Reporting: [createPredictionReport\(\)](#), [generateFullReport\(\)](#), [generatePresentationMultiple\(\)](#), [generateSummaryPredictionReport\(\)](#)

generatePresentationMultiple
generatePresentationMultiple

Description

Generates a presentation from a Strategus result

Usage

```

generatePresentationMultiple(
  server,
  username,
  password,
  dbms,
  resultsSchema = NULL,
  targetId = 1,
  targetName = "target cohort",
  cmSubsetId = 2,

```

```

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

```

Arguments

server	The server containing the result database
username	The username for an account that can access the result database
password	The password for an account that can access the result database
dbms	The dbms used to access the result database
resultsSchema	The result database schema
targetId	The cohort definition id for the target cohort
targetName	A friendly name for the target cohort
cmSubsetId	Optional a subset ID for the cohort method/prediction results
sccsSubsetId	Optional a subset ID for the SCCS and characterization results
indicationName	A name for the indication if used or NULL
outcomeIds	The cohort definition id for the outcome
outcomeNames	Friendly names for the outcomes
comparatorIds	The cohort method comparator cohort id
comparatorNames	Friendly names for the comparators
covariateIds	A vector of covariateIds to include in the characterization
details	a list with the studyPeriod and restrictions
title	A title for the presentation
lead	The name of the presentor
date	The date of the presentation
backgroundText	a character with any background text
evaluationText	a list of bullet points for the evaluation

outputLocation The file location and name to save the protocol

outputName The name of the html protocol that is created

intermediateDir
The work directory for quarto

Details

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

Value

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

See Also

Other Reporting: [createPredictionReport\(\)](#), [generateFullReport\(\)](#), [generatePresentation\(\)](#), [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\(\)](#), [generatePresentation\(\)](#), [generatePresentationMultiple\(\)](#)

getBinaryCaseSeries	<i>A function to extract case series characterization results</i>
---------------------	---

Description

A function to extract case series characterization results

Usage

```
getBinaryCaseSeries(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetId = NULL,
  outcomeId = NULL,
  databaseIds = NULL,
  riskWindowStart = NULL,
  riskWindowEnd = NULL,
  startAnchor = NULL,
  endAnchor = NULL,
  conceptIds = NULL,
  minVal = NULL
)
```

Arguments

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

Details

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

Value

A data.frame with the characterization case series results

See Also

Other Characterization: [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemographics\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

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

Description

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

Usage

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

Arguments

<code>connectionHandler</code>	A connection handler that connects to the database and extracts sql queries. Create a connection handler via <code>'ResultModelManager::ConnectionHandler\$new()'</code> .
<code>schema</code>	The result database schema (e.g., 'main' for sqlite)
<code>cTablePrefix</code>	The prefix used for the characterization results tables
<code>cgTablePrefix</code>	The prefix used for the cohort generator results tables
<code>databaseTable</code>	The name of the table with the database details (default 'database_meta_data')
<code>targetId</code>	An integer corresponding to the target cohort ID
<code>outcomeId</code>	An integer corresponding to the outcome cohort ID
<code>databaseId</code>	The database ID to restrict results to
<code>analysisIds</code>	The feature extraction analysis ID of interest (e.g., 201 is condition)
<code>riskWindowStart</code>	(optional) A vector of time-at-risk risk window starts to restrict to
<code>riskWindowEnd</code>	(optional) A vector of time-at-risk risk window ends to restrict to
<code>startAnchor</code>	(optional) A vector of time-at-risk start anchors to restrict to
<code>endAnchor</code>	(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: [getBinaryCaseSeries\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemographics\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

```

    outcomeId = 3
)

```

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

Description

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

Usage

```

getCaseBinaryFeatures(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL,
  databaseIds = NULL,
  analysisIds = c(3),
  riskWindowStart = NULL,
  riskWindowEnd = NULL,
  startAnchor = NULL,
  endAnchor = NULL
)

```

Arguments

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

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 offset 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 offset 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
- `covariateId` the id of the feature
- `covariateName` the name of the feature
- `sumValue` the number of cases who have the feature value of 1
- `averageValue` the mean feature value

See Also

Other Characterization: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemographics\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```

conDet <- getExampleConnectionDetails()

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

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

```

getCaseContinuousFeatures

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

Description

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

Usage

```

getCaseContinuousFeatures(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL,
  analysisIds = NULL,
  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')
targetIds	A vector of integers corresponding to the target cohort IDs
outcomeIds	A vector of integers corresponding to the outcome cohort IDs
analysisIds	The feature extraction analysis ID of interest (e.g., 201 is condition)
databaseIds	(optional) A vector of database IDs to restrict results 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

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 time at risk start point
- riskWindowEnd the time at risk end point
- startAnchor the time at risk start point offset
- endAnchor the time at risk end point offset
- covariateName the name of the feature
- covariateId the id of the feature
- countValue the number of cases who have the feature
- minValue the minimum value observed for the feature
- maxValue the maximum value observed for the feature
- averageValue the mean value observed for the feature
- standardDeviation the standard deviation of the value observed for the feature
- medianValue the median value observed for the feature
- p10Value the 10th percentile of the value observed for the feature
- p25Value the 25th percentile of the value observed for the feature
- p75Value the 75th percentile of the value observed for the feature
- p90Value the 90th percentile of the value observed for the feature

See Also

Other Characterization: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemographics\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

getCaseCounts

Extract the outcome cohort counts result

Description

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

Usage

```
getCaseCounts(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = NULL,
  databaseIds = NULL,
  riskWindowStart = NULL,
  riskWindowEnd = NULL,
  startAnchor = NULL,
  endAnchor = NULL
)
```

Arguments

<code>connectionHandler</code>	A connection handler that connects to the database and extracts sql queries. Create a connection handler via <code>'ResultModelManager::ConnectionHandler\$new()'</code> .
<code>schema</code>	The result database schema (e.g., 'main' for sqlite)
<code>cTablePrefix</code>	The prefix used for the characterization results tables
<code>cgTablePrefix</code>	The prefix used for the cohort generator results tables
<code>databaseTable</code>	The name of the table with the database details (default 'database_meta_data')
<code>targetIds</code>	A vector of integers corresponding to the target cohort IDs
<code>outcomeIds</code>	A vector of integers corresponding to the outcome cohort IDs
<code>databaseIds</code>	(optional) A vector of database IDs to restrict to
<code>riskWindowStart</code>	(optional) A vector of time-at-risk risk window starts to restrict to
<code>riskWindowEnd</code>	(optional) A vector of time-at-risk risk window ends to restrict to
<code>startAnchor</code>	(optional) A vector of time-at-risk start anchors to restrict to
<code>endAnchor</code>	(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

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
- `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 offset 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 offset the end anchor that is the end of the time-at-risk
- `endAnchor` the end anchor is either the target cohort start or cohort end date

See Also

Other Characterization: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemographic\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

getCaseTargetBinaryFeatures

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

Description

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

Usage

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

Arguments

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

Details

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

Value

Returns a `data.frame` with the columns:

- `databaseName` the name of the database
- `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)
- `covariateId` the id of the feature
- `covariateName` the name of the feature
- `sumValue` the number of target patients who have the feature value of 1 (minus those excluded due to having the outcome prior)
- `rawSum` the number of target patients who have the feature value of 1 (ignoring exclusions)
- `rawAverage` the fraction of target patients who have the feature value of 1 (ignoring exclusions)

See Also

Other Characterization: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemographics\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```

conDet <- getExampleConnectionDetails()

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

tbF <- getCaseTargetBinaryFeatures (
  connectionHandler = connectionHandler,
  schema = 'main'
)

```

getCaseTargetCounts	<i>Extract the target cohort counts result</i>
---------------------	--

Description

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

Usage

```

getCaseTargetCounts(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  outcomeIds = 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)
cTablePrefix	The prefix used for the characterization results tables
cgTablePrefix	The prefix used for the cohort generator results tables
databaseTable	The name of the table with the database details (default 'database_meta_data')
targetIds	A vector of integers corresponding to the target cohort IDs
outcomeIds	A vector of integers corresponding to the outcome cohort IDs
databaseIds	A vector of database IDs to restrict to

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
- `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
- `minPriorObservation` the minimum required observation days prior to index for an entry
- `outcomeWashoutDays` patients with the outcome occurring within this number of days prior to index are excluded (NA means no exclusion)

See Also

Other Characterization: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemographics\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

getCharacterizationCohortBinary

A function to extract cohort aggregate binary feature characterization results

Description

A function to extract cohort aggregate binary feature characterization results

Usage

```
getCharacterizationCohortBinary(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = 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')
targetIds	A vector of integers corresponding to the target cohort 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: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCaseTargetBinaryFeatures()`, `getCaseTargetCounts()`, `getCharacterizationCohortContinuous()`, `getCharacterizationDemographics()`, `getCharacterizationOutcomes()`, `getCharacterizationTargets()`, `getContinuousCaseSeries()`, `getContinuousRiskFactors()`, `getDechallengeRechallenge()`, `getDechallengeRechallengeFails()`, `getIncidenceOutcomes()`, `getIncidenceRates()`, `getIncidenceTargets()`, `getTargetBinaryFeatures()`, `getTargetContinuousFeatures()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

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

binCohort <- getCharacterizationCohortBinary(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetIds = 1,
  databaseIds = 'eunomia'
)
```

getCharacterizationCohortContinuous

A function to extract cohort aggregate continuous feature characterization results

Description

A function to extract cohort aggregate continuous feature characterization results

Usage

```
getCharacterizationCohortContinuous(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = 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')
targetIds	A vector of integers corresponding to the target cohort 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: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationDemographics\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

conCohort <- getCharacterizationCohortContinuous(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetIds = 1,
  databaseIds = 'eunomia'
)
```

getCharacterizationDemographics

Extract the binary age groups for the cases and targets

Description

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

Usage

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

Arguments

connectionHandler	A connection handler that connects to the database and extracts sql queries. Create a connection handler via 'ResultModelManager::ConnectionHandler\$new()'.
schema	The result database schema (e.g., 'main' for sqlite)
cTablePrefix	The prefix used for the characterization results tables
cgTablePrefix	The prefix used for the cohort generator results tables
databaseTable	The name of the table with the database details (default 'database_meta_data')
targetId	An integer corresponding to the target cohort ID
outcomeId	An 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 offset 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 offset the end anchor that is the end of the time-at-risk
- `endAnchor` the end anchor is either the target cohort start or cohort end date
- `covariateName` the name of the feature
- `sumValue` the number of cases who have the feature value of 1
- `averageValue` the mean feature value

See Also

Other Characterization: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
# example code

conDet <- getExampleConnectionDetails()

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

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

getCharacterizationOutcomes

A function to extract the outcomes found in characterization

Description

A function to extract the outcomes found in characterization

Usage

```
getCharacterizationOutcomes(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  targetId = NULL,
  printTimes = FALSE,
  useDcrc = TRUE,
  useTte = TRUE,
  useRf = 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
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

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: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemograph](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

getCharacterizationTargets

A function to extarct the targets found in characterization

Description

A function to extarct the targets found in characterization

Usage

```
getCharacterizationTargets(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  printTimes = FALSE,
  useTte = TRUE,
  useDcrc = TRUE,
  useRf = 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

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: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemography\(\)](#), [getCharacterizationOutcomes\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

getCmDiagnosticsData *Extract the cohort method diagnostic 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,
  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
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 unique 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` (deprecated) 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 equipoise diagnostic passed or failed.
- `mdrrDiagnostic` whether the mdrr (power) diagnostic passed or failed.
- `attritionDiagnostic` (deprecated) 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: `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	<i>Extract the cohort method results</i>
-----------------	--

Description

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

Usage

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

Arguments

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

Details

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

Value

Returns a data.frame with the columns:

- databaseName the name of the database
- databaseId the unique 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 occurring during the time at risk for the target cohort people
- comparatorOutcomes the total number of outcomes occurring 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: [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	<i>Extract the cohort method meta analysis results</i>
---------------------	--

Description

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

Usage

```

getCmMetaEstimation(
  connectionHandler,
  schema,
  cmTablePrefix = "cm_",
  cgTablePrefix = "cg_",
  esTablePrefix = "es_",
  targetIds = NULL,
  outcomeIds = NULL,
  comparatorIds = NULL,
  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
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 occurring during the time at risk for the target cohort people across included database
- `comparatorOutcomes` the total number of outcomes occurring 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: [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,
  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

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: [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	<i>A function to extract the outcomes found in cohort method</i>
---------------	--

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: [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,
  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
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: [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	<i>Extract the cohort method table specified</i>
------------	--

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,
  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
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: [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: [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'  
)
```

getCohortCounts	<i>Extract the cohort counts</i>
-----------------	----------------------------------

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: [getCohortDefinitions\(\)](#), [getCohortInclusionRules\(\)](#), [getCohortInclusionStats\(\)](#), [getCohortInclusionSummary\(\)](#), [getCohortMeta\(\)](#), [getCohortSubsetDefinitions\(\)](#), [processCohorts\(\)](#)

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: [getCohortCounts\(\)](#), [getCohortInclusionRules\(\)](#), [getCohortInclusionStats\(\)](#), [getCohortInclusionSummary\(\)](#), [getCohortMeta\(\)](#), [getCohortSubsetDefinitions\(\)](#), [processCohorts\(\)](#)

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: [getCohortCounts\(\)](#), [getCohortDefinitions\(\)](#), [getCohortInclusionStats\(\)](#), [getCohortInclusionSummary\(\)](#), [getCohortMeta\(\)](#), [getCohortSubsetDefinitions\(\)](#), [processCohorts\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

cohortIncluisionsRules <- 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: [getCohortCounts\(\)](#), [getCohortDefinitions\(\)](#), [getCohortInclusionRules\(\)](#), [getCohortInclusionSummary\(\)](#), [getCohortMeta\(\)](#), [getCohortSubsetDefinitions\(\)](#), [processCohorts\(\)](#)

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: [getCohortCounts\(\)](#), [getCohortDefinitions\(\)](#), [getCohortInclusionRules\(\)](#), [getCohortInclusionStats\(\)](#), [getCohortMeta\(\)](#), [getCohortSubsetDefinitions\(\)](#), [processCohorts\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

getCohortMeta	<i>Extract the cohort meta</i>
---------------	--------------------------------

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: [getCohortCounts\(\)](#), [getCohortDefinitions\(\)](#), [getCohortInclusionRules\(\)](#), [getCohortInclusionStats\(\)](#), [getCohortInclusionSummary\(\)](#), [getCohortSubsetDefinitions\(\)](#), [processCohorts\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

getCohortSubsetDefinitions

Extract the cohort subset definition details

Description

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

Usage

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

Arguments

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

Details

Specify the connectionHandler, the schema and the subset IDs

Value

Returns a data.frame with the cohort subset details

See Also

Other Cohorts: [getCohortCounts\(\)](#), [getCohortDefinitions\(\)](#), [getCohortInclusionRules\(\)](#), [getCohortInclusionStats\(\)](#), [getCohortInclusionSummary\(\)](#), [getCohortMeta\(\)](#), [processCohorts\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

getContinuousCaseSeries

A function to extract case series continuous feature characterization results

Description

A function to extract case series continuous feature characterization results

Usage

```

getContinuousCaseSeries(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetId = NULL,
  outcomeId = NULL,
  databaseIds = NULL,
  riskWindowStart = NULL,
  riskWindowEnd = NULL,
  startAnchor = NULL,
  endAnchor = NULL
)

```

Arguments

<code>connectionHandler</code>	A connection handler that connects to the database and extracts sql queries. Create a connection handler via 'ResultModelManager::ConnectionHandler\$new()'.
<code>schema</code>	The result database schema (e.g., 'main' for sqlite)
<code>cTablePrefix</code>	The prefix used for the characterization results tables
<code>cgTablePrefix</code>	The prefix used for the cohort generator results tables
<code>databaseTable</code>	The name of the table with the database details (default 'database_meta_data')
<code>targetId</code>	An integer corresponding to the target cohort ID
<code>outcomeId</code>	An integer corresponding to the outcome cohort ID
<code>databaseIds</code>	(optional) One or more unique identifiers for the databases
<code>riskWindowStart</code>	(optional) A riskWindowStart to restrict to
<code>riskWindowEnd</code>	(optional) A riskWindowEnd to restrict to
<code>startAnchor</code>	(optional) A startAnchor to restrict to
<code>endAnchor</code>	(optional) An endAnchor 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: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemograph](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

```
getContinuousRiskFactors
```

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

Description

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

Usage

```
getContinuousRiskFactors(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetId = NULL,
  outcomeId = NULL,
  analysisIds = NULL,
  databaseIds = NULL,
  riskWindowStart = NULL,
  riskWindowEnd = NULL,
  startAnchor = NULL,
  endAnchor = NULL
)
```

Arguments

<code>connectionHandler</code>	A connection handler that connects to the database and extracts sql queries. Create a connection handler via <code>'ResultModelManager::ConnectionHandler\$new()'</code> .
<code>schema</code>	The result database schema (e.g., 'main' for sqlite)
<code>cTablePrefix</code>	The prefix used for the characterization results tables
<code>cgTablePrefix</code>	The prefix used for the cohort generator results tables
<code>databaseTable</code>	The name of the table with the database details (default 'database_meta_data')
<code>targetId</code>	An integer corresponding to the target cohort ID
<code>outcomeId</code>	An integer corresponding to the outcome cohort ID
<code>analysisIds</code>	The feature extraction analysis ID of interest (e.g., 201 is condition)
<code>databaseIds</code>	(optional) A vector of database IDs to restrict to
<code>riskWindowStart</code>	(optional) A vector of time-at-risk risk window starts to restrict to
<code>riskWindowEnd</code>	(optional) A vector of time-at-risk risk window ends to restrict to
<code>startAnchor</code>	(optional) A vector of time-at-risk start anchors to restrict to
<code>endAnchor</code>	(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: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemography\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

```

        outcomeId = 3
    )

```

getDatabaseDetails	<i>Extract the database used in the analyses</i>
--------------------	--

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
- sourceReleaseDate the 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",
  targetIds = NULL,
  outcomeIds = NULL
)
```

Arguments

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

Details

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

Value

Returns a data.frame with the columns:

- `databaseName` the name of the database
- `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: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCaseTargetBinaryFeatures()`, `getCaseTargetCounts()`, `getCharacterizationCohortBinary()`, `getCharacterizationCohortContinuous()`, `getCharacterizationDemograph`, `getCharacterizationOutcomes()`, `getCharacterizationTargets()`, `getContinuousCaseSeries()`, `getContinuousRiskFactors()`, `getDechallengeRechallengeFails()`, `getIncidenceOutcomes()`, `getIncidenceRates()`, `getIncidenceTargets()`, `getTargetBinaryFeatures()`, `getTargetContinuousFeatures()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

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_",
  targetId = 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
targetId	An integer corresponding to the target cohort ID
outcomeId	An 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: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemography\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

conCohort <- getDechallengeRechallengeFails(
  connectionHandler = connectionHandler,
  schema = 'main',
  targetId = 1,
  outcomeId = 3,
  databaseId = 'eunomia'
)
```

`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\(\)](#), [getOutcomeTable\(\)](#), [getTargetTable\(\)](#), [kableDark\(\)](#), [printReactable\(\)](#), [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
- 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'
)
```

getIncidenceOutcomes *A function to extract the outcomes found in incidence*

Description

A function to extract the outcomes found in incidence

Usage

```
getIncidenceOutcomes(
  connectionHandler,
  schema,
  ciTablePrefix = "ci_",
  cgTablePrefix = "cg_",
  targetId = NULL
)
```

Arguments

<code>connectionHandler</code>	A connection handler that connects to the database and extracts sql queries. Create a connection handler via <code>'ResultModelManager::ConnectionHandler\$new()'</code> .
<code>schema</code>	The result database schema (e.g., 'main' for sqlite)
<code>ciTablePrefix</code>	The prefix used for the cohort incidence results tables
<code>cgTablePrefix</code>	The prefix used for the cohort generator results tables
<code>targetId</code>	An integer corresponding to the target cohort ID

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: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemography\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

getIncidenceRates	<i>Extract the cohort incidence result</i>
-------------------	--

Description

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

Usage

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

Arguments

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

Details

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

Value

Returns a data.frame with the columns:

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

See Also

Other Characterization: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemography\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

getIncidenceTargets *A function to extract the targets found in incidence*

Description

A function to extract the targets found in incidence

Usage

```
getIncidenceTargets(
  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: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemographic\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```

conDet <- getExampleConnectionDetails()

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

cohorts <- getIncidenceTargets(
  connectionHandler = connectionHandler,
  schema = 'main'
)

```

getOutcomeTable	<i>Extract the outcome cohorts and where they are used in the analyses.</i>
-----------------	---

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\(\)](#), [getExampleConnectionDetails\(\)](#), [getTargetTable\(\)](#), [kableDark\(\)](#), [printReactable\(\)](#), [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\(\)](#), [getPredictionModelDesign\(\)](#), [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\(\)](#), [getPredictionModelDesign\(\)](#), [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 participants 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()`, `getPredictionModelDesign()`, `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

<code>connectionHandler</code>	A connection handler that connects to the database and extracts sql queries. Create a connection handler via <code>'ResultModelManager::ConnectionHandler\$new()'</code> .
<code>schema</code>	The result database schema (e.g., <code>'main'</code> 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\(\)](#), [getPredictionHyperParamSearch\(\)](#), [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	<i>Extract model lift at given model sensitivity</i>
-------------------	--

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 survival 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\(\)](#), [getPredictionPerformanceTable\(\)](#), [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
- 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\(\)](#), [getPredictionPerformanceTable\(\)](#), [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

<code>connectionHandler</code>	A connection handler that connects to the database and extracts sql queries. Create a connection handler via <code>'ResultModelManager::ConnectionHandler\$new()'</code> .
<code>schema</code>	The result database schema (e.g., 'main' for sqlite)
<code>plpTablePrefix</code>	The prefix used for the patient level prediction results tables
<code>cgTablePrefix</code>	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\(\)](#), [getPredictionPerformanceTable\(\)](#), [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\(\)](#), [getPredictionPerformanceTable\(\)](#), [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 (scs) diagnostic results

Description

This function extracts the scs 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: [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	<i>Extract the self controlled case series (scCs) results</i>
-------------------	---

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 of 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).
- mdr 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: [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: [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	<i>Extract the SCCS model table</i>
--------------	-------------------------------------

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: [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

<code>connectionHandler</code>	A connection handler that connects to the database and extracts sql queries. Create a connection handler via <code>'ResultModelManager::ConnectionHandler\$new()'</code> .
<code>schema</code>	The result database schema (e.g., 'main' for sqlite)
<code>sccsTablePrefix</code>	The prefix used for the cohort generator results tables
<code>cgTablePrefix</code>	The prefix used for the cohort generator results tables
<code>databaseTable</code>	The name of the table with the database details (default 'database_meta_data')
<code>databaseIds</code>	the database IDs to restrict to
<code>exposuresOutcomeSetIds</code>	the exposureOutcomeIds to restrict to
<code>indicationIds</code>	The indications that the target was nested to
<code>outcomeIds</code>	A vector of integers corresponding to the outcome cohort IDs
<code>targetIds</code>	A vector of integers corresponding to the target cohort IDs
<code>analysisIds</code>	the analysis IDs to restrict to
<code>covariateIds</code>	the covariate IDs to restrict to
<code>covariateAnalysisIds</code>	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: `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	<i>A function to extract the outcomes found in self controlled case series</i>
-----------------	--

Description

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

Usage

```
getSccsOutcomes(
  connectionHandler,
  schema,
  sccsTablePrefix = "sccs_",
  cgTablePrefix = "cg_",
  targetId = NULL
)
```

Arguments

<code>connectionHandler</code>	A connection handler that connects to the database and extracts sql queries. Create a connection handler via <code>'ResultModelManager::ConnectionHandler\$new()'</code> .
<code>schema</code>	The result database schema (e.g., 'main' for sqlite)
<code>sccsTablePrefix</code>	The prefix used for the cohort generator results tables
<code>cgTablePrefix</code>	The prefix used for the cohort generator results tables
<code>targetId</code>	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: `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	<i>Extract the SCCS table specified</i>
--------------	---

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: [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: [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	<i>Extract the SCCS time-to-event</i>
--------------------	---------------------------------------

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: [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'
)

```

getTargetBinaryFeatures

Extract aggregate statistics of binary feature analysis IDs of interest for targets (ignoring excluding people with prior outcome)

Description

This function extracts the feature extraction results for targets corresponding to specified target but does not exclude any patients with the outcome during the outcome washout (so it agnostic to the outcome of interest).

Usage

```
getTargetBinaryFeatures(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetId = NULL,
  databaseIds = NULL,
  analysisIds = NULL,
  conceptIds = NULL
)
```

Arguments

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

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 minimum required observation days prior to index for an entry
- covariateId the id of the feature
- covariateName the name of the feature
- sumValue the number of target patients who have the feature value of 1 (target patients are restricted to first occurrence and require min prior observation days)
- averageAvalue the fraction of target patients who have the feature value of 1 (target patients are restricted to first occurrence and require min prior observation days)

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCaseTargetBinaryFeatures()`, `getCaseTargetCounts()`, `getCharacterizationCohortBinary()`, `getCharacterizationCohortContinuous()`, `getCharacterizationDemographic()`, `getCharacterizationOutcomes()`, `getCharacterizationTargets()`, `getContinuousCaseSeries()`, `getContinuousRiskFactors()`, `getDechallengeRechallenge()`, `getDechallengeRechallengeFails()`, `getIncidenceOutcomes()`, `getIncidenceRates()`, `getIncidenceTargets()`, `getTargetContinuousFeatures()`, `getTimeToEvent()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

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

tbf <- getTargetBinaryFeatures (
  connectionHandler = connectionHandler,
  schema = 'main',
  targetId = 1
)
```

getTargetContinuousFeatures

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

Description

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

Usage

```
getTargetContinuousFeatures(
  connectionHandler,
  schema,
  cTablePrefix = "c_",
  cgTablePrefix = "cg_",
  databaseTable = "database_meta_data",
  targetIds = NULL,
  analysisIds = NULL,
  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)
cTablePrefix	The prefix used for the characterization results tables
cgTablePrefix	The prefix used for the cohort generator results tables
databaseTable	The name of the table with the database details (default 'database_meta_data')
targetIds	A vector of integers corresponding to the target cohort IDs
analysisIds	The feature extraction analysis ID of interest (e.g., 201 is condition)
databaseIds	(Optional) A vector of database IDs to restrict to

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: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemograph](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

tcf <- getTargetContinuousFeatures(
  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 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 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\(\)](#), [getExampleConnectionDetails\(\)](#), [getOutcomeTable\(\)](#), [kableDark\(\)](#), [printReactable\(\)](#), [removeSpaces\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

getTimeToEvent

Extract the time to event result

Description

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

Usage

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

Arguments

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

Details

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

Value

Returns a data.frame with the columns:

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

See Also

Other Characterization: `getBinaryCaseSeries()`, `getBinaryRiskFactors()`, `getCaseBinaryFeatures()`, `getCaseContinuousFeatures()`, `getCaseCounts()`, `getCaseTargetBinaryFeatures()`, `getCaseTargetCounts()`, `getCharacterizationCohortBinary()`, `getCharacterizationCohortContinuous()`, `getCharacterizationDemograph`, `getCharacterizationOutcomes()`, `getCharacterizationTargets()`, `getContinuousCaseSeries()`, `getContinuousRiskFactors()`, `getDechallengeRechallenge()`, `getDechallengeRechallengeFails()`, `getIncidenceOutcomes()`, `getIncidenceRates()`, `getIncidenceTargets()`, `getTargetBinaryFeatures()`, `getTargetContinuousFeatures()`, `plotAgeDistributions()`, `plotSexDistributions()`

Examples

```
conDet <- getExampleConnectionDetails()

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

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

kableDark	<i>output a nicely formatted html table</i>
-----------	---

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\(\)](#), [getExampleConnectionDetails\(\)](#), [getOutcomeTable\(\)](#), [getTargetTable\(\)](#), [printReactable\(\)](#), [removeSpaces\(\)](#)

Examples

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

OhdsiReportGenerator *OhdsiReportGenerator*

Description

A package for extracting analyses results and creating reports.

Author(s)

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

Authors:

- Anthony Sena <sena@ohdsi.org>

See Also

Useful links:

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

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

Description

Creates bar charts for the target and case age groups.

Usage

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

Arguments

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

Details

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

Value

Returns a ggplot with the distributions

See Also

Other Characterization: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemographics\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotSexDistributions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

plotAgeDistributions(ageData = ageData)
```

plotCmEstimates	<i>Plots the cohort method results for one analysis</i>
-----------------	---

Description

Creates nice cohort method plots

Usage

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

Arguments

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

Details

Input the cohort method data

Value

Returns a ggplot with the estimates

See Also

Other Estimation: `getCMEstimation()`, `getCmDiagnosticsData()`, `getCmMetaEstimation()`, `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 = 1,
  outcomeIds = 3
)
plotCmEstimates(
  cmData = cmEst,
  cmMeta = NULL,
  targetName = 'target',
  comparatorName = 'comp',
  selectedAnalysisId = 1
)
```

<code>plotSccsEstimates</code>	<i>Plots the self controlled case series results for one analysis</i>
--------------------------------	---

Description

Creates nice self controlled case series plots

Usage

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

Arguments

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

Details

Input the self controlled case series data

Value

Returns a ggplot with the estimates

See Also

Other Estimation: [getCMEstimation\(\)](#), [getCmDiagnosticsData\(\)](#), [getCmMetaEstimation\(\)](#), [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,
  targetName = 'target',
  selectedAnalysisId = 1
)
```

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

Description

Creates bar charts for the target and case sex.

