Package ‘SelfControlledCaseSeries’

May 8, 2026

Type Package

Title Self-Controlled Case Series

Version 6.1.5

Date 2026-05-07

Maintainer Martijn Schuemie <`schuemie@ohdsi.org`>

Description Execute the self-controlled case series (SCCS) design using observational data in the OMOP Common Data Model. Extracts all necessary data from the database and transforms it to the format required for SCCS. Age and season can be modeled using splines assuming constant hazard within calendar months. Event-dependent censoring of the observation period can be corrected for. Many exposures can be included at once (MSCCS), with regularization on all coefficients except for the exposure of interest. Includes diagnostics for all major assumptions of the SCCS.

VignetteBuilder knitr

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

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

Depends R (>= 4.1.0), Cyclops (>= 3.6.0), DatabaseConnector (>= 6.0.0), Andromeda (>= 1.0.0)

Imports SqlRender (>= 1.16.0), dplyr (>= 1.0.0), Rcpp (>= 1.0.9), ParallelLogger (>= 3.4.0), EmpiricalCalibration, splines, ggplot2 (>= 3.4.0), methods, utils, checkmate, readr, ResultModelManager, jsonlite, digest, R6

Suggests zip, testthat, knitr, rmarkdown, Eunomia, withr

License Apache License 2.0

LinkingTo Rcpp

NeedsCompilation yes

RoxygenNote 7.3.3

Encoding UTF-8

Author Martijn Schuemie [aut, cre],
Patrick Ryan [aut],
Trevor Shaddox [aut],
Marc Suchard [aut]

Repository CRAN

Date/Publication 2026-05-08 11:50:02 UTC

Contents

`checkEventExposureIndependenceAssumption`

Check diagnostic for event-dependent exposure

Description

This diagnostic tests whether there is a dependency between the event and subsequent exposures. This requires you have indicated one of the era covariates to be a pre-exposure window. This function simply checks whether the confidence interval for the effect estimate of that pre- exposure window overlaps with the `nullBounds`.

To designate an era covariate to be the pre-exposure window, set `preExposure = TRUE` when calling `createEraCovariateSettings()`. Note that, by default, `preExposure` will be `TRUE` if `start` is smaller than 0.

Usage

```
checkEventExposureIndependenceAssumption(sccsModel, nullBounds = c(0.8, 1.25))
```

Arguments

`sccsModel`	A fitted SCCS model as created using `fitSccsModel()`.
`nullBounds`	The bounds for the null hypothesis on the incidence rate ratio scale.

Value

A tibble with one row per pre-exposure window and four columns: `ratio` indicates the estimates incidence rate ratio for the pre-exposure window. `lb` and `ub` represent the upper and lower bounds of the 95 percent confidence interval, and `pass` is `TRUE` if the confidence interval intersects the null bounds.

`checkEventObservationIndependenceAssumption` *Check diagnostic for event-dependent observation end*

Description

This diagnostic tests whether there is a dependency between the event and the end of observation. It does so by adding a probe window at the end of observation, and checking whether the rate of the outcome is elevated (or decreased) during this window.

The end of observation probe window will automatically be added to the model by the `createSccsIntervalData()` function, unless the `endOfObservationEraLength` argument is set to 0. This function extracts the estimate for that window from the model, and compares it to the `nullBounds`.#'

Usage

```
checkEventObservationIndependenceAssumption(sccsModel, nullBounds = c(0.5, 2))
```

Arguments

`sccsModel`	A fitted SCCS model as created using `fitSccsModel()`.
`nullBounds`	The bounds for the null hypothesis on the incidence rate ratio scale.

Value

A tibble with one row and four columns: `ratio` indicates the estimates incidence rate ratio for the probe at the end of observation. `lb` and `ub` represent the upper and lower bounds of the 95 percent confidence interval, and `pass` is `TRUE` if the confidence interval intersects the null bounds.

`checkRareOutcomeAssumption` *Check if rare outcome assumption is violated*

Description

Check if rare outcome assumption is violated

Usage

```
checkRareOutcomeAssumption(
  studyPopulation,
  firstOutcomeOnly = NULL,
  maxPrevalence = 0.1
)
```

Arguments

`studyPopulation`	An object created using the `createStudyPopulation()` function.
`firstOutcomeOnly`	Was the analysis restricted to the first outcome only? If left at NULL, will be determined by whether `firstOutcomeOnly` was set to `TRUE` when calling `createStudyPopulation()` or whether each person only had one outcome when pulling the data from the server.
`maxPrevalence`	The maximum allowed prevalence (proportion of people with the outcome) allowed when restricting to first outcome only.

Details

Most SCCS analyses restrict to the first outcome occurrence per person to avoid violating the assumption that subsequent occurrences are independent. This is fine, as long as the outcome is rare. According to Farrington et al., the magnitude of the bias from violating this assumption is 0.5p, where p is the prevalence. By default we set the threshold for p at 0.1, corresponding to at most 5 percent bias.

The prevalence was computed in the `getDbSccsData()` function, within the population defined by the `observation_period` table, and restricted to the study period(s) and nesting cohort if used.

Value

A tibble with one row and three columns: `outcomeProportion` indicates the proportion of people having the outcome at least once. `firstOutcomeOnly` indicated whether the analysis was restricted to the first outcome only. `rare` is TRUE if the rare outcome assumption is met, or the analysis was not restricted to the first outcome.

References

Farrington P, Whitaker H, Ghebremichael-Weldeselassie Y, Self-Controlled Case Series Studies: A Modelling Guide with R, CRC Press, 2018

`checkTimeStabilityAssumption`

Check stability of outcome rate over time

Description

Check stability of outcome rate over time

Usage

```
checkTimeStabilityAssumption(
  studyPopulation,
  sccsModel = NULL,
  maxRatio = 1.1,
  alpha = 0.05
)
```

Arguments

`studyPopulation`	An object created using the `createStudyPopulation()` function.
`sccsModel`	Optional: A fitted SCCS model as created using `fitSccsModel()`. If the model contains splines for seasonality and or calendar time these will be adjusted for before computing stability.
`maxRatio`	The maximum global ratio between the observed and expected count.
`alpha`	The alpha (type 1 error) used to test for stability.

Details

Computes for each month the observed and expected count, and computes the (weighted) mean ratio between the two. If splines are used to adjust for seasonality and/or calendar time, these adjustments are taken into consideration when considering the expected count. A one-sided p-value is computed against the null hypothesis that the ratio is smaller than `maxRatio`. If this p-value exceeds the specified alpha value, the series is considered stable.

Value

A tibble with one row and three columns: `ratio` indicates the estimated mean ratio between observed and expected. `p` is the p-value against the null-hypothesis that the ratio is smaller than `maxRatio`, and `pass` is `TRUE` if `p` is greater than `alpha`.

`computeMdrr` *Compute the minimum detectable relative risk*

Description

Compute the minimum detectable relative risk

Usage

```
computeMdrr(
  object,
  exposureCovariateId,
  alpha = 0.05,
  power = 0.8,
  twoSided = TRUE,
  method = "SRL1"
)
```

Arguments

`object`	An object either of type SccsIntervalData as created using the createSccsIntervalData function, or an object of type `SccsModel` as created using the `fitSccsModel()` function.
`exposureCovariateId`	Covariate Id for the health exposure of interest.
`alpha`	Type I error.
`power`	1 - beta, where beta is the type II error.
`twoSided`	Consider a two-sided test?
`method`	The type of sample size formula that will be used. Allowable values are "proportion", "binomial", "SRL1", "SRL2", or "ageEffects". Currently "ageEffects" is not supported.

Details

Compute the minimum detectable relative risk (MDRR) for a given study population, using the observed time at risk and total time in days and number of events. Five sample size formulas are implemented: sampling proportion, binomial proportion, 2 signed root likelihood ratio methods, and likelihood extension for age effects. The expressions by Musonda (2006) are used.

Value

A data frame with the MDRR, number of events, time at risk, and total time.

References

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

`computePreExposureGainP` *Compute P for pre-exposure risk gain*

Description

This function is deprecated. Use `computePreExposureGain()` instead.

Usage

```
computePreExposureGainP(sccsData, studyPopulation, exposureEraId = NULL)
```

Arguments

`sccsData`	An object of type SccsData as created using the getDbSccsData function.
`studyPopulation`	An object created using the `createStudyPopulation()` function.
`exposureEraId`	The exposure to create the era data for. If not specified it is assumed to be the one exposure for which the data was loaded from the database.

Value

A one-sided p-value for whether the rate before exposure is higher than after, against the null of no change.#'

`computeTimeStability` *Check stability of outcome rate over time*

Description

Check stability of outcome rate over time

Usage

```
computeTimeStability(
  studyPopulation,
  sccsModel = NULL,
  maxRatio = 1.1,
  alpha = 0.05
)
```

Arguments

`studyPopulation`	An object created using the `createStudyPopulation()` function.
`sccsModel`	Optional: A fitted SCCS model as created using `fitSccsModel()`. If the model contains splines for seasonality and or calendar time these will be adjusted for before computing stability.
`maxRatio`	The maximum global ratio between the observed and expected count.
`alpha`	The alpha (type 1 error) used to test for stability.

Details

DEPRECATED. Use `checkTimeStabilityAssumption()` instead.

Value

A tibble with one row and three columns: `ratio` indicates the estimated mean ratio between observed and expected. `p` is the p-value against the null-hypothesis that the ratio is smaller than `maxRatio`, and `pass` is `TRUE` if `p` is greater than `alpha`.

`convertJsonToSccsAnalysesSpecifications`
Convert JSON to SccsAnalysesSpecifications

Description

Convert JSON to SccsAnalysesSpecifications

Usage

```
convertJsonToSccsAnalysesSpecifications(json)
```

Arguments

`json`	A string containing the JSON representation.

Value

An object of type `SccsAnalysesSpecifications`.

`convertSccsAnalysesSpecificationsToJson`

Convert SccsAnalysesSpecifications to JSON

Description

Convert SccsAnalysesSpecifications to JSON

Usage

```
convertSccsAnalysesSpecificationsToJson(sccsAnalysesSpecifications)
```

Arguments

`sccsAnalysesSpecifications`	An object of type `SccsAnalysesSpecifications` as created by the `createSccsAnalysesSpecificatio` function.

Value

A string containing the JSON representation.

`convertUntypedListToSccsAnalysesSpecifications`

Convert untyped list to SccsAnalysesSpecifications

Description

Convert untyped list to SccsAnalysesSpecifications

Usage

```
convertUntypedListToSccsAnalysesSpecifications(untypedList)
```

Arguments

`untypedList`	A list of untyped objects. For example, these could be objects from a call to `jsonlite::fromJSON()`. Importantly, `simplifyDataFrame` must be set to `FALSE` when doing so.

Value

An object of type `SccsAnalysesSpecifications`.

`createAgeCovariateSettings`

Create age covariate settings

Description

Create age covariate settings

Usage

```
createAgeCovariateSettings(
  ageKnots = 5,
  allowRegularization = FALSE,
  computeConfidenceIntervals = FALSE
)
```

Arguments

`ageKnots`	If a single number is provided this is assumed to indicate the number of knots to use for the spline, and the knots are automatically spaced according to equal percentiles of the data. If more than one number is provided these are assumed to be the exact location of the knots in age-days
`allowRegularization`	When fitting the model, should the covariates defined here be allowed to be regularized?
`computeConfidenceIntervals`	Should confidence intervals be computed for the covariates defined here? Setting this to FALSE might save computing time when fitting the model. Will be turned to FALSE automatically when `allowRegularization = TRUE`.

Details

Create an object specifying whether and how age should be included in the model. Age can be included by splitting patient time into calendar months. During a month, the relative risk attributed to age is assumed to be constant, and the risk from month to month is modeled using a quadratic spline.

Value

An object of type `AgeCovariateSettings`.

`createCalendarTimeCovariateSettings`
Create calendar time settings

Description

Create calendar time settings

Usage

```
createCalendarTimeCovariateSettings(
  calendarTimeKnots = 5,
  allowRegularization = FALSE,
  computeConfidenceIntervals = FALSE
)
```

Arguments

`calendarTimeKnots`	If a single number is provided this is assumed to indicate the number of knots to use for the spline. See details on how knots are placed. If a series of dates is provided these are assumed to be the exact location of the knots.
`allowRegularization`	When fitting the model, should the covariates defined here be allowed to be regularized?
`computeConfidenceIntervals`	Should confidence intervals be computed for the covariates defined here? Setting this to FALSE might save computing time when fitting the model. Will be turned to FALSE automatically when `allowRegularization = TRUE`.

Details

Create an object specifying whether and how calendar time should be included in the model. Calendar time can be included by splitting patient time into calendar months. During a month, the relative risk attributed to calendar time is assumed to be constant, and the risk from month to month is modeled using a quadratic spline.

Whereas the seasonality covariate uses a cyclic spline, repeating every year, this calendar time covariate can model trends over years.

If a number of knots is specified, knots are automatically spaced according to equal percentiles of the data (people observed). If more than one study period is provided, two more knots (start and end) are automatically added for each additional study period. So if `calendarTimeKnots = 5` and there are 3 study periods, the total number of knots will be 5 + 2 * (3 - 1) = 9.#'

Value

An object of type `CalendarTimeCovariateSettings`.

`createControlIntervalSettings`

Create control interval settings

Description

Create control interval settings

Usage

```
createControlIntervalSettings(
  includeEraIds = NULL,
  excludeEraIds = NULL,
  start = 0,
  startAnchor = "era start",
  end = 0,
  endAnchor = "era end",
  firstOccurrenceOnly = FALSE
)
```

Arguments

`includeEraIds`	One or more IDs of variables in the SccsData object that should be used to construct this covariate. If no IDs are specified, all variables will be used.
`excludeEraIds`	One or more IDs of variables in the [SccsData] object that should not be used to construct this covariate.
`start`	The start of the control interval (in days) relative to the `startAnchor`.
`startAnchor`	The anchor point for the start of the control interval. Can be `"era start"` or `"era end"`.
`end`	The end of the control interval (in days) relative to the `endAnchor`.
`endAnchor`	The anchor point for the end of the control interval. Can be `"era start"` or `"era end"`.
`firstOccurrenceOnly`	Should only the first occurrence of the exposure be used?

Details

Create an object specifying how to create a control interval for the self-controlled risk interval (SCRI) design.

Value

An object of type `ControlIntervalSettings`.

