Package 'SelfControlledCaseSeries'

May 25, 2022

Type Package

Title Self-Controlled Case Series

Version 3.3.0

Date 2022-05-25

Maintainer Martijn Schuemie <schuemie@ohdsi.org>

Description SelfControlledCaseSeries is an R package for performing self-controlled case series (SCCS) analyses in an observational database in the OMOP Common Data Model. It 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.

VignetteBuilder knitr

URL https://github.com/OHDSI/SelfControlledCaseSeries

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

Depends R (>= 3.2.2),
Cyclops (>= 3.1.1),
DatabaseConnector (>= 5.0.0),
Andromeda

Imports SqlRender (>= 1.8.3),
dplyr (>= 1.0.0),
Rcpp (>= 0.11.2),
ParallelLogger (>= 2.0.2),
splines,
ggplot2,
methods,
utils,
cli,
pillar,
checkmate

Suggests testthat,
knitr,
rmarkdown,
EmpiricalCalibration,
Eunomia

Remotes ohdsi/Eunomia

License Apache License 2.0

LinkingTo Rcpp

NeedsCompilation yes

RoxygenNote 7.1.2

Roxygen list(markdown = TRUE)

Encoding UTF-8

R topics documented:

`computeMdrr` *Compute the minimum detectable relative risk*

Description

Compute the minimum detectable relative risk

Usage

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

Arguments

`sccsIntervalData`	An object of type SccsIntervalData as created using the createSccsIntervalData 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

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

Description

Compute 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 ratio between the (adjusted) rate in a month, and the mean (adjusted) rate that we would consider to be irrelevant.
`alpha`	The alpha (type 1 error) used to test for stability. A Bonferroni correction will be applied for the number of months tested.

Details

Computes for each calendar month the rate of the outcome, and evaluates whether that rate is constant over time. If splines are used to adjust for seasonality and/or calendar time, these adjustments are taken into consideration. For each month a two-sided p-value is computed against the null hypothesis that the rate in that month deviates from the mean rate no more than `maxRatio`. This p-value is compared to an alpha value, using a Bonferroni correction to adjust for the multiple testing across months.

Value

A tibble with information on the temporal stability per month. The column `stable` indicates whether the rate of the outcome is within the expected range for that month, assuming the rate is constant over time.

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 cubic 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, and the knots are automatically spaced according to equal percentiles of the data. 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 cubic spline.

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

Value

An object of type `seasonalitySettings`.

---

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 ControlSettings.

---

createCreateSccsIntervalDataArgs

Create a parameter object for the function createSccsIntervalData

---

Description

Create a parameter object for the function createSccsIntervalData

Usage

```
createCreateSccsIntervalDataArgs(
  eraCovariateSettings,
  ageCovariateSettings = NULL,
  seasonalityCovariateSettings = NULL,
  calendarTimeCovariateSettings = NULL,
  minCasesForAgeSeason = NULL,
  minCasesForTimeCovariates = 10000,
  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 createSeasonalityCovariateSettings() function.

`calendarTimeCovariateSettings`
: An object of type calendarTimeCovariateSettings as created using the createCalendarTimeCovariateSettings() function.

`minCasesForAgeSeason`
: DEPRECATED: Use minCasesForTimeCovariates instead.

`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.

`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.

`createCreateScriIntervalDataArgs` *Create a parameter object for the function createScriIntervalData*

Description

Create a parameter object for the function createScriIntervalData

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.

createCreateStudyPopulationArgs

Create a parameter object for the function createStudyPopulation

Description

Create a parameter object for the function createStudyPopulation

Usage

```
createCreateStudyPopulationArgs(
  firstOutcomeOnly = FALSE,
  naivePeriod = 0,
  minAge = NULL,
  maxAge = 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.

Details

Create an object defining the parameter values.

createEraCovariateSettings

Create era covariate settings

Description

Create era covariate settings

Usage

```
createEraCovariateSettings(
  includeEraIds = NULL,
  excludeEraIds = NULL,
  label = "Covariates",
  stratifyById = TRUE,
  start = 0,
  startAnchor = "era start",
  end = 0,
  endAnchor = "era end",
  firstOccurrenceOnly = FALSE,
  splitPoints = c(),
  allowRegularization = FALSE,
  profileLikelihood = 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.
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?
splitPoints	To split the risk window into several smaller windows, specify the end of each sub- window relative to the start of the main risk window. If add ExposedDaysToStart is TRUE, the split points will be considered to be relative to the end of the main risk window instead.
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.

Details

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

Value

An object of type `EraCovariateSettings`.

---

createExposureOutcome *Create a exposure-outcome combination.*

Description

Create a exposure-outcome combination.

Usage

```
createExposureOutcome(exposureId, outcomeId, ...)
```

Arguments

`exposureId`	A concept ID identifying the target drug 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 createSccsAnalysis function.
`outcomeId`	A concept ID identifying the outcome in the outcome table.
`...`	Custom variables, to be used in the analyses.

Details

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

---

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.

---

createGetDbSccsDataArgs

Create a parameter object for the function getDbSccsData

Description

Create a parameter object for the function getDbSccsData

Usage

```
createGetDbSccsDataArgs(
  useCustomCovariates = FALSE,
  useNestingCohort = FALSE,
  nestingCohortId = NULL,
  deleteCovariatesSmallCount = 100,
  studyStartDate = "",
  studyEndDate = "",
  maxCasesPerOutcome = 0,
  exposureIds = "exposureId",
  customCovariateIds = ""
)
```

Arguments

useCustomCovariates	Create covariates from a custom table?
useNestingCohort	Should the study be nested in a cohort (e.g. people with a specific indication)? If not, the study will be nested in the general population.
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.
studyStartDate	A calendar date specifying the minimum date where data is used. Date format is 'yyyymmdd'.

studyEndDate A calendar date specifying the maximum date where data is used. Date format is 'yyyymmdd'.

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 define the exposures of interest. If exposureTable = DRUG_ERA, exposureIds should be CONCEPT_ID. If exposureTable <> DRUG_ERA, exposureIds is used to select the cohort_concept_id in the cohort-like table. If no exposureIds 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.

---

createSccsAnalysis *Create a SelfControlledCaseSeries analysis specification*

Description

Create a SelfControlledCaseSeries analysis specification

Usage

```
createSccsAnalysis(
  analysisId = 1,
  description = "",
  exposureType = NULL,
  outcomeType = NULL,
  getDbSccsDataArgs,
  createStudyPopulationArgs,
  design = "SCCS",
  createSccsIntervalDataArgs = NULL,
  createScriIntervalDataArgs = NULL,
  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.

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.

`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.

`design`
: Either "SCCS" for the general self-controlled case series design, or "SCRI" for the self-controlled risk interval design.

`createSccsIntervalDataArgs`
: An object representing the arguments to be used when calling the createSccsIntervalData function. Ignored when `design = "SCRI"`.

`createScriIntervalDataArgs`
: An object representing the arguments to be used when calling the createScriIntervalData function. Ignored when `design = "SCCS"`.

`fitSccsModelArgs`
: An object representing the arguments to be used when calling the fitSccsModel function.

Details

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

`createSccsIntervalData` *Create SCCS era data*

Description

Create SCCS era data

Usage

```
createSccsIntervalData(
  studyPopulation,
  sccsData,
  eraCovariateSettings,
  ageCovariateSettings = NULL,
  seasonalityCovariateSettings = NULL,
  calendarTimeCovariateSettings = NULL,
  minCasesForAgeSeason = NULL,
  minCasesForTimeCovariates = 10000,
  eventDependentObservation = FALSE
)
```

Arguments

`studyPopulation`
: An object created using the `createStudyPopulation()` function.

`sccsData`
: An object of type SccsData as created using the getDbSccsData function.

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 createSeasonalityCovar function.

calendarTimeCovariateSettings
: An object of type calendarTimeCovariateSettings as created using the createCalendarTimeCov function.

minCasesForAgeSeason
: DEPRECATED: Use minCasesForTimeCovariates instead.

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.

eventDependentObservation
: Should the extension proposed by Farrington et al. be used to adjust for event-dependent observation time?

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.

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

---

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),
  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,
  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.
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.

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.

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,
  eraCovariateSettings,
  controlIntervalSettings
)
```

Arguments

studyPopulation
: An object created using the createStudyPopulation() function.

sccsData
: An object of type SccsData as created using the getDbSccsData function.

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 createControlIntervalSettin 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 cubic spline.

Value

An object of type `seasonalitySettings`.

---

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,
  firstOutcomeOnly = FALSE,
  naivePeriod = 0,
  minAge = NULL,
  maxAge = NULL
)
```

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.
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.

cyclicSplineDesign *Create a design matrix for a cyclic spline*

Description

Create a design matrix for a cyclic spline

Usage

```
cyclicSplineDesign(x, knots, ord = 4)
```

Arguments

x	Vector of coordinates of the points to be interpolated.
knots	Location of the knots.
ord	Order of the spline function.

Details

This function is used by other functions in this package.

---

fitSccsModel *Fit the SCCS model*

Description

Fit the SCCS model

Usage

```
fitSccsModel(
  sccsIntervalData,
  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

sccsIntervalData	An object of type SccsIntervalData as created using the createSccsIntervalData function.
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

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

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.

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.

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,
  oracleTempSchema = NULL,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  outcomeDatabaseSchema = cdmDatabaseSchema,
  outcomeTable = "condition_era",
  outcomeIds,
  exposureDatabaseSchema = cdmDatabaseSchema,
  exposureTable = "drug_era",
  exposureIds = c(),
  useCustomCovariates = FALSE,
  customCovariateDatabaseSchema = cdmDatabaseSchema,
  customCovariateTable = "cohort",
  customCovariateIds = c(),
  useNestingCohort = FALSE,
  nestingCohortDatabaseSchema = cdmDatabaseSchema,
  nestingCohortTable = "cohort",
  nestingCohortId = NULL,
  deleteCovariatesSmallCount = 100,
  studyStartDate = "",
  studyEndDate = "",
  cdmVersion = "5",
  maxCasesPerOutcome = 0
)
```

Arguments

`connectionDetails`
: An R object of type `ConnectionDetails` created using the function `DatabaseConnector::createCo` 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'.

`oracleTempSchema`
: DEPRECATED: use `tempEmulationSchema` instead.

`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 tablename that contains the outcome cohorts. If outcomeTable is not CONDITION_OCCURRENCE or CONDITION_ERA, then expectation is outcomeTