Package: SelfControlledCohort (via r-universe)

October 11, 2026

Type Package

Title Self-Controlled Cohort Population-Level Estimation

Version 2.1.0

Date 2026-09-07

Maintainer Jamie Gilbert <gilbert@ohdsi.org>

Description Estimates incidence rate ratios by comparing time exposed with time unexposed among an exposed cohort using self-controlled cohort methodology as described in Ryan et al. (2013) <doi:10.1002/pds.3457>. Functions used for empirical calibration of effect estimates, confidence intervals, and p-values are included to control for residual bias.

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

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

Language en-US

Depends DatabaseConnector (>= 5.0.0), R (>= 4.1.0)

Imports SqlRender (>= 1.4.3), ParallelLogger, rateratio.test, checkmate, dplyr, EmpiricalCalibration, ResultModelManager, Andromeda, readr, rlang, cli, stats, CohortGenerator

Suggests withr, testthat, knitr, rmarkdown, Eunomia, duckdb, DiagrammeR, tidyr, R.utils, jsonlite, DT, ggplot2, shiny, shinycssloaders, plotly, reactable, shinyWidgets

License Apache License 2.0

VignetteBuilder knitr

Encoding UTF-8

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Jamie Gilbert [cre, aut] (ORCID:
<https://orcid.org/0000-0003-2294-3459>), Martijn Schuemie
[aut], Patrick Ryan [aut], Observational Health Data Science
and Informatics [cph]

Repository https://cran.r-universe.dev

Date/Publication 2026-10-07 16:40:18 UTC

RemoteUrl https://github.com/cran/SelfControlledCohort

RemoteRef HEAD

RemoteSha f438a88046a4a61ed6d840679e740aa1f37f13b0

Contents

computeEase . . . 3
computeMdrrForRateRatio . . . 4
createExposureOutcome . . . 5
createResultsDataModel . . . 6
createRunSelfControlledCohortArgs . . . 7
createSccAnalysis . . . 9
createSelfControlledCohortModuleSpecifications . . . 10
execute . . . 11
getDataMigrator . . . 13
getDefaultDiagnosticThresholds . . . 14
getDefaultExportManager . . . 15
getDiagnosticsSummary . . . 15
getModuleInfo . . . 16
getResultsDataModelSpecifications . . . 17
getResultsFolders . . . 17
getSccRiskWindowStats . . . 18
loadExposureOutcomeList . . . 20
loadSccAnalysisList . . . 20
migrateDataModel . . . 21
runSccAnalyses . . . 22
runSccDiagnostics . . . 24
runSccRiskWindows . . . 26
runSelfControlledCohort . . . 28
saveExposureOutcomeList . . . 32
saveSccAnalysisList . . . 33
uploadResults . . . 34

Index **36**

computeEase *Compute Expected Absolute Systematic Error (EASE)*

Description

Computes the expected absolute systematic error from the null distribution fitted on negative control estimates. EASE summarizes both bias (mean of null) and imprecision (spread of null) into a single metric.

Usage

```
computeEase(negatives)
```

Arguments

negatives	Data frame of negative control results with columns ‘rr‘ and ‘seLogRr‘

Details

EASE is computed by fitting a null distribution to the negative control log rate ratios using `EmpiricalCalibration::fitNul` then calling `EmpiricalCalibration::computeExpectedAbsoluteSystematicError()`.

Lower values indicate less systematic error. A value of 0 means no detectable bias. The default threshold of 0.25 is aligned with SCCS package conventions.

Value

Numeric EASE value, or NA if the null distribution could not be fitted.

References

Schuemie MJ, Hripcsak G, Ryan PB, Madigan D, Suchard MA. Empirical confidence interval calibration for population-level effect estimation studies in observational healthcare data. PNAS. 2018;115(11):2571-2577.

Examples

```
if (interactive()) {
# Compute EASE from negative control results
negativeControls <- data.frame(
  rr = c(0.95, 1.02, 0.98, 1.05),
  seLogRr = c(0.2, 0.18, 0.22, 0.19)
)
ease <- computeEase(negativeControls)
}
```

`computeMdrrForRateRatio` *Compute Minimum Detectable Relative Risk (MDRR) for rate ratio*

Description

Calculates the minimum detectable relative risk for a two-sample Poisson rate comparison using the Signed Root Likelihood (SRL1) method described by Musonda et al. (2006). This diagnostic assesses whether the study has adequate statistical power to detect clinically meaningful effects in a self-controlled design.

Usage

```
computeMdrrForRateRatio(
  exposedPersonTime,
  unexposedPersonTime,
  exposedEvents,
  unexposedEvents,
  alpha = 0.05,
  power = 0.8
)
```

Arguments

`exposedPersonTime`	Total person-time in exposed window (in days)
`unexposedPersonTime`	Total person-time in unexposed window (in days)
`exposedEvents`	Number of outcome events in exposed window
`unexposedEvents`	Number of outcome events in unexposed window
`alpha`	Significance level (default: 0.05)
`power`	Desired power (default: 0.80)

Details

The MDRR is the minimum incidence rate ratio that can be detected with the given sample size, alpha, and power. This implementation uses the SRL1 method from Musonda (2006), which is more accurate for self-controlled studies than standard binomial approximations.

Value

Numeric value representing the MDRR. Values > 10.0 typically indicate low power.

References

Musonda P, Farrington CP, Whitaker HJ (2006) Samples sizes for self-controlled case series studies, Statistics in Medicine, 15;25(15):2618-31

Examples

```
if (interactive()) {
# Calculate MDRR for a study with 100 exposed person-years and 200 unexposed person-years
mdrr <- computeMdrrForRateRatio(
  exposedPersonTime = 36500,    # 100 person-years in days
  unexposedPersonTime = 73000,  # 200 person-years in days
  exposedEvents = 10,
  unexposedEvents = 15
)
}
```

`createExposureOutcome` *Create exposure-outcome combinations.*

Description

Create exposure-outcome combinations.

Usage

```
createExposureOutcome(exposureId, outcomeId, trueEffectSize = NA)
```

Arguments

`exposureId`	A concept ID indentifying the drug of interest in the exposure table. If multiple strategies for picking the exposure will be tested in the analysis, a named list of numbers can be provided instead. In the analysis, the name of the number to be used can be specified using the `exposureType` parameter in the `createSccAnalysis` function.
`outcomeId`	A concept ID indentifying the outcome of interest in the outcome table. If multiple strategies for picking the outcome will be tested in the analysis, a named list of numbers can be provided instead. In the analysis, the name of the number to be used can be specified using the #' `outcomeType` parameter in the `createSccAnalysis` function.
`trueEffectSize`	Should this be set to 1 this will be considererd a negative control

Details

Create a hypothesis of interest, to be used with the `runSccAnalyses` function.

Value

An object of class `exposureOutcome` specifying an exposure-outcome pair for analysis.

Examples

```
if (interactive()) {
# Create a simple exposure-outcome pair
eo1 <- createExposureOutcome(exposureId = 1124300,
                             outcomeId = 444382)

# Create a negative control (true effect size = 1)
eo2 <- createExposureOutcome(exposureId = 1124300,
                             outcomeId = 444382,
                             trueEffectSize = 1)
}
```

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

Description

Create the results data model tables on a database server.

Usage

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

Arguments

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

Details

Only PostgreSQL and SQLite servers are supported.

Value

Invisibly returns NULL. Creates database tables for results storage as a side effect.

Examples

```
if (interactive()) {
connectionDetails <- DatabaseConnector::createConnectionDetails(
  dbms = "sqlite",
  server = "myResults.db"
)

createResultsDataModel(
  connectionDetails = connectionDetails,
  databaseSchema = "main"
)
}
```

---

createRunSelfControlledCohortArgs *Create a parameter object for the function runSelfControlledCohort*

Description

Create a parameter object for the function runSelfControlledCohort

Usage

```
createRunSelfControlledCohortArgs(
  firstExposureOnly = TRUE,
  firstOutcomeOnly = TRUE,
  minAge = "",
  maxAge = "",
  studyStartDate = "",
  studyEndDate = "",
  addLengthOfExposureExposed = TRUE,
  riskWindowStartExposed = 1,
  riskWindowEndExposed = 30,
  addLengthOfExposureUnexposed = TRUE,
  riskWindowEndUnexposed = -1,
  riskWindowStartUnexposed = -30,
  hasFullTimeAtRisk = FALSE,
  washoutPeriod = 0,
  followupPeriod = 0,
  runDiagnostics = TRUE,
  diagnostics = c("all"),
  diagnosticThresholds = getDefaultDiagnosticThresholds()
)
```

Arguments

`firstExposureOnly`	If TRUE, only use first occurrence of each drug concept idfor each person
`firstOutcomeOnly`	If TRUE, only use first occurrence of each condition conceptid for each person.
`minAge`	Integer for minimum allowable age.
`maxAge`	Integer for maximum allowable age.
`studyStartDate`	Date for minimum allowable data for index exposure. Dateformat is 'yyyymmdd'.
`studyEndDate`	Date for maximum allowable data for index exposure. Dateformat is 'yyyymmdd'.
`addLengthOfExposureExposed`	If TRUE, use the duration from drugEraStart -> drugEraEnd as part of timeAtRisk.
`riskWindowStartExposed`	Integer of days to add to drugEraStart for start oftimeAtRisk (0 to include index date, 1 to start the dayafter).
`riskWindowEndExposed`	Additional window to add to end of exposure period (ifaddLengthOfExposureExposed = TRUE, then add to exposure enddate, else add to exposure start date).
`addLengthOfExposureUnexposed`	If TRUE, use the duration from exposure start -> exposureend as part of timeAtRisk looking back before exposurestart.
`riskWindowEndUnexposed`	Integer of days to add to exposure start for end oftimeAtRisk (0 to include index date, -1 to end the daybefore).
`riskWindowStartUnexposed`	Additional window to add to start of exposure period (ifaddLengthOfExposureUnexposed = TRUE, then add to exposureend date, else add to exposure start date).
`hasFullTimeAtRisk`	If TRUE, restrict to people who have full time-at-riskexposed and unexposed.
`washoutPeriod`	Integer to define required time observed before exposurestart.
`followupPeriod`	Integer to define required time observed after exposurestart. absolute time between treatment and outcome. Note, may add significant computation time on some database engines. If set true in one analysis will default to true for all others.
`runDiagnostics`	If TRUE, run diagnostic tests on the results.
`diagnostics`	Character vector specifying which diagnostics to run. Options: "all", "mdrr", "pre_exposure_gain", "event_dependent", "ease". Default is "all".
`diagnosticThresholds`	Named list of diagnostic thresholds. See getDefaultDiagnosticThresholds() for defaults.

Details

Create an object defining the parameter values.

Value

A parameter object of class `args` for use with `runSelfControlledCohort`.

Examples

```
if (interactive()) {
# Create default parameters
args1 <- createRunSelfControlledCohortArgs()

# Custom risk window settings
args2 <- createRunSelfControlledCohortArgs(
  riskWindowStartExposed = 1,
  riskWindowEndExposed = 60,
  hasFullTimeAtRisk = TRUE
)
}
```

---

createSccAnalysis *Create a SelfControlledCohort analysis specification*

Description

Create a SelfControlledCohort analysis specification

Usage

```
createSccAnalysis(
  analysisId = 1,
  description = "",
  exposureType = NULL,
  outcomeType = NULL,
  runSelfControlledCohortArgs,
  controlType = "outcome",
  runDiagnostics = TRUE,
  diagnostics = c("all"),
  diagnosticThresholds = getDefaultDiagnosticThresholds()
)
```

Arguments

`analysisId`	An integer that will be used later to refer to this specific set of analysis choices.
`description`	A short description of the analysis.

`exposureType` If more than one exposure is provided for each exposureOutcome, this field should be used to select the specific exposure to use in this analysis.

`outcomeType` If more than one outcome is provided for each exposureOutcome, this field should be used to select the specific outcome to use in this analysis.

`runSelfControlledCohortArgs`
An object representing the arguments to be used when calling the `runSelfControlledCohort` function.

`controlType` Character string specifying the type of control. Options are "outcome" or "exposure". Default is "outcome".

`runDiagnostics` Logical indicating whether to run diagnostic tests on the results. Default is TRUE.

`diagnostics` Character vector specifying which diagnostics to run. Options: "all", "mdrr", "pre_exposure_gain", "event_dependent", "ease". Default is "all".

`diagnosticThresholds`
Named list of diagnostic thresholds. See getDefaultDiagnosticThresholds() for defaults.

Details

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

Value

An object of class `sccAnalysis` containing the analysis specifications.

Examples

```
if (interactive()) {
# Create SCC analysis with default risk window settings
sccArgs <- createRunSelfControlledCohortArgs(riskWindowStartExposed = 1,
                                             riskWindowEndExposed = 30)

analysis1 <- createSccAnalysis(analysisId = 1,
                               description = "30-day exposed risk window",
                               runSelfControlledCohortArgs = sccArgs)
}
```

---

createSelfControlledCohortModuleSpecifications

Create Self-Controlled Cohort Module Specifications

Description

Creates a specifications object for the Self-Controlled Cohort module to be used within the OHDSI Strategus framework.

Usage

```
createSelfControlledCohortModuleSpecifications(
  analysisSettings,
  exposureOutcomeList,
  computeThreads = parallel::detectCores() - 1
)
```

Arguments

analysisSettings	A list of analysis settings containing the analysis configuration and parameters for self-controlled cohort analyses.
exposureOutcomeList	A list of objects of type exposureOutcome as created using the createExposureOutcome function. Each object defines an exposure-outcome pair with exposureId, outcomeId, and optionally trueEffectSize (set to 1 for negative controls).
computeThreads	Integer specifying the number of threads to use for parallel computation. Default is the number of available cores minus 1.

Value

A list object of class 'SelfControlledCohortModuleSpecifications' and 'ModuleSpecifications' containing the module name, version, repository information, and analysis settings.

Examples

```
if (interactive()) {
eo1 <- createExposureOutcome(exposureId = 1124300, outcomeId = 444382)
analysis1 <- createSccAnalysis(analysisId = 1,
                               description = "Main",
                        runSelfControlledCohortArgs = createRunSelfControlledCohortArgs())

moduleSpec <- createSelfControlledCohortModuleSpecifications(
  analysisSettings = list(analysis1),
  exposureOutcomeList = list(eo1)
)
}
```

---

execute *Execute Self-Controlled Cohort Analyses for Strategus*

Description

Executes Self-Controlled Cohort analyses within the OHDSI Strategus framework using the provided analysis specifications and settings.

Usage

```
execute(
  connectionDetails,
  executionSettings,
  analysisSpecifications,
  databaseId,
  exportFolder
)
```

Arguments

connectionDetails	An object containing the details required to connect to the database.
executionSettings	A list of settings for execution, including database schemas and cohort table names.
analysisSpecifications	A list containing analysis settings, exposure-outcome pairs, and compute thread count.
databaseId	A string identifier for the target database.
exportFolder	Path to the folder where results and exports will be written.

Details

This function validates the input specifications, extracts unique exposure and outcome cohort IDs, identifies negative control pairs, and iterates over each analysis setting to run the Self-Controlled Cohort analysis. Results are exported to the specified folder. Diagnostic settings are handled per-analysis, and warnings are issued if no negative controls are found.

Value

No return value. Results are written to the specified export folder as a side effect.

Examples

```
if (interactive()) {
eo1 <- createExposureOutcome(exposureId = 1124300, outcomeId = 444382)
analysis1 <- createSccAnalysis(analysisId = 1,
                               description = "Main",
                      runSelfControlledCohortArgs = createRunSelfControlledCohortArgs())

moduleSpec <- createSelfControlledCohortModuleSpecifications(
  analysisSettings = list(analysis1),
  exposureOutcomeList = list(eo1)
)
execute(
  connectionDetails = connectionDetails,
  executionSettings = list(
    workDatabaseSchema = "main",
```

```
    cohortTableNames = CohortGenerator::getCohortTableNames(cohortTable = "cohort"),
    cdmDatabaseSchema = "main"
  ),
  analysisSpecifications = moduleSpec,
  databaseId = "MyDatabase",
  exportFolder = tempdir()
)
}
```

`getDataMigrator` *Get database migrations instance*

Description

Returns ResultModelManager DataMigrationsManager instance.

Usage

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

Arguments

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

Value

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

See Also

[ResultModelManager::DataMigrationManager] which this function is a utility for.

Examples

```
if (interactive()) {
migrator <- getDataMigrator(
  connectionDetails = connectionDetails,
  databaseSchema = "results"
)
}
```

`getDefaultDiagnosticThresholds` *Get default diagnostic thresholds*

Description

Returns default thresholds for diagnostic tests following SCCS standards. Version 2.1.0+ uses revised diagnostics that align with SelfControlledCaseSeries package.

Usage

```
getDefaultDiagnosticThresholds()
```

Details

Thresholds:

- mdrrMaxAcceptable - Maximum acceptable MDRR (default: 10.0). Higher values indicate low power.
- maxPreExposureProportion - Maximum proportion of persons with pre-exposure outcomes (default: 0.05)
- preExposurePThreshold - Significance threshold for pre-exposure gain test (default: 0.05)
- maxEventDependentCensoring - Maximum proportion censored within 30 days of outcome (default: 0.25)
- minEventsPerWindow - Minimum events required in each window (default: 3)
- easeMaxAcceptable - Maximum acceptable EASE (default: 0.25). Requires negative controls.

Value

A list of diagnostic thresholds

Examples

```
if (interactive()) {
# Get default thresholds
   thresholds <- getDefaultDiagnosticThresholds()

   # Modify specific thresholds
   customThresholds <- getDefaultDiagnosticThresholds()
   customThresholds$mdrrMaxAcceptable <- 5.0
}
```

getDefaultExportManager *Get Default export manager*

Description

Returns the default export manager class for writing csv file results

Usage

```
getDefaultExportManager(resultExportPath, databaseId)
```

Arguments

resultExportPath	Folder where result files are exported
databaseId	Unique identifier for database - required

Value

An instance of `ResultModelManager::ResultExportManager` configured for SelfControlledCohort results.

Examples

```
if (interactive()) {
exportManager <- getDefaultExportManager(
  resultExportPath = tempdir(),
  databaseId = "CCAE"
)
}
```

getDiagnosticsSummary *Get diagnostics summary*

Description

Returns a summary of diagnostic results for each target-outcome pair, including blinding status.

Usage

```
getDiagnosticsSummary(diagnosticResults)
```

Arguments

diagnosticResults
: Data frame of diagnostic results as returned by runSccDiagnostics

Value

A data frame with blinding status per target-outcome pair

Examples

```
if (interactive()) {
connectionDetails <- Eunomia::getEunomiaConnectionDetails()
connection <- DatabaseConnector::connect(connectionDetails)

diagnostics <- runSccDiagnostics(
  connection = connection,
  cdmDatabaseSchema = "main",
  resultsTable = "#scc_results",
  riskWindowsTable = "#risk_windows",
  analysisId = 1,
  databaseId = "Eunomia",
  estimates = resultsData,
  diagnostics = "all",
  thresholds = getDefaultDiagnosticThresholds()
)

DatabaseConnector::disconnect(connection)
summary <- getDiagnosticsSummary(diagnostics)
}
```

---

getModuleInfo *Get module information*

Description

Get module information

Usage

```
getModuleInfo()
```

Value

A list with module metadata

Examples

```
if (interactive())
moduleInfo <- getModuleInfo()
```

`getResultsDataModelSpecifications` *Get specifications for CohortMethod results data model*

Description

Get specifications for CohortMethod results data model

Usage

```
getResultsDataModelSpecifications()
```

Value

A data frame object with specifications

Examples

```
if (interactive()) {
specs <- getResultsDataModelSpecifications()
}
```

`getResultsFolders` *Get results folders for an analysis specification*

Description

Get results folders for an analysis specification

Usage

```
getResultsFolders(analysisSpecification, exportFolder)
```

Arguments

`analysisSpecification`	An analysis specification object containing analysis settings and exposure-outcome pairs.
`exportFolder`	The base folder where results are exported. Individual analysis results will be in subfolders named A_analysisId.

Value

A character vector of paths to results folders for each analysis setting.

Examples

```
if (interactive())
resultsFolders <- getResultsFolders(analysisSpec, tempdir())
```

`getSccRiskWindowStats` *Get Self-Controlled Cohort Risk Window Statistics*

Description

Compute statistics from risk windows.

Usage

```
getSccRiskWindowStats(
  connection,
  outcomeDatabaseSchema,
  databaseId,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  outcomeIds = NULL,
  outcomeTable = "condition_era",
  firstOutcomeOnly = TRUE,
  resultsDatabaseSchema = NULL,
  riskWindowsTable = "#risk_windows",
  resultExportPath = "scc_result",
  analysisId = 1,
 resultExportManager = ResultModelManager::createResultExportManager(tableSpecification
  = getResultsDataModelSpecifications(), exportDir = resultExportPath, databaseId =
    databaseId)
)
```

Arguments

`connection`	DatabaseConnector connection instance
`outcomeDatabaseSchema`	The name of the database schema that is the location where the data used to define the outcome cohorts is available. If exposureTable = CONDITION_ERA, exposureDatabaseSchema is not used by assumed to be cdmSchema. Requires read permissions to this database.
`databaseId`	Unique identifier for database - required

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

outcomeIds
: The condition_concept_ids or cohort_definition_ids of the outcomes of interest. If empty, all the outcomes in the outcome table will be included.

outcomeTable
: The tablename that contains the outcome cohorts. If outcomeTable <> CONDITION_OCCURRENCE, then expectation is outcomeTable has format of COHORT table: COHORT_DEFINITION_ID, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.

firstOutcomeOnly
: If TRUE, only use first occurrence of each condition concept id for each person.

resultsDatabaseSchema
: Schema to oputput results to. Ignored if resultsTable and riskWindowsTable are temporary.

riskWindowsTable
: String: optionally store the risk windows in a (non-temporary) table.

resultExportPath
: Folder where result files are exported

analysisId
: An integer unique to this analysis

resultExportManager
: ResultModelManager::ResultExportManager instance - customize this to implement an alternative mechanism for exporting results

Details

Requires a risk window table to be created first with 'runSccRiskWindows'

Value

list containing data frames: treatmentTimeDistribution, timeToOutcomeDistribution, timeToOutcomeDistributionExposed, timeToOutcomeDistributionUnexposed

Examples

```
if (interactive()) {
# First, create the risk windows table
connectionDetails <- Eunomia::getEunomiaConnectionDetails()
connection <- DatabaseConnector::connect(connectionDetails)
riskWindowsTable <- "computed_risk_windows"
runSccRiskWindows(connection,
                  cdmDatabaseSchema = "main",
                  exposureIds = c(1102527, 1125315),
                  resultsDatabaseSchema = "main",
                  riskWindowsTable = riskWindowsTable,
                  exposureTable = "drug_era")
# Get stats based on outcomes of interest
tarStats <- getSccRiskWindowStats(connection,
                                  outcomeDatabaseSchema = "main",
```

```
                                    databaseId = "Eunomia",
                                    resultsDatabaseSchema = "main",
                                    riskWindowsTable = riskWindowsTable,
                                    outcomeTable = "condition_era",
                                    outcomeIds = 192671)
DatabaseConnector::disconnect(connection)
}
```

`loadExposureOutcomeList` *Load a list of exposureOutcome from file*

Description

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

Usage

```
loadExposureOutcomeList(file)
```

Arguments

`file`	The name of the file

Value

A list of objects of type `exposureOutcome`.

Examples

```
tempFile <- file.path(tempdir(), "exposureOutcomes.json")
saveExposureOutcomeList(list(createExposureOutcome(1124300, 444382)), tempFile)
eoList <- loadExposureOutcomeList(tempFile)
```

`loadSccAnalysisList` *Load a list of sccAnalysis from file*

Description

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

Usage

```
loadSccAnalysisList(file)
```

Arguments

file	The name of the file

Value

A list of objects of type sccAnalysis.

Examples

```
if (interactive()) {
tempFile <- file.path(tempdir(), "analyses.json")
saveSccAnalysisList(list(createSccAnalysis(analysisId = 1,
  runSelfControlledCohortArgs = createRunSelfControlledCohortArgs())), tempFile)
analysisList <- loadSccAnalysisList(tempFile)
}
```

---

migrateDataModel *Migrate Data model*

Description

Migrate data from current state to next state

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

Usage

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

Arguments

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

Value

Invisibly returns NULL. Migrates the database schema as a side effect.

Examples

```
if (interactive()) {
migrateDataModel(
  connectionDetails = connectionDetails,
  databaseSchema = "results"
)
}
```

`runSccAnalyses` *Run a list of analyses*

Description

Run a list of analyses

Usage

```
runSccAnalyses(
  connectionDetails,
  cdmDatabaseSchema,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  exposureDatabaseSchema = cdmDatabaseSchema,
  exposureTable = "drug_era",
  outcomeDatabaseSchema = cdmDatabaseSchema,
  outcomeTable = "condition_occurrence",
  resultsFolder = "./SelfControlledCohortOutput",
  sccAnalysisList,
  exposureOutcomeList,
  databaseId,
  controlType = "outcome",
  analysisThreads = 1,
  computeThreads = 1
)
```

Arguments

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

`cdmDatabaseSchema`
: Name of database schema that contains the OMOP CDM and vocabulary.

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

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

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

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

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

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

`sccAnalysisList`
: A list of objects of type `sccAnalysis` as created using the `createSccAnalysis` function.

`exposureOutcomeList`
: A list of objects of type `exposureOutcome` as created using the `createExposureOutcome` function.

`databaseId`
: Unique identifier for database - required

`controlType`
: Calibrate effect estimates with outcome (default) or exposure controls

`analysisThreads`
: The number of parallel threads to use to execute the analyses.

`computeThreads`
: Number of parallel threads for computing IRRs with exact confidence intervals.

Details

Run a list of analyses for the drug-comparator-outcomes of interest. This function will run all specified analyses against all hypotheses of interest, meaning that the total number of outcome models is 'length(cmAnalysisList) * length(drugComparatorOutcomesList)'.

Value

Invisibly returns a data frame containing a reference table for all exposure-outcome-analysis combinations executed.

Examples

```
if (interactive()) {
connectionDetails <- Eunomia::getEunomiaConnectionDetails()

eo1 <- createExposureOutcome(exposureId = 1124300, outcomeId = 444382)
```

```
analysis1 <- createSccAnalysis(analysisId = 1,
                               description = "Main analysis",
                       runSelfControlledCohortArgs = createRunSelfControlledCohortArgs())

results <- runSccAnalyses(
  connectionDetails = connectionDetails,
  cdmDatabaseSchema = "main",
  exposureOutcomeList = list(eo1),
  sccAnalysisList = list(analysis1),
  databaseId = "Eunomia",
  resultsFolder = tempdir()
)
}
```

`runSccDiagnostics` *Run Self-Controlled Cohort Diagnostics*

Description

Runs a suite of diagnostic tests to assess the validity of SCC analysis assumptions. Version 1.6.0+ implements diagnostics aligned with SelfControlledCaseSeries package standards.