Usage

```
plotSexDistributions(
  sexData,
  riskWindowStart = "1",
  riskWindowEnd = "365",
```

```

    startAnchor = "cohort start",
    endAnchor = "cohort start"
  )

```

Arguments

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

Details

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

Value

Returns a ggplot with the distributions

See Also

Other Characterization: [getBinaryCaseSeries\(\)](#), [getBinaryRiskFactors\(\)](#), [getCaseBinaryFeatures\(\)](#), [getCaseContinuousFeatures\(\)](#), [getCaseCounts\(\)](#), [getCaseTargetBinaryFeatures\(\)](#), [getCaseTargetCounts\(\)](#), [getCharacterizationCohortBinary\(\)](#), [getCharacterizationCohortContinuous\(\)](#), [getCharacterizationDemographics\(\)](#), [getCharacterizationOutcomes\(\)](#), [getCharacterizationTargets\(\)](#), [getContinuousCaseSeries\(\)](#), [getContinuousRiskFactors\(\)](#), [getDechallengeRechallenge\(\)](#), [getDechallengeRechallengeFails\(\)](#), [getIncidenceOutcomes\(\)](#), [getIncidenceRates\(\)](#), [getIncidenceTargets\(\)](#), [getTargetBinaryFeatures\(\)](#), [getTargetContinuousFeatures\(\)](#), [getTimeToEvent\(\)](#), [plotAgeDistributions\(\)](#)

Examples

```

conDet <- getExampleConnectionDetails()

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

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

```

printReactable	<i>prints a reactable in a quarto document</i>
----------------	--

Description

This function lets you print a reactable in a quarto document

Usage

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

Arguments

data	The data for the table
columns	The formatting for the columns
groupBy	A column or columns to group the table by
defaultPageSize	The number of rows in the table
highlight	whether to highlight the row of interest
striped	whether the rows change color to give a striped appearance
searchable	whether you can search in the table
filterable	whether you can filter the table

Details

Input the values for `reactable::reactable`

Value

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

See Also

Other helper: [addTarColumn\(\)](#), [formatBinaryCovariateName\(\)](#), [getExampleConnectionDetails\(\)](#), [getOutcomeTable\(\)](#), [getTargetTable\(\)](#), [kableDark\(\)](#), [removeSpaces\(\)](#)

Examples

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

processCohorts	<i>Extract the cohort parents and children cohorts (cohorts derieved from the parent cohort)</i>
----------------	--

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: [getCohortCounts\(\)](#), [getCohortDefinitions\(\)](#), [getCohortInclusionRules\(\)](#), [getCohortInclusionStats\(\)](#), [getCohortInclusionSummary\(\)](#), [getCohortMeta\(\)](#), [getCohortSubsetDefinitions\(\)](#)

Examples

```
conDet <- getExampleConnectionDetails()

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

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

parents <- processCohorts(cohortDef)
```

removeSpaces	<i>removeSpaces</i>
--------------	---------------------

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\(\)](#), [getExampleConnectionDetails\(\)](#), [getOutcomeTable\(\)](#), [getTargetTable\(\)](#), [kableDark\(\)](#), [printReactable\(\)](#)

Examples

removeSpaces(' made up. string')

Index

* Characterization

- getBinaryCaseSeries, [13](#)
- getBinaryRiskFactors, [15](#)
- getCaseBinaryFeatures, [17](#)
- getCaseContinuousFeatures, [19](#)
- getCaseCounts, [21](#)
- getCaseTargetBinaryFeatures, [23](#)
- getCaseTargetCounts, [25](#)
- getCharacterizationCohortBinary, [27](#)
- getCharacterizationCohortContinuous, [28](#)
- getCharacterizationDemographics, [30](#)
- getCharacterizationOutcomes, [32](#)
- getCharacterizationTargets, [33](#)
- getContinuousCaseSeries, [55](#)
- getContinuousRiskFactors, [57](#)
- getDechallengeRechallenge, [60](#)
- getDechallengeRechallengeFails, [62](#)
- getIncidenceOutcomes, [67](#)
- getIncidenceRates, [69](#)
- getIncidenceTargets, [71](#)
- getTargetBinaryFeatures, [110](#)
- getTargetContinuousFeatures, [112](#)
- getTimeToEvent, [116](#)
- plotAgeDistributions, [120](#)
- plotSexDistributions, [123](#)

* Cohorts

- getCohortCounts, [48](#)
- getCohortDefinitions, [49](#)
- getCohortInclusionRules, [50](#)
- getCohortInclusionStats, [51](#)
- getCohortInclusionSummary, [52](#)
- getCohortMeta, [53](#)
- getCohortSubsetDefinitions, [54](#)
- processCohorts, [126](#)

* Database

- getDatabaseDetails, [59](#)

* Estimation

- getCmDiagnosticsData, [34](#)
- getCMEstimation, [37](#)
- getCmMetaEstimation, [39](#)
- getCmNegativeControlEstimates, [41](#)
- getCmOutcomes, [42](#)
- getCmPropensityModel, [44](#)
- getCmTable, [45](#)
- getCmTargets, [46](#)
- getSccsDiagnosticsData, [95](#)
- getSccsEstimation, [97](#)
- getSccsMetaEstimation, [100](#)
- getSccsModel, [102](#)
- getSccsNegativeControlEstimates, [103](#)
- getSccsOutcomes, [105](#)
- getSccsTable, [106](#)
- getSccsTargets, [108](#)
- getSccsTimeToEvent, [109](#)
- plotCmEstimates, [121](#)
- plotSccsEstimates, [122](#)