`createCreateSccsIntervalDataArgs`

Create a parameter object for the `createSccsIntervalData()` *function*

Description

Create a parameter object for the `createSccsIntervalData()` function

Usage

```
createCreateSccsIntervalDataArgs(
  eraCovariateSettings,
  ageCovariateSettings = NULL,
  seasonalityCovariateSettings = NULL,
  calendarTimeCovariateSettings = NULL,
  minCasesForTimeCovariates = 10000,
  endOfObservationEraLength = 30,
  eventDependentObservation = FALSE
)
```

Arguments

`eraCovariateSettings`	Either an object of type `EraCovariateSettings` as created using the `createEraCovariateSettings()` function, or a list of such objects.
`ageCovariateSettings`	An object of type `AgeCovariateSettings` as created using the `createAgeCovariateSettings()` function.
`seasonalityCovariateSettings`	An object of type `SeasonalityCovariateSettings` as created using the `createSeasonalityCovariate` function.
`calendarTimeCovariateSettings`	An object of type `CalendarTimeCovariateSettings` as created using the `createCalendarTimeCovaria` function.
`minCasesForTimeCovariates`	Minimum number of cases to use to fit age, season and calendar time splines. If needed (and available), cases that are not exposed will be included.
`endOfObservationEraLength`	Length in days of the probe that is inserted at the end of a patient's observation time. This probe will be used to test whether there is event- dependent observation end. Set to 0 to not include the probe.
`eventDependentObservation`	Should the extension proposed by Farrington et al. be used to adjust for event-dependent observation time?

Details

Create an object defining the parameter values.

Value

An object of type `CreateSccsIntervalDataArgs`.

References

Farrington, C. P., Anaya-Izquierdo, A., Whitaker, H. J., Hocine, M.N., Douglas, I., and Smeeth, L. (2011). Self-Controlled case series analysis with event-dependent observation periods. Journal of the American Statistical Association 106 (494), 417-426

`createCreateScriIntervalDataArgs`

Create a parameter object for the `createScriIntervalData()` *function*

Description

Create a parameter object for the `createScriIntervalData()` function

Usage

```
createCreateScriIntervalDataArgs(eraCovariateSettings, controlIntervalSettings)
```

Arguments

`eraCovariateSettings`	Either an object of type `EraCovariateSettings` as created using the `createEraCovariateSettings()` function, or a list of such objects.
`controlIntervalSettings`	An object of type `ControlIntervalSettings` as created using the `createControlIntervalSettings()` function.

Details

Create an object defining the parameter values.

Value

An object of type `CreateScriIntervalDataArgs`.

`createCreateStudyPopulationArgs` *Create a parameter object for the `createStudyPopulation()` function*

Description

Create a parameter object for the `createStudyPopulation()` function

Usage

```
createCreateStudyPopulationArgs(
  firstOutcomeOnly = FALSE,
  naivePeriod = 0,
  minAge = NULL,
  maxAge = NULL,
  genderConceptIds = NULL,
  restrictTimeToEraId = NULL
)
```

Arguments

`firstOutcomeOnly`	Whether only the first occurrence of an outcome should be considered.
`naivePeriod`	The number of days at the start of a patient's observation period that should not be included in the risk calculations. Note that the naive period can be used to determine current covariate status right after the naive period, and whether an outcome is the first one.
`minAge`	Minimum age at which patient time will be included in the analysis. Note that information prior to the min age is still used to determine exposure status after the minimum age (e.g. when a prescription was started just prior to reaching the minimum age). Also, outcomes occurring before the minimum age is reached will be considered as prior outcomes when using first outcomes only. Age should be specified in years, but non-integer values are allowed. If not specified, no age restriction will be applied.
`maxAge`	Maximum age at which patient time will be included in the analysis. Age should be specified in years, but non-integer values are allowed. If not specified, no age restriction will be applied.
`genderConceptIds`	Set of gender concept IDs to restrict the population to. If not specified, no restriction on gender will be applied.
`restrictTimeToEraId`	If provided, study time (for all patients) will be restricted to the calendar time when that era was observed in the data. For example, if the era ID refers to a drug, study time will be restricted to when the drug was on the market.

Details

Create an object defining the parameter values.

Value

An object of type `CreateStudyPopulationArgs`.

`createDefaultSccsMultiThreadingSettings`

Create default SelfControlledCaseSeries multi-threading settings

Description

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

Usage

```
createDefaultSccsMultiThreadingSettings(maxCores)
```

Arguments

`maxCores`	Maximum number of CPU cores to use.

Value

An object of type `SccsMultiThreadingSettings`.

See Also

`createSccsMultiThreadingSettings()`

Examples

```
settings <- createDefaultSccsMultiThreadingSettings(10)
```

`createEraCovariateSettings` *Create era covariate settings*

Description

Create era covariate settings

Usage

```
createEraCovariateSettings(
  includeEraIds,
  excludeEraIds = NULL,
  label = "Covariates",
  stratifyById = FALSE,
  start = 0,
  startAnchor = "era start",
  end = 0,
  endAnchor = "era end",
  firstOccurrenceOnly = FALSE,
  allowRegularization = FALSE,
  profileLikelihood = FALSE,
  exposureOfInterest = FALSE,
  preExposure = start < 0
)
```

Arguments

`includeEraIds`	One or more IDs of variables in the SccsData object that should be used to construct this covariate. If set to NULL, all variables will be used.
`excludeEraIds`	One or more IDs of variables in the [SccsData] object that should not be used to construct this covariate.
`label`	A label used to identify the covariates created using these settings.
`stratifyById`	Should a single covariate be created for every ID in the SccsData object, or should a single covariate be constructed? For example, if the IDs identify exposures to different drugs, should a covariate be constructed for every drug, or a single covariate for exposure to any of these drugs. Note that overlap will be considered a single exposure.
`start`	The start of the risk window (in days) relative to the `startAnchor`.
`startAnchor`	The anchor point for the start of the risk window. Can be `"era start"` or `"era end"`.
`end`	The end of the risk window (in days) relative to the `endAnchor`.
`endAnchor`	The anchor point for the end of the risk window. Can be `"era start"` or `"era end"`.

`firstOccurrenceOnly`	Should only the first occurrence of the exposure be used?
`allowRegularization`	When fitting the model, should the covariates defined here be allowed to be regularized?
`profileLikelihood`	When fitting the model, should the likelihood profile be computed for the covariate defined here? The likelihood profile can be used to avoid making normal approximations on the likelihood and can be used in methods specifically designed to make use of the profile, but may take a while to compute.
`exposureOfInterest`	If TRUE, the fitted coefficient for this variable will be reported when using `runSccsAnalyses()`. Requires `includeEraIds` to be a exposure reference ID as defined in `createExposure()`.
`preExposure`	If TRUE, this variable will be used for the pre-exposure diagnostic.

Details

Create an object specifying how to create a (set of) era-based covariates.

Value

An object of type `EraCovariateSettings`.

`createExposure` *Create exposure definition*

Description

Create exposure definition

Usage

```
createExposure(exposureId, exposureIdRef = "exposureId", trueEffectSize = NA)
```

Arguments

`exposureId`	An integer used to identify the exposure in the exposure cohort table.
`exposureIdRef`	A string used to refer to the exposure when defining covariates using the `createEraCovariateSettings(` function.
`trueEffectSize`	For negative and positive controls: the known true effect size. To be used for empirical calibration. Negative controls have `trueEffectSize = 1`. If the true effect size is unknown, use `trueEffectSize = NA`.

Details

Create an exposure definition, to be used with the createExposuresOutcome function.

Value

An object of type `Exposure`.

`createExposuresOutcome` *Create a exposures-outcome combination.*

Description

Create a exposures-outcome combination.

Usage

```
createExposuresOutcome(outcomeId, exposures, nestingCohortId = NULL)
```

Arguments

`outcomeId`	An integer used to identify the outcome in the outcome cohort table.
`exposures`	A list of object of type `Exposure` as created by `createExposure()`.
`nestingCohortId`	(Optional) the nesting cohort ID.

Details

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

Value

An object of type `ExposuresOutcome`.

`createFitSccsModelArgs` *Create a parameter object for the function fitSccsModel*

Description

Create a parameter object for the function fitSccsModel

Usage

```
createFitSccsModelArgs(
  prior = createPrior("laplace", useCrossValidation = TRUE),
 control = createControl(cvType = "auto", selectorType = "byPid", startingVariance =
    0.1, seed = 1, resetCoefficients = TRUE, noiseLevel = "quiet"),
  profileGrid = NULL,
  profileBounds = c(log(0.1), log(10))
)
```

Arguments

`prior`	The prior used to fit the model. See Cyclops::createPrior for details.
`control`	The control object used to control the cross-validation used to determine the hyperparameters of the prior (if applicable). See Cyclops::createControl for details.
`profileGrid`	A one-dimensional grid of points on the log(relative risk) scale where the likelihood for coefficient of variables is sampled. See details.
`profileBounds`	The bounds (on the log relative risk scale) for the adaptive sampling of the likelihood function.

Details

Create an object defining the parameter values.

Likelihood profiling is only done for variables for which `profileLikelihood` is set to `TRUE` when calling `createEraCovariateSettings()`. Either specify the `profileGrid` for a completely user-defined grid, or `profileBounds` for an adaptive grid. Both should be defined on the log IRR scale. When both `profileGrid` and `profileGrid` are `NULL` likelihood profiling is disabled.

To make use of the more efficient Hermite interpolation in evidence synthesis, set `profileGrid = seq(log(0.1), log(10), length.out = 8)` and `profileBounds = NULL`.

Value

An object of type `FitSccsModelArgs`.

---

createGetDbSccsDataArgs

Create a parameter object for the function getDbSccsData

Description

Create a parameter object for the function getDbSccsData

Usage

```
createGetDbSccsDataArgs(
  nestingCohortId = NULL,
  deleteCovariatesSmallCount = 0,
  studyStartDates = c(),
  studyEndDates = c(),
  maxCasesPerOutcome = 0,
  exposureIds = "exposureId",
  customCovariateIds = NULL
)
```

Arguments

`nestingCohortId`	A cohort definition ID identifying the records in the nestingCohortTable to use as nesting cohort.
`deleteCovariatesSmallCount`	The minimum count for a covariate to appear in the data to be kept.
`studyStartDates`	A character object specifying the minimum dates where data is used. Date format is 'yyyymmdd'. Use "" to indicate all time prior. See section for more information.
`studyEndDates`	A character object specifying the maximum dates where data is used. Date format is 'yyyymmdd'. Use "" to indicate to the end of observation. See section for more information.
`maxCasesPerOutcome`	If there are more than this number of cases for a single outcome cases will be sampled to this size. maxCasesPerOutcome = 0 indicates no maximum size.
`exposureIds`	A list of identifiers to extract from the exposure table. If exposureTable = DRUG_ERA, exposureIds should be CONCEPT_ID. If exposureTable = "drug_era", exposureIds is used to select the drug_concept_id. If no exposure IDs are provided, all drugs or cohorts in the exposureTable are included as exposures.
`customCovariateIds`	A list of cohort definition IDs identifying the records in the customCovariateTable to use for building custom covariates.

Details

Create an object defining the parameter values.

Value

An object of type `GetDbSccsDataArgs`.

`createResultsDataModel`

Create the results data model tables on a database server.

Description

Create the results data model tables on a database server.

Usage

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

Arguments

`connectionDetails`	DatabaseConnector connectionDetails instance @seealsoDatabaseConnector::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.

`createSccsAnalysesSpecifications`

Create full SCCS analysis specifications

Description

Create full SCCS analysis specifications

Usage

```
createSccsAnalysesSpecifications(
  sccsAnalysisList,
  exposuresOutcomeList,
  analysesToExclude = NULL,
  combineDataFetchAcrossOutcomes = FALSE,
 sccsDiagnosticThresholds = SelfControlledCaseSeries::createSccsDiagnosticThresholds(),
  controlType = "outcome"
)
```

Arguments

`sccsAnalysisList`	A list of objects of type `SccsAnalysis` as created using the `createSccsAnalysis()` function.
`exposuresOutcomeList`	A list of objects of type `ExposuresOutcome` as created using the `createExposuresOutcome()` function.
`analysesToExclude`	Analyses to exclude. See the Analyses to Exclude section for details.
`combineDataFetchAcrossOutcomes`	Should fetching data from the database be done one outcome at a time, or for all outcomes in one fetch? Combining fetches will be more efficient if there is large overlap in the subjects that have the different outcomes.
`sccsDiagnosticThresholds`	An object of type `SccsDiagnosticThresholds` as created using createSccsDiagnosticThresholds().

`controlType`	Type of negative (and positive) controls. Can be "outcome" or "exposure". When set to "outcome", controls with the same exposure (and nesting cohort) are grouped together for calibration. When set to "exposure", controls with the same outcome are grouped together.

Details

Analyses to Exclude:

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

- exposureId
- outcomeId
- nestingCohortId
- analysisId

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

Value

An object of type `SccsAnalysesSpecifications`.

`createSccsAnalysis` *Create a SelfControlledCaseSeries analysis specification*

Description

Create a SelfControlledCaseSeries analysis specification

Usage

```
createSccsAnalysis(
  analysisId = 1,
  description = "",
  getDbSccsDataArgs,
  createStudyPopulationArgs,
  createIntervalDataArgs,
  fitSccsModelArgs
)
```

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.
`getDbSccsDataArgs`	An object representing the arguments to be used when calling the getDbSccsData function.
`createStudyPopulationArgs`	An object representing the arguments to be used when calling the getDbSccsData function.
`createIntervalDataArgs`	An object representing the arguments to be used when calling the createSccsIntervalData or createScriIntervalData function.
`fitSccsModelArgs`	An object representing the arguments to be used when calling the fitSccsModel function.

Value

An object of type `SccsAnalysis`, to be used with the runSccsAnalyses function.

`createSccsDiagnosticThresholds`

Create SCCS diagnostics thresholds

Description

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

Usage

```
createSccsDiagnosticThresholds(
  mdrrThreshold = 10,
  easeThreshold = 0.25,
  timeTrendMaxRatio = 1.1,
  rareOutcomeMaxPrevalence = 0.1,
  eventObservationDependenceNullBounds = c(0.5, 2),
  eventExposureDependenceNullBounds = c(0.8, 1.25)
)
```

Arguments

`mdrrThreshold`	What is the maximum allowed minimum detectable relative risk (MDRR)?
`easeThreshold`	What is the maximum allowed expected absolute systematic error (EASE).
`timeTrendMaxRatio`	The maximum global ratio between the observed and expected count for the time stability diagnostic.

`rareOutcomeMaxPrevalence`	The maximum allowed prevalence (proportion of people with the outcome) allowed when restricting to first outcome only.
`eventObservationDependenceNullBounds`	The bounds for the null hypothesis for the incidence rate ratio of the end-of-observation probe window.
`eventExposureDependenceNullBounds`	The bounds for the null hypothesis for the incidence rate of the pre-exposure window.

Value

An object of type `SccsDiagnosticThresholds`.

`createSccsIntervalData`

Create SCCS era data

Description

Create SCCS era data

Usage

```
createSccsIntervalData(studyPopulation, sccsData, createSccsIntervalDataArgs)
```

Arguments

`studyPopulation`	An object created using the `createStudyPopulation()` function.
`sccsData`	An object of type SccsData as created using the getDbSccsData function.
`createSccsIntervalDataArgs`	An object of type `CreateSccsIntervalDataArgs` as created by the `createCreateSccsIntervalDataAr` function.

Details

This function creates covariates based on the data in the `sccsData` argument, according to the provided settings. It chops patient time into periods during which all covariates remain constant. The output details these periods, their durations, and a sparse representation of the covariate values.

Value

An object of type `SccsIntervalData`.

`createSccsMultiThreadingSettings`
Create SelfControlledCaseSeries multi-threading settings

Description

Create SelfControlledCaseSeries multi-threading settings

Usage

```
createSccsMultiThreadingSettings(
  getDbSccsDataThreads = 1,
  createStudyPopulationThreads = 1,
  createIntervalDataThreads = 1,
  fitSccsModelThreads = 1,
  cvThreads = 1,
  calibrationThreads = 1
)
```

Arguments

`getDbSccsDataThreads`	The number of parallel threads to use for building the `SccsData` objects.
`createStudyPopulationThreads`	The number of parallel threads to use for building the `studyPopulation` objects.
`createIntervalDataThreads`	The number of parallel threads to use for building the `SccsIntervalData` objects.
`fitSccsModelThreads`	The number of parallel threads to use for fitting the models.
`cvThreads`	The number of parallel threads to use for the cross- validation when estimating the hyperparameter for the outcome model. Note that the total number of CV threads at one time could be `fitSccsModelThreads * cvThreads`.
`calibrationThreads`	The number of parallel threads to use for empirical calibration.

Value

An object of type `SccsMultiThreadingSettings`.

See Also

`createDefaultSccsMultiThreadingSettings()`

`createSccsSimulationSettings` *Create SCCS simulation settings*

Description

Create SCCS simulation settings

Usage

```
createSccsSimulationSettings(
  meanPatientTime = 4 * 365,
  sdPatientTime = 2 * 365,
  minAge = 18 * 365,
  maxAge = 65 * 365,
  minBaselineRate = 0.001,
  maxBaselineRate = 0.01,
  minCalendarTime = as.Date("2000-01-01"),
  maxCalendarTime = as.Date("2010-01-01"),
  eraIds = c(1, 2),
  patientUsages = c(0.2, 0.1),
  usageRate = c(0.01, 0.01),
  usageRateSlope = c(0, 0),
  meanPrescriptionDurations = c(14, 30),
  sdPrescriptionDurations = c(7, 14),
  simulationRiskWindows = list(createSimulationRiskWindow(relativeRisks = 1),
    createSimulationRiskWindow(relativeRisks = 1.5)),
  includeAgeEffect = TRUE,
  ageKnots = 5,
  includeSeasonality = TRUE,
  seasonKnots = 5,
  includeCalendarTimeEffect = TRUE,
  calendarTimeKnots = 5,
  calendarTimeMonotonic = FALSE,
  outcomeId = 10
)
```

Arguments

`meanPatientTime`	Mean number of observation days per patient.
`sdPatientTime`	Standard deviation of the observation days per patient.
`minAge`	The minimum age in days.
`maxAge`	The maximum age in days.
`minBaselineRate`	The minimum baseline rate (per day).

`maxBaselineRate`	The maximum baseline rate (per day).
`minCalendarTime`	The minimum date patients are to be observed.
`maxCalendarTime`	The maximum date patients are to be observed.
`eraIds`	The IDs for the covariates to be generated.
`patientUsages`	The fraction of patients that use the drugs.
`usageRate`	The rate of prescriptions per person that uses the drug.
`usageRateSlope`	The change in the usage rate from one day to the next. `usageRate` is the intercept at day 0
`meanPrescriptionDurations`	The mean duration of a prescription, per drug.
`sdPrescriptionDurations`	The standard deviation of the duration of a prescription, per drug.
`simulationRiskWindows`	One or a list of objects of type `SimulationRiskWindow` as created using the `createSimulationRiskWindow()` function. function.
`includeAgeEffect`	Include an age effect for the outcome?
`ageKnots`	Number of knots in the age spline.
`includeSeasonality`	Include seasonality for the outcome?
`seasonKnots`	Number of knots in the seasonality spline.
`includeCalendarTimeEffect`	Include a calendar time effect for the outcome?
`calendarTimeKnots`	Number of knots in the calendar time spline.
`calendarTimeMonotonic`	Should the calender time effect be monotonic?
`outcomeId`	The ID to be used for the outcome.

Details

Create an object of settings for an SCCS simulation.

Value

An object of type `SccsSimulationSettings`.

`createScriIntervalData` *Create Self-Controlled Risk Interval (SCRI) era data*

Description

Create Self-Controlled Risk Interval (SCRI) era data

Usage

```
createScriIntervalData(studyPopulation, sccsData, createScriIntervalDataArgs)
```

Arguments

`studyPopulation`	An object created using the `createStudyPopulation()` function.
`sccsData`	An object of type SccsData as created using the getDbSccsData function.
`createScriIntervalDataArgs`	An object of type `CreateScriIntervalDataArgs` as created by the `createCreateScriIntervalDataAr` function.

Details

This function creates interval data according to the elf-Controlled Risk Interval (SCRI) design. Unlike the generic SCCS design, where all patient time is used to establish a background rate, in the SCRI design a specific control interval (relative to the exposure) needs to be defined. The final model will only include time that is either part of the risk interval (defined using the `eraCovariateSettings` argument, or the control interval (defined using `controlIntervalSettings`).

Value

An object of type SccsIntervalData.

References

Greene SK, Kulldorff M, Lewis EM, Li R, Yin R, Weintraub ES, Fireman BH, Lieu TA, Nordin JD, Glanz JM, Baxter R, Jacobsen SJ, Broder KR, Lee GM. Near real-time surveillance for influenza vaccine safety: proof-of-concept in the Vaccine Safety Datalink Project. Am J Epidemiol. 2010 Jan 15;171(2):177-88. doi: 10.1093/aje/kwp345.

`createSeasonalityCovariateSettings` *Create seasonality settings*

Description

Create seasonality settings

Usage

```
createSeasonalityCovariateSettings(
  seasonKnots = 5,
  allowRegularization = FALSE,
  computeConfidenceIntervals = FALSE
)
```

Arguments

`seasonKnots`	If a single number is provided this is assumed to indicate the number of knots to use for the spline, and the knots are automatically equally spaced across the year. If more than one number is provided these are assumed to be the exact location of the knots in days relative to the start of the year.
`allowRegularization`	When fitting the model, should the covariates defined here be allowed to be regularized?
`computeConfidenceIntervals`	Should confidence intervals be computed for the covariates defined here? Setting this to FALSE might save computing time when fitting the model. Will be turned to FALSE automatically when `allowRegularization = TRUE`.

Details

Create an object specifying whether and how seasonality should be included in the model. Seasonality can be included by splitting patient time into calendar months. During a month, the relative risk attributed to season is assumed to be constant, and the risk from month to month is modeled using a cyclic quadratic spline.

Value

An object of type `SeasonalityCovariateSettings`.

---

createSimulationRiskWindow *Create a risk window definition for simulation*

Description

Create a risk window definition for simulation

Usage

```
createSimulationRiskWindow(
  start = 0,
  end = 0,
  endAnchor = "era end",
  splitPoints = c(),
  relativeRisks = c(0)
)
```

Arguments

`start`	Start of the risk window relative to exposure start.
`end`	The end of the risk window (in days) relative to the `endAnchor`.
`endAnchor`	The anchor point for the end of the risk window. Can be `"era start"` or `"era end"`.
`splitPoints`	Subdivision of the risk window in to smaller sub-windows.
`relativeRisks`	Either a single number representing the relative risk in the risk window, or when splitPoints have been defined a vector of relative risks, one for each sub-window.

Value

An object of type `SimulationRiskWindow`.

---

createStudyPopulation *Create a study population*

Description

Create a study population

Usage

```
createStudyPopulation(sccsData, outcomeId = NULL, createStudyPopulationArgs)
```

Arguments

`sccsData`	An object of type SccsData as created using the getDbSccsData function.
`outcomeId`	The outcome to create the era data for. If not specified it is assumed to be the one outcome for which the data was loaded from the database.
`createStudyPopulationArgs`	An object of type `CreateStudyPopulationArgs` as created using the `createCreateStudyPopulationAr` function.

Details

Create a study population for a specific outcome, applying several restrictions.

Value

A `list` specifying the study population, with the following items:

- `cases`: A `tibble` with one row per observation period of a person with the outcome.
- `outcomes`: A `tibble` listing the days when a case has the outcome.
- `metaData`: A `list` with meta data about the study population, including the attrition.

`cyclicSplineDesign` *Create a design matrix for a cyclic spline*

Description

Create a design matrix for a cyclic spline

Usage

```
cyclicSplineDesign(x, knots, ord = 3)
```

Arguments

`x`	Vector of coordinates of the points to be interpolated.
`knots`	Location of the knots.
`ord`	Order of the spline function. `ord` = 3 implies quadratic.

Details

This function is used by other functions in this package.

`exportToCsv` *Export SCCSresults to CSV files*

Description

Export SCCSresults to CSV files

Usage

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

Arguments

`outputFolder`	The folder where runCmAnalyses() generated all results.
`exportFolder`	The folder where the CSV files will written.
`databaseId`	A unique ID for the database. This will be appended to most tables.
`minCellCount`	To preserve privacy: the minimum number of subjects contributing to a count before it can be included in the results. If the count is below this threshold, it will be set to `-minCellCount`.
`maxCores`	Maximum number of CPU cores to use.

Details

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

Value

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

`fitSccsModel` *Fit the SCCS model*

Description

Fit the SCCS model

Usage

```
fitSccsModel(sccsIntervalData, fitSccsModelArgs)
```

Arguments

`sccsIntervalData`	An object of type SccsIntervalData as created using the createSccsIntervalData function.
`fitSccsModelArgs`	An object of type `FitSccsModelArgs` as created by the `createFitSccsModelArgs()` function.

Details

Fits the SCCS model as a conditional Poisson regression. When allowed, coefficients for some or all covariates can be regularized.

Value

An object of type `SccsModel`. Generic functions `print`, `coef`, and `confint` are available.

References

Suchard, M.A., Simpson, S.E., Zorych, I., Ryan, P., and Madigan, D. (2013). Massive parallelization of serial inference algorithms for complex generalized linear models. ACM Transactions on Modeling and Computer Simulation 23, 10

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

Description

Get the attrition table for a population

Usage

```
getAttritionTable(object)
```

Arguments

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

Value

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

`getDataMigrator` *Get database migrations instance*

Description

Returns ResultModelManager DataMigrationsManager instance.

Usage

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

Arguments

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

Value

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

---

`getDbSccsData` *Load data for SCCS from the database*

Description

Load all data needed to perform an SCCS analysis from the database.

Usage

```
getDbSccsData(
  connectionDetails,
  cdmDatabaseSchema,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  outcomeDatabaseSchema = cdmDatabaseSchema,
  outcomeTable = "condition_era",
  outcomeIds,
  exposureDatabaseSchema = cdmDatabaseSchema,
  exposureTable = "drug_era",
  customCovariateDatabaseSchema = cdmDatabaseSchema,
  customCovariateTable = "cohort",
  nestingCohortDatabaseSchema = cdmDatabaseSchema,
  nestingCohortTable = "cohort",
  getDbSccsDataArgs
)
```

Arguments

`connectionDetails`	An R object of type `ConnectionDetails` created using the function `DatabaseConnector::createConne` function.
`cdmDatabaseSchema`	The name of the database schema that contains the OMOP CDM instance. Requires read permissions to this database. On SQL Server, this should specify both the database and the schema, so for example 'cdm_instance.dbo'.
`tempEmulationSchema`	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
`outcomeDatabaseSchema`	The name of the database schema that is the location where the data used to define the outcome cohorts is available. If `outcomeTable = "condition_era"`, `outcomeDatabaseSchema` is not used. Requires read permissions to this database.
`outcomeTable`	The table name that contains the outcome cohorts. If `outcomeTable` is not `"condition_era"`, then expectation is `outcomeTable` has format of cohort table (see details).
`outcomeIds`	A list of IDs used to define outcomes. If `outcomeTable` is not `"condition_era"` the list contains records found in the `cohort_definition_id` field.
`exposureDatabaseSchema`	The name of the database schema that is the location where the exposure data used to define the exposure eras is available. If `exposureTable = "drug_era"`, `exposureDatabaseSchema` is not used but assumed to be equal to `cdmDatabaseSchema`. Requires read permissions to this database.
`exposureTable`	The tablename that contains the exposure cohorts. If `exposureTable` is not "drug_era", then expectation is `exposureTable` has format of a cohort table (see details).

`customCovariateDatabaseSchema`	The name of the database schema that is the location where the custom covariate data is available.
`customCovariateTable`	Name of the table holding the custom covariates. This table should have the same structure as the cohort table (see details).
`nestingCohortDatabaseSchema`	The name of the database schema that is the location where the nesting cohort is defined.
`nestingCohortTable`	Name of the table holding the nesting cohort. This table should have the same structure as the cohort table (see details).
`getDbSccsDataArgs`	An object of type `GetDbSccsDataArgs` as created by the `createGetDbSccsDataArgs()` function.

Details

This function downloads several types of information:

- Information on the occurrences of the outcome(s) of interest. Note that information for multiple outcomes can be fetched in one go, and later the specific outcome can be specified for which we want to build a model.
- Information on the observation time and age for the people with the outcomes.
- Information on exposures of interest which we want to include in the model.

Five different database schemas can be specified, for five different types of information: The

- **cdmDatabaseSchema** is used to extract patient age and observation period. The
- **outcomeDatabaseSchema** is used to extract information about the outcomes, the
- **exposureDatabaseSchema** is used to retrieve information on exposures, and the
- **customCovariateDatabaseSchema** is optionally used to find additional, user-defined covariates. All four locations could point to the same database schema.
- **nestingCohortDatabaseSchema** is optionally used to define a cohort in which the analysis is nested, for example a cohort of diabetics patients.

All five locations could point to the same database schema.

Cohort tables are assumed to have the following fields: `cohort_definition_id`, `subject_id`, `cohort_start_date`, and `cohort_end_date`.

Value

An SccsData object.

Study period start and end dates

A study start and end date define a period when patient data will be included in the analysis. Multiple non-overlapping periods can be defined, which for example will allow for excluding the time of the COVID pandemic, when most outcome rates were unstable.

`getDiagnosticsSummary` *Get a summary report of the analyses diagnostics*

Description

Get a summary report of the analyses diagnostics

Usage

```
getDiagnosticsSummary(outputFolder)
```

Arguments

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

Value

A tibble containing summary diagnostics for each outcome-covariate-analysis combination.