Usage

```
runSccDiagnostics(
  connection,
  cdmDatabaseSchema,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  resultsTable,
  riskWindowsTable,
  outcomeTable = "condition_era",
  outcomeDatabaseSchema = cdmDatabaseSchema,
  analysisId,
  databaseId,
  estimates = NULL,
  controlType = "outcome",
  diagnostics = c("all"),
  thresholds = getDefaultDiagnosticThresholds(),
 computeThreads = getOption("strategus.SelfControlledCohort.computeThreads", default =
    parallel::detectCores() - 1),
  resultExportManager
)
```

Arguments

`connection`	DatabaseConnector connection instance
`cdmDatabaseSchema`	Name of database schema that contains OMOP CDM

`tempEmulationSchema`
: Schema for temp table emulation (Oracle, Impala)

`resultsTable`
: Name of the results table (contains counts)

`riskWindowsTable`
: Name of the risk windows table

`outcomeTable`
: Name of outcome table (e.g., "condition_era", "cohort")

`outcomeDatabaseSchema`
: Schema containing outcome table

`analysisId`
: Analysis identifier

`databaseId`
: Database identifier for results export

`estimates`
: Data frame of raw SCC results (including rr and se_log_rr)

`controlType`
: Calibrate effect estimates with outcome (default) or exposure controls. Determines whether EASE is grouped by target or outcome.

`diagnostics`
: Character vector of diagnostics to run. Options: "all", "mdrr", "pre_exposure_gain", "event_dependent", "ease"

`thresholds`
: Named list of diagnostic thresholds (see getDefaultDiagnosticThresholds)

`computeThreads`
: Number of parallel threads to use for CPU-bound diagnostic computations (e.g., MDRR and EASE). Default is the number of available cores minus 1.

`resultExportManager`
: ResultModelManager::ResultExportManager instance

Value

Invisible data frame of diagnostic results

Examples

```
if (interactive()) {
  connectionDetails <- Eunomia::getEunomiaConnectionDetails()
  connection <- DatabaseConnector::connect(connectionDetails)

  diagnostics <- runSccDiagnostics(
    connection = connection,
    cdmDatabaseSchema = "main",
    resultsTable = "#scc_results",
    riskWindowsTable = "#risk_windows",
    analysisId = 1,
    databaseId = "Eunomia",
    estimates = resultsData,
    diagnostics = "all",
    thresholds = getDefaultDiagnosticThresholds()
  )

  DatabaseConnector::disconnect(connection)
}
```

`runSccRiskWindows` *Run Self-Controlled Cohort Risk Windows*

Description

Compute time at risk exposed and time at risk unexposed for risk window parameters. See 'getSccRiskWindowStats' for example usage.

Usage

```
runSccRiskWindows(
  connection,
  cdmDatabaseSchema,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  exposureIds = NULL,
  exposureDatabaseSchema = cdmDatabaseSchema,
  exposureTable = "drug_era",
  firstExposureOnly = TRUE,
  minAge = "",
  maxAge = "",
  studyStartDate = "",
  studyEndDate = "",
  addLengthOfExposureExposed = TRUE,
  riskWindowStartExposed = 1,
  riskWindowEndExposed = 30,
  addLengthOfExposureUnexposed = TRUE,
  riskWindowEndUnexposed = -1,
  riskWindowStartUnexposed = -30,
  hasFullTimeAtRisk = FALSE,
  washoutPeriod = 0,
  followupPeriod = 0,
  riskWindowsTable = "#risk_windows",
  keepResultsTables = TRUE,
  analysisId = 1,
  resultsDatabaseSchema = NULL
)
```

Arguments

`connection`	DatabaseConnector connection instance
`cdmDatabaseSchema`	Name of database schema that contains the OMOP CDM and vocabulary.
`tempEmulationSchema`	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.

`exposureIds` A vector containing the drug_concept_ids or cohort_definition_ids of the exposures of interest. If empty, all exposures in the exposure table will be included.

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

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

`firstExposureOnly`
If TRUE, only use first occurrence of each drug concept id for each person

`minAge` Integer for minimum allowable age.

`maxAge` Integer for maximum allowable age.

`studyStartDate` Date for minimum allowable data for index exposure. Date format is 'yyyymmdd'.

`studyEndDate` Date for maximum allowable data for index exposure. Date format is 'yyyymmdd'.

`addLengthOfExposureExposed`
If TRUE, use the duration from drugEraStart -> drugEraEnd as part of timeAtRisk.

`riskWindowStartExposed`
Integer of days to add to drugEraStart for start of timeAtRisk (0 to include index date, 1 to start the day after).

`riskWindowEndExposed`
Additional window to add to end of exposure period (if addLengthOfExposureExposed = TRUE, then add to exposure end date, else add to exposure start date).

`addLengthOfExposureUnexposed`
If TRUE, use the duration from exposure start -> exposure end as part of timeAtRisk looking back before exposure start.

`riskWindowEndUnexposed`
Integer of days to add to exposure start for end of timeAtRisk (0 to include index date, -1 to end the day before).

`riskWindowStartUnexposed`
Additional window to add to start of exposure period (if addLengthOfExposureUnexposed = TRUE, then add to exposure end date, else add to exposure start date).

`hasFullTimeAtRisk`
If TRUE, restrict to people who have full time-at-risk exposed and unexposed.

`washoutPeriod` Integer to define required time observed before exposure start.

`followupPeriod` Integer to define required time observed after exposure start.

`riskWindowsTable`
String: optionally store the risk windows in a (non-temporary) table.

keepResultsTables
: Keep the results tables in place if they exist. This allows the data set to be added to with aditional targets and outcomes. (ignored if temporary tables are used, default)

analysisId
: An integer unique to this analysis

resultsDatabaseSchema
: Schema to oputput results to. Ignored if resultsTable and riskWindowsTable are temporary.

Value

Invisibly returns NULL. Creates the risk windows table in the database as a side effect.

Examples

```
if (interactive()) {
connectionDetails <- Eunomia::getEunomiaConnectionDetails()
connection <- DatabaseConnector::connect(connectionDetails)

runSccRiskWindows(
  connection = connection,
  cdmDatabaseSchema = "main",
  exposureIds = c(1124300),
  riskWindowStartExposed = 1,
  riskWindowEndExposed = 30,
  riskWindowsTable = "#risk_windows"
)

DatabaseConnector::disconnect(connection)
}
```

---

runSelfControlledCohort

Run self-controlled cohort

Description

runSelfControlledCohort generates population-level estimation by comparing exposed and unexposed time among exposed cohort.

Usage

```
runSelfControlledCohort(
  connectionDetails = NULL,
  cdmDatabaseSchema,
  connection = NULL,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
```

```
  exposureIds = NULL,
  outcomeIds = NULL,
  negativeControlPairs = NULL,
  exposureOutcomeList = NULL,
  controlType = "outcome",
  exposureDatabaseSchema = cdmDatabaseSchema,
  exposureTable = "drug_era",
  outcomeDatabaseSchema = cdmDatabaseSchema,
  outcomeTable = "condition_era",
  firstExposureOnly = TRUE,
  firstOutcomeOnly = TRUE,
  minAge = "",
  maxAge = "",
  studyStartDate = "",
  studyEndDate = "",
  addLengthOfExposureExposed = TRUE,
  riskWindowStartExposed = 1,
  riskWindowEndExposed = 30,
  addLengthOfExposureUnexposed = TRUE,
  riskWindowEndUnexposed = -1,
  riskWindowStartUnexposed = -30,
  hasFullTimeAtRisk = FALSE,
  washoutPeriod = 0,
  followupPeriod = 0,
 computeThreads = getOption("strategus.SelfControlledCohort.computeThreads", default =
   parallel::detectCores() - 1),
  riskWindowsTable = "#risk_windows",
  resultsTable = "#results",
  keepResultsTables = TRUE,
  extractResults = TRUE,
  resultsDatabaseSchema = NULL,
  resultExportPath = "scc_result",
  databaseId,
  analysisId = 1,
  analysisDescription = paste("SCC analysis", analysisId),
  resultExportManager = getDefaultExportManager(resultExportPath, databaseId),
  runDiagnostics = TRUE,
  diagnostics = c("all"),
  diagnosticThresholds = getDefaultDiagnosticThresholds()
)
```

Arguments

connectionDetails	An R object of type connectionDetails created using the function createConnectionDetails in the DatabaseConnector package.
cdmDatabaseSchema	Name of database schema that contains the OMOP CDM and vocabulary.
connection	DatabaseConnector connection instance

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

exposureIds
: A vector containing the drug_concept_ids or cohort_definition_ids of the exposures of interest. If empty, all exposures in the exposure table will be included.

outcomeIds
: The condition_concept_ids or cohort_definition_ids of the outcomes of interest. If empty, all the outcomes in the outcome table will be included.

negativeControlPairs
: A list of vectors for pairs of negative control

exposureOutcomeList
: A list of objects of type `exposureOutcome` as created using `createExposureOutcome`, defining the full set of exposure-outcome pairs for the study. Used to populate the `scc_outcome_exposure` results table.

controlType
: Calibrate effect estimates with outcome (default) or exposure controls

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

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

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

outcomeTable
: The tablename that contains the outcome cohorts. If outcomeTable <> CONDITION_OCCURRENCE, then expectation is outcomeTable has format of COHORT table: COHORT_DEFINITION_ID, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.

firstExposureOnly
: If TRUE, only use first occurrence of each drug concept id for each person

firstOutcomeOnly
: If TRUE, only use first occurrence of each condition concept id for each person.

minAge
: Integer for minimum allowable age.

maxAge
: Integer for maximum allowable age.

studyStartDate
: Date for minimum allowable data for index exposure. Date format is 'yyyymmdd'.

studyEndDate
: Date for maximum allowable data for index exposure. Date format is 'yyyymmdd'.

addLengthOfExposureExposed
: If TRUE, use the duration from drugEraStart -> drugEraEnd as part of timeAtRisk.

`riskWindowStartExposed`
: Integer of days to add to drugEraStart for start of timeAtRisk (0 to include index date, 1 to start the day after).

`riskWindowEndExposed`
: Additional window to add to end of exposure period (if addLengthOfExposure-Exposed = TRUE, then add to exposure end date, else add to exposure start date).

`addLengthOfExposureUnexposed`
: If TRUE, use the duration from exposure start -> exposure end as part of timeAtRisk looking back before exposure start.

`riskWindowEndUnexposed`
: Integer of days to add to exposure start for end of timeAtRisk (0 to include index date, -1 to end the day before).

`riskWindowStartUnexposed`
: Additional window to add to start of exposure period (if addLengthOfExposure-Unexposed = TRUE, then add to exposure end date, else add to exposure start date).

`hasFullTimeAtRisk`
: If TRUE, restrict to people who have full time-at-risk exposed and unexposed.

`washoutPeriod` Integer to define required time observed before exposure start.

`followupPeriod` Integer to define required time observed after exposure start.

`computeThreads` Number of parallel threads for computing IRRs with exact confidence intervals.

`riskWindowsTable`
: String: optionally store the risk windows in a (non-temporary) table.

`resultsTable` String: optionally store the summary results (number exposed/ unexposed patients per outcome-exposure pair) in a (non-temporary) table. Note that this table does not store the rate ratios, only the values required to calculate rate ratios.