* Prediction

- getFullPredictionPerformances, [65](#)
- getPredictionAggregateTopPredictors, [74](#)
- getPredictionCohorts, [76](#)
- getPredictionCovariates, [77](#)
- getPredictionDiagnostics, [78](#)
- getPredictionDiagnosticTable, [80](#)
- getPredictionHyperParamSearch, [81](#)
- getPredictionIntercept, [83](#)
- getPredictionLift, [84](#)
- getPredictionModelDesigns, [85](#)
- getPredictionOutcomes, [87](#)
- getPredictionPerformances, [88](#)
- getPredictionPerformanceTable, [90](#)
- getPredictionTargets, [91](#)
- getPredictionTopPredictors, [93](#)

* Reporting

- createPredictionReport, 4
- generateFullReport, 6
- generatePresentation, 8
- generatePresentationMultiple, 10
- generateSummaryPredictionReport, 12
- * helper**
 - addTarColumn, 3
 - formatBinaryCovariateName, 5
 - getExampleConnectionDetails, 64
 - getOutcomeTable, 72
 - getTargetTable, 114
 - kableDark, 118
 - printReactable, 125
 - removeSpaces, 127
- addTarColumn, 3, 6, 64, 74, 116, 119, 125, 127
- createPredictionReport, 4, 8, 10, 12, 13
- formatBinaryCovariateName, 4, 5, 64, 74, 116, 119, 125, 127
- generateFullReport, 5, 6, 10, 12, 13
- generatePresentation, 5, 8, 12, 13
- generatePresentationMultiple, 5, 8, 10, 10, 13
- generateSummaryPredictionReport, 5, 8, 10, 12, 12
- getBinaryCaseSeries, 13, 16, 18, 21, 23, 24, 26, 28, 29, 31, 33, 34, 57, 58, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- getBinaryRiskFactors, 15, 15, 18, 21, 23, 24, 26, 28, 29, 31, 33, 34, 57, 58, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- getCaseBinaryFeatures, 15, 16, 17, 21, 23, 24, 26, 28, 29, 31, 33, 34, 57, 58, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- getCaseContinuousFeatures, 15, 16, 18, 19, 23, 24, 26, 28, 29, 31, 33, 34, 57, 58, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- getCaseCounts, 15, 16, 18, 21, 21, 24, 26, 28, 29, 31, 33, 34, 57, 58, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- getCaseTargetBinaryFeatures, 15, 16, 18, 21, 23, 23, 26, 28, 29, 31, 33, 34, 57, 58, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- getCaseTargetCounts, 15, 16, 18, 21, 23, 24, 25, 28, 29, 31, 33, 34, 57, 58, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- getCharacterizationCohortBinary, 15, 16, 18, 21, 23, 24, 26, 27, 29, 31, 33, 34, 57, 58, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- getCharacterizationCohortContinuous, 15, 16, 18, 21, 23, 24, 26, 28, 28, 31, 33, 34, 57, 58, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- getCharacterizationDemographics, 15, 16, 18, 21, 23, 24, 26, 28, 29, 30, 33, 34, 57, 58, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- getCharacterizationOutcomes, 15, 16, 18, 21, 23, 24, 26, 28, 29, 31, 32, 34, 57, 58, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- getCharacterizationTargets, 15, 16, 18, 21, 23, 24, 26, 28, 29, 31, 33, 33, 57, 58, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- getCmDiagnosticsData, 34, 38, 40, 42–44, 46, 47, 97, 99, 101, 103, 105–108, 110, 122, 123
- getCMEstimation, 36, 37, 40, 42–44, 46, 47, 97, 99, 101, 103, 105–108, 110, 122, 123
- getCmMetaEstimation, 36, 38, 39, 42–44, 46, 47, 97, 99, 101, 103, 105–108, 110, 122, 123
- getCmNegativeControlEstimates, 36, 38, 40, 41, 43, 44, 46, 47, 97, 99, 101, 103, 105–108, 110, 122, 123
- getCmOutcomes, 36, 38, 40, 42, 42, 44, 46, 47, 97, 99, 101, 103, 105–108, 110, 122, 123
- getCmPropensityModel, 36, 38, 40, 42, 43, 44, 46, 47, 97, 99, 101, 103, 105–108, 110, 122, 123
- getCmTable, 36, 38, 40, 42–44, 45, 47, 97, 99, 101, 103, 105–108, 110, 122, 123
- getCmTargets, 36, 38, 40, 42–44, 46, 46, 97, 99, 101, 103, 105–108, 110, 122, 123
- getCohortCounts, 48, 49, 51–55, 126