`getFileReference` *Get file reference*

Description

Get file reference

Usage

```
getFileReference(outputFolder)
```

Arguments

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

Value

A tibble containing the names of the files for various artifacts created for each outcome-exposures set.

`getModel` *Output the full model*

Description

Output the full model

Usage

```
getModel(sccsModel)
```

Arguments

`sccsModel`	An object of type `SccsModel` as created using the `fitSccsModel()` function.

Value

A `tibble` with the coefficients and confidence intervals (when not-regularized) for all covariates in the model.

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

Description

Get specifications for SelfControlledCaseSeries results data model

Usage

```
getResultsDataModelSpecifications()
```

Value

A tibble data frame object with specifications

`getResultsSummary` *Get a summary report of the analyses results*

Description

Get a summary report of the analyses results

Usage

```
getResultsSummary(outputFolder)
```

Arguments

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

Value

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

`hasAgeEffect` *Does the model contain an age effect?*

Description

Does the model contain an age effect?

Usage

```
hasAgeEffect(sccsModel)
```

Arguments

`sccsModel`	An object of type `SccsModel` as created using the `fitSccsModel()` function.

Value

TRUE if the model contains an age effect, otherwise FALSE.

`hasCalendarTimeEffect` *Does the model contain a calendar time effect?*

Description

Does the model contain a calendar time effect?

Usage

```
hasCalendarTimeEffect(sccsModel)
```

Arguments

`sccsModel`	An object of type `SccsModel` as created using the `fitSccsModel()` function.

Value

TRUE if the model contains an age effect, otherwise FALSE.

`hasSeasonality` *Does the model contain a seasonality effect?*

Description

Does the model contain a seasonality effect?

Usage

```
hasSeasonality(sccsModel)
```

Arguments

`sccsModel`	An object of type `SccsModel` as created using the `fitSccsModel()` function.

Value

TRUE if the model contains an age effect, otherwise FALSE.

`isSccsData` *Check whether an object is a SccsData object*

Description

Check whether an object is a SccsData object

Usage

```
isSccsData(x)
```

Arguments

`x`	The object to check.

Value

A logical value.

`isSccsIntervalData` *Check whether an object is a SccsIntervalData object*

Description

Check whether an object is a SccsIntervalData object

Usage

```
isSccsIntervalData(x)
```

Arguments

`x`	The object to check.

Value

A logical value.

`loadExposuresOutcomeList` *Load a list of* `ExposuresOutcome` *from file*

Description

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

Usage

```
loadExposuresOutcomeList(file)
```

Arguments

`file`	The name of the file

Value

A list of objects of type `ExposuresOutcome`.

`loadSccsAnalysisList` *Load a list of sccsAnalysis from file*

Description

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

Usage

```
loadSccsAnalysisList(file)
```

Arguments

`file`	The name of the file

Value

A list of objects of type `SccsAnalysis`.

`loadSccsData` *Load the cohort method data from a file*

Description

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

Usage

```
loadSccsData(file)
```

Arguments

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

Value

An object of class SccsData.

`loadSccsIntervalData` *Load the cohort method data from a file*

Description

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

Usage

```
loadSccsIntervalData(file)
```

Arguments

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

Value

An object of class SccsIntervalData.

`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_")

`plotAgeEffect` *Plot the age effect*

Description

Plot the age effect

Usage

```
plotAgeEffect(sccsModel, rrLim = c(0.1, 10), title = NULL, fileName = NULL)
```

Arguments

`sccsModel`	An object of type `sccsModel` as created using the `fitSccsModel` function.
`rrLim`	The limits on the incidence rate ratio scale in the plot.
`title`	Optional: the main title for the plot
`fileName`	Name of the file where the plot should be saved, for example 'plot.png'. See the function `ggsave` in the ggplot2 package for supported file formats.

Details

Plot the spline curve of the age effect.

Value

A Ggplot object. Use the ggsave function to save to file.

`plotAgeSpans` *Plot the age ranges spanned by each observation period.*

Description

Plot the age ranges spanned by each observation period.

Usage

```
plotAgeSpans(
  studyPopulation,
  maxPersons = 10000,
  title = NULL,
  fileName = NULL
)
```

Arguments

`studyPopulation`	An object created using the `createStudyPopulation()` function.
`maxPersons`	The maximum number of persons to plot. If there are more than this number of persons a random sample will be taken to avoid visual clutter.
`title`	Optional: the main title for the plot
`fileName`	Name of the file where the plot should be saved, for example 'plot.png'. See the function `ggplot2::ggsave()` for supported file formats.

Details

Plots a line per patient from their age at observation start to their age at observation end.

Value

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

`plotCalendarTimeEffect` *Plot the calendar time effect*

Description

Plot the calendar time effect

Usage

```
plotCalendarTimeEffect(
  sccsModel,
  rrLim = c(0.1, 10),
  title = NULL,
  fileName = NULL
)
```

Arguments

`sccsModel`	An object of type `sccsModel` as created using the `fitSccsModel` function.
`rrLim`	The limits on the incidence rate ratio scale in the plot.
`title`	Optional: the main title for the plot
`fileName`	Name of the file where the plot should be saved, for example 'plot.png'. See the function `ggsave` in the ggplot2 package for supported file formats.

Details

Plot the spline curve of the calendar time effect.

Value

A Ggplot object. Use the ggsave function to save to file.

`plotCalendarTimeSpans` *Plot the calendar time ranges spanned by each observation period.*

Description

Plot the calendar time ranges spanned by each observation period.

Usage

```
plotCalendarTimeSpans(
  studyPopulation,
  maxPersons = 10000,
  title = NULL,
  fileName = NULL
)
```

Arguments

`studyPopulation`	An object created using the `createStudyPopulation()` function.
`maxPersons`	The maximum number of persons to plot. If there are more than this number of persons a random sample will be taken to avoid visual clutter.
`title`	Optional: the main title for the plot
`fileName`	Name of the file where the plot should be saved, for example 'plot.png'. See the function `ggplot2::ggsave()` for supported file formats.

Details

Plots a line per patient from their observation start to their observation end.

Value

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

`plotEventObservationDependence`

Plot time from event to observation end for censored and uncensored time.

Description

Plot time from event to observation end for censored and uncensored time.

Usage

```
plotEventObservationDependence(studyPopulation, title = NULL, fileName = NULL)
```

Arguments

`studyPopulation`	An object created using the `createStudyPopulation()` function.
`title`	Optional: the main title for the plot
`fileName`	Name of the file where the plot should be saved, for example 'plot.png'. See the function `ggplot2::ggsave()` for supported file formats.

Details

This plot shows whether there is a difference in time between (first) event and the observation period end for periods that are ' censored' and those that are 'uncensored'. By 'censored' we mean periods that end before we would normally expect. Here, we define periods to be uncensored if they end at either the study end date (if specified), database end date (i.e. the date after which no data is captured in the database), or maximum age (if specified). All other periods are assumed to be censored.

As proposed by Farrington et al., by comparing the two plots, we can gain some insight into whether the censoring is dependent on the occurrence of the event.

Value

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

References

Farrington P, Whitaker H, Ghebremichael Weldeselassie Y (2018), Self-controlled case series studies: A modelling guide with R, Taylor & Francis

---

`plotEventToCalendarTime`

Plot the ratio of observed to expected events over calendar time.

Description

Plot the ratio of observed to expected events over calendar time.