`keepResultsTables`
: Keep the results tables in place if they exist. This allows the data set to be added to with aditional targets and outcomes. (ignored if temporary tables are used, default)

`extractResults` Export results to disk. In the case of very large exposure/outcome set pairs it is often more ideal to create a permanent results table with the @resultsTable parameter and extract and calibrate small subsets on demand. Performing calibration across the full set of outcome/exposure pairs can take a significant amount of time. If set to false, no relative risk ratios will be produced.

`resultsDatabaseSchema`
: Schema to oputput results to. Ignored if resultsTable and riskWindowsTable are temporary.

`resultExportPath`
: Folder where result files are exported

`databaseId` Unique identifier for database - required

`analysisId` An integer unique to this analysis

`analysisDescription`
: A string description of the analysis (optional)

`resultExportManager`
: ResultModelManager::ResultExportManager instance - customize this to implement an alternative mechanism for exporting results

`runDiagnostics`
: If TRUE, run diagnostic tests on the results

`diagnostics`
: Character vector specifying which diagnostics to run. Options: "all", "mdrr", "pre_exposure_gain", "event_dependent", "ease". Default is "all".

`diagnosticThresholds`
: Named list of diagnostic thresholds. See getDefaultDiagnosticThresholds()

Details

Population-level estimation method that estimates incidence rate comparison of exposed/unexposed time within an exposed cohort. If multiple exposureIds and outcomeIds are provided, estimates will be generated for every combination of exposure and outcome.

Value

An object of type `sccResults` containing the results of the analysis.

References

Ryan PB, Schuemie MJ, Madigan D.Empirical performance of a self-controlled cohort method: lessons for developing a risk identification and analysis system. Drug Safety 36 Suppl1:S95-106, 2013

Examples

```
if (interactive()) {
# Use Eunomia synthetic database for demonstration
connectionDetails <- Eunomia::getEunomiaConnectionDetails()

result <- runSelfControlledCohort(
  connectionDetails = connectionDetails,
  cdmDatabaseSchema = "main",
  exposureIds = c(1124300),
  outcomeIds = 444382,
  outcomeTable = "condition_era",
  databaseId = "Eunomia"
)
}
```

---

saveExposureOutcomeList

Save a list of exposureOutcome to file

Description

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

Usage

```
saveExposureOutcomeList(exposureOutcomeList, file)
```

Arguments

exposureOutcomeList	The exposureOutcome list to be written to file
file	The name of the file where the results will be written

Value

Invisibly returns NULL. Saves the exposure-outcome list to file as a side effect.

Examples

```
if (interactive()) {
eo1 <- createExposureOutcome(exposureId = 1124300, outcomeId = 444382)
saveExposureOutcomeList(list(eo1), file.path(tempdir(), "exposureOutcomes.json"))
}
```

---

saveSccAnalysisList *Save a list of sccAnalysis to file*

Description

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

Usage

```
saveSccAnalysisList(sccAnalysisList, file)
```

Arguments

sccAnalysisList	The sccAnalysis list to be written to file
file	The name of the file where the results will be written

Value

Invisibly returns NULL. Saves the analysis list to file as a side effect.

Examples

```
if (interactive()) {
analysis1 <- createSccAnalysis(analysisId = 1,
                               description = "30-day risk window",
                        runSelfControlledCohortArgs = createRunSelfControlledCohortArgs())
saveSccAnalysisList(list(analysis1), file.path(tempdir(), "analyses.json"))
}
```

`uploadResults` *Upload results to the database server.*

Description

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

Usage

```
uploadResults(
  connectionDetails,
  schema,
  resultsFolder,
  forceOverWriteOfSpecifications = FALSE,
  purgeSiteDataBeforeUploading = FALSE,
  tablePrefix = "",
  ...
)
```

Arguments

`connectionDetails`	An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package.
`schema`	The schema on the server where the tables have been created.
`resultsFolder`	Path to result files
`forceOverWriteOfSpecifications`	If TRUE, specifications of the phenotypes, cohort definitions, and analysis will be overwritten if they already exist on the database. Only use this if these specifications have changed since the last upload.
`purgeSiteDataBeforeUploading`	If TRUE, before inserting data for a specific databaseId all the data for that site will be dropped. This assumes the input zip file contains the full data for that data site.
`tablePrefix`	(Optional) string to insert before table names for database table names
`...`	See ResultModelManager::uploadResults

Value

Invisibly returns NULL. Uploads results to the database as a side effect.

Examples

```
if (interactive()) {
connectionDetails <- DatabaseConnector::createDonnectionDetails(
  dbms = "sqlite",
  server = "myResults.db"
)
uploadResults(
  connectionDetails = connectionDetails,
  schema = "main",
  resultsFolder = tempdir()
)
}
```

Index

computeEase, 3
computeMdrrForRateRatio, 4
createConnectionDetails, *34*
createExposureOutcome, 5, *11*, *23*
createResultsDataModel, 6, *34*
createRunSelfControlledCohortArgs, 7
createSccAnalysis, *5*, 9, *23*
createSelfControlledCohortModuleSpecifications, 10

execute, 11

getDataMigrator, 13
getDefaultDiagnosticThresholds, 14
getDefaultExportManager, 15
getDiagnosticsSummary, 15
getModuleInfo, 16
getResultsDataModelSpecifications, 17
getResultsFolders, 17
getSccRiskWindowStats, 18

loadExposureOutcomeList, 20
loadSccAnalysisList, 20

migrateDataModel, 21

runSccAnalyses, *5*, *10*, 22
runSccDiagnostics, 24
runSccRiskWindows, 26
runSelfControlledCohort, *9, 10*, 28

saveExposureOutcomeList, 32
saveSccAnalysisList, 33

uploadResults, 34