- `getCohortDefinitions`, 48, 49, 51–55, 126
- `getCohortInclusionRules`, 48, 49, 50, 52–55, 126
- `getCohortInclusionStats`, 48, 49, 51, 51, 53–55, 126
- `getCohortInclusionSummary`, 48, 49, 51, 52, 52, 54, 55, 126
- `getCohortMeta`, 48, 49, 51–53, 53, 55, 126
- `getCohortSubsetDefinitions`, 48, 49, 51–54, 54, 126
- `getContinuousCaseSeries`, 15, 16, 18, 21, 23, 24, 26, 28, 29, 31, 33, 34, 55, 58, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- `getContinuousRiskFactors`, 15, 16, 18, 21, 23, 24, 26, 28, 29, 31, 33, 34, 57, 57, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- `getDatabaseDetails`, 59
- `getDechallengeRechallenge`, 15, 16, 18, 21, 23, 24, 26, 28, 29, 31, 33, 34, 57, 58, 60, 63, 68, 70, 71, 112, 114, 118, 120, 124
- `getDechallengeRechallengeFails`, 15, 16, 18, 21, 23, 24, 26, 28, 29, 31, 33, 34, 57, 58, 62, 62, 68, 70, 71, 112, 114, 118, 120, 124
- `getExampleConnectionDetails`, 4, 6, 64, 74, 116, 119, 125, 127
- `getFullPredictionPerformances`, 65, 75, 77, 78, 80–83, 85, 86, 88, 90–92, 94
- `getIncidenceOutcomes`, 15, 16, 18, 21, 23, 24, 26, 28, 29, 31, 33, 34, 57, 58, 62, 63, 67, 70, 71, 112, 114, 118, 120, 124
- `getIncidenceRates`, 15, 16, 18, 21, 23, 24, 26, 28, 29, 31, 33, 34, 57, 58, 62, 63, 68, 69, 71, 112, 114, 118, 120, 124
- `getIncidenceTargets`, 15, 16, 18, 21, 23, 24, 26, 28, 29, 31, 33, 34, 57, 58, 62, 63, 68, 70, 71, 112, 114, 118, 120, 124
- `getOutcomeTable`, 4, 6, 64, 72, 116, 119, 125, 127
- `getPredictionAggregateTopPredictors`, 67, 74, 77, 78, 80–83, 85, 86, 88, 90–92, 94
- `getPredictionCohorts`, 67, 75, 76, 78, 80–83, 85, 86, 88, 90–92, 94
- `getPredictionCovariates`, 67, 75, 77, 77, 80–83, 85, 86, 88, 90–92, 94
- `getPredictionDiagnostics`, 67, 75, 77, 78, 78, 81–83, 85, 86, 88, 90–92, 94
- `getPredictionDiagnosticTable`, 67, 75, 77, 78, 80, 80, 82, 83, 85, 86, 88, 90–92, 94
- `getPredictionHyperParamSearch`, 67, 75, 77, 78, 80, 81, 81, 83, 85, 86, 88, 90–92, 94
- `getPredictionIntercept`, 67, 75, 77, 78, 80–82, 83, 85, 86, 88, 90–92, 94
- `getPredictionLift`, 67, 75, 77, 78, 80–83, 84, 86, 88, 90–92, 94
- `getPredictionModelDesigns`, 67, 75, 77, 78, 80–83, 85, 85, 88, 90–92, 94
- `getPredictionOutcomes`, 67, 75, 77, 78, 80–83, 85, 86, 87, 90–92, 94
- `getPredictionPerformances`, 67, 75, 77, 78, 80–83, 85, 86, 88, 88, 91, 92, 94
- `getPredictionPerformanceTable`, 67, 75, 77, 78, 80–83, 85, 86, 88, 90, 90, 92, 94
- `getPredictionTargets`, 67, 75, 77, 78, 80–83, 85, 86, 88, 90, 91, 91, 94
- `getPredictionTopPredictors`, 67, 75, 77, 78, 80–83, 85, 86, 88, 90–92, 93
- `getSccsDiagnosticsData`, 36, 38, 40, 42–44, 46, 47, 95, 99, 101, 103, 105–108, 110, 122, 123
- `getSccsEstimation`, 36, 38, 40, 42–44, 46, 47, 97, 97, 101, 103, 105–108, 110, 122, 123
- `getSccsMetaEstimation`, 36, 38, 40, 42–44, 46, 47, 97, 99, 100, 103, 105–108, 110, 122, 123
- `getSccsModel`, 36, 38, 40, 42–44, 46, 47, 97, 99, 101, 102, 105–108, 110, 122, 123
- `getSccsNegativeControlEstimates`, 36, 38, 40, 42–44, 46, 47, 97, 99, 101, 103, 103, 106–108, 110, 122, 123
- `getSccsOutcomes`, 36, 38, 40, 42–44, 46, 47, 97, 99, 101, 103, 105, 105, 107, 108, 110, 122, 123
- `getSccsTable`, 36, 38, 40, 42–44, 46, 47, 97, 99, 101, 103, 105, 106, 106, 108, 110, 122, 123
- `getSccsTargets`, 36, 38, 40, 42–44, 46, 47,

97, 99, 101, 103, 105–107, 108, 110,
122, 123

getSccsTimeToEvent, 36, 38, 40, 42–44, 46,
47, 97, 99, 101, 103, 105–108, 109,
122, 123

getTargetBinaryFeatures, 15, 16, 18, 21,
23, 24, 26, 28, 29, 31, 33, 34, 57, 58,
62, 63, 68, 70, 71, 110, 114, 118,
120, 124

getTargetContinuousFeatures, 15, 16, 18,
21, 23, 24, 26, 28, 29, 31, 33, 34, 57,
58, 62, 63, 68, 70, 71, 112, 112, 118,
120, 124

getTargetTable, 4, 6, 64, 74, 114, 119, 125,
127

getTimeToEvent, 15, 16, 18, 21, 23, 24, 26,
28, 29, 31, 33, 34, 57, 58, 62, 63, 68,
70, 71, 112, 114, 116, 120, 124

kableDark, 4, 6, 64, 74, 116, 118, 125, 127

OhdsiReportGenerator, 119

OhdsiReportGenerator-package
(OhdsiReportGenerator), 119

plotAgeDistributions, 15, 16, 18, 21, 23,
24, 26, 28, 29, 31, 33, 34, 57, 58, 62,
63, 68, 70, 71, 112, 114, 118, 120,
124

plotCmEstimates, 36, 38, 40, 42–44, 46, 47,
97, 99, 101, 103, 105–108, 110, 121,
123

plotSccsEstimates, 36, 38, 40, 42–44, 46,
47, 97, 99, 101, 103, 105–108, 110,
122, 122

plotSexDistributions, 15, 16, 18, 21, 23,
24, 26, 28, 29, 31, 33, 34, 57, 58, 62,
63, 68, 70, 71, 112, 114, 118, 120,
123

printReactable, 4, 6, 64, 74, 116, 119, 125,
127

processCohorts, 48, 49, 51–55, 126

removeSpaces, 4, 6, 64, 74, 116, 119, 125, 127