Usage

```
plotEventToCalendarTime(
  studyPopulation,
  sccsModel = NULL,
  title = NULL,
  fileName = NULL
)
```

Arguments

`studyPopulation`	An object created using the `createStudyPopulation()` function.
`sccsModel`	Optional: A fitted SCCS model as created using `fitSccsModel()`. If the model contains splines for seasonality and or calendar time a panel will be added with outcome counts adjusted for these splines.
`title`	Optional: the main title for the plot
`fileName`	Name of the file where the plot should be saved, for example 'plot.png'. See the function `ggplot2::ggsave()` for supported file formats.

Details

Plot the ratio of observed to expected events over calendar time. The expected count expected rate considers which persons were observed during that month, and if specified in the model, the adjustment for season and calendar time.

Value

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

`plotExposureCentered` *Plot information centered around the start of exposure*

Description

Plot information centered around the start of exposure

Usage

```
plotExposureCentered(
  studyPopulation,
  sccsData,
  exposureEraId = NULL,
  highlightExposedEvents = TRUE,
  title = NULL,
  fileName = NULL
)
```

Arguments

`studyPopulation`	An object created using the `createStudyPopulation()` function.
`sccsData`	An object of type SccsData as created using the getDbSccsData function.
`exposureEraId`	The exposure to create the era data for. If not specified it is assumed to be the one exposure for which the data was loaded from the database.
`highlightExposedEvents`	Highlight events that occurred during the exposure era using a different color?
`title`	Optional: the main title for the plot
`fileName`	Name of the file where the plot should be saved, for example 'plot.png'. See the function `ggplot2::ggsave()` for supported file formats.

Details

This plot shows the number of events and the number of subjects under observation in week-sized intervals relative to the start of the first exposure.

Value

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

`plotSeasonality` *Plot the seasonality effect*

Description

Plot the seasonality effect

Usage

```
plotSeasonality(sccsModel, rrLim = c(0.1, 10), title = NULL, fileName = NULL)
```

Arguments

`sccsModel`	An object of type `sccsModel` as created using the `fitSccsModel` function.
`rrLim`	The limits on the incidence rate ratio scale in the plot.
`title`	Optional: the main title for the plot
`fileName`	Name of the file where the plot should be saved, for example 'plot.png'. See the function `ggsave` in the ggplot2 package for supported file formats.

Details

Plot the spline curve of the seasonality effect.

Value

A Ggplot object. Use the ggsave function to save to file.

`runSccsAnalyses` *Run a list of analyses*

Description

Run a list of analyses

Usage

```
runSccsAnalyses(
  connectionDetails,
  cdmDatabaseSchema,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  exposureDatabaseSchema = cdmDatabaseSchema,
  exposureTable = "drug_era",
  outcomeDatabaseSchema = cdmDatabaseSchema,
  outcomeTable = "cohort",
  customCovariateDatabaseSchema = cdmDatabaseSchema,
```

```
  customCovariateTable = "cohort",
  nestingCohortDatabaseSchema = cdmDatabaseSchema,
  nestingCohortTable = "cohort",
  outputFolder,
  sccsMultiThreadingSettings = createSccsMultiThreadingSettings(),
  sccsAnalysesSpecifications
)
```

Arguments

`connectionDetails`	An R object of type `ConnectionDetails` created using the function `DatabaseConnector::createConne`
`cdmDatabaseSchema`	The name of the database schema that contains the OMOP CDM instance. Requires read permissions to this database. On SQL Server, this should specify both the database and the schema, so for example 'cdm_instance.dbo'.
`tempEmulationSchema`	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
`exposureDatabaseSchema`	The name of the database schema that is the location where the exposure data used to define the exposure cohorts is available. If `exposureTable` = `"DRUG_ERA"`, `exposureDatabaseSchema` is not used but assumed to be `cdmDatabaseSchema`. Requires read permissions to this database.
`exposureTable`	The table name 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. Requires read permissions to this database.
`outcomeTable`	The table name that contains the outcome cohorts.
`customCovariateDatabaseSchema`	The name of the database schema that is the location where the custom covariate data is available.
`customCovariateTable`	Name of the table holding the custom covariates. This table should have the same structure as the cohort table.
`nestingCohortDatabaseSchema`	The name of the database schema that is the location where the nesting cohort is defined.
`nestingCohortTable`	Name of the table holding the nesting cohort. This table should have the same structure as the cohort table.
`outputFolder`	Name of the folder where all the outputs will written to.
`sccsMultiThreadingSettings`	An object of type `SccsMultiThreadingSettings` as created using the `createSccsMultiThreadingSett` or `createDefaultSccsMultiThreadingSettings()` functions.

`sccsAnalysesSpecifications`	An object of type `SccsAnalysesSpecifications` as created using the `createSccsAnalysesSpecificat` function

Details

Run a list of analyses for the exposures-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(sccsAnalysisList) * length(exposuresOutcomeList)` When you provide several analyses it will determine whether any of the analyses have anything in common, and will take advantage of this fact.

Analyses to Exclude:

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

- exposureId
- outcomeId
- nestingCohortId
- analysisId

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

Value

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

`saveExposuresOutcomeList`

Save a list of `ExposuresOutcome` *to file*

Description

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

Usage

```
saveExposuresOutcomeList(exposuresOutcomeList, file)
```

Arguments

`exposuresOutcomeList`	A list of objects of type `ExposuresOutcome` as created using the `createExposuresOutcome()` function.
`file`	The name of the file where the results will be written

`saveSccsAnalysisList` *Save a list of SccsAnalysis to file*

Description

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

Usage

```
saveSccsAnalysisList(sccsAnalysisList, file)
```

Arguments

`sccsAnalysisList`	A list of objects of type `SccsAnalysis` as created using the `createSccsAnalysis()` function.
`file`	The name of the file where the results will be written

`saveSccsData` *Save the cohort method data to file*

Description

Saves an object of type SccsData to a file.

Usage

```
saveSccsData(sccsData, file)
```

Arguments

`sccsData`	An object of type SccsData as created using the getDbSccsData function.
`file`	The name of the file where the data will be written. If the file already exists it will be overwritten.

Value

Returns no output.

`saveSccsIntervalData` *Save the cohort method data to file*

Description

Saves an object of type SccsIntervalData to a file.

Usage

```
saveSccsIntervalData(sccsIntervalData, file)
```

Arguments

`sccsIntervalData`	An object of type SccsIntervalData as created using the createSccsIntervalData function.
`file`	The name of the file where the data will be written. If the file already exists it will be overwritten.

Value

Returns no output.

`SccsData-class` *SCCS Data*

Description

`SccsData` is an S4 class that inherits from Andromeda. It contains information on the cases and their covariates.

A `SccsData` is typically created using `getDbSccsData()`, can only be saved using `saveSccsData()`, and loaded using `loadSccsData()`.

Usage

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

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

Arguments

`object`	An object of type `SccsData`.

`SccsIntervalData-class` *SCCS Interval Data*

Description

SccsIntervalData‘ is an S4 class that inherits from Andromeda. It contains information on the cases and their covariates, divided in non-overlapping time intervals.

A `SccsIntervalData` is typically created using `createSccsIntervalData()`, can only be saved using `saveSccsIntervalData()`, and loaded using `loadSccsIntervalData()`.

Usage

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

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

Arguments

`object`	An object of type `SccsIntervalData`.

`simulateSccsData` *Simulate SCCS data*

Description

Simulate SCCS data

Usage

```
simulateSccsData(nCases, settings)
```

Arguments

`nCases`	The number of cases to simulate.
`settings`	An object of type `SccsSimulationSettings` as created using the `createSccsSimulationSettings()` function.

Value

An object of type `SccsData`.

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

Description

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

Usage

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

Arguments

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

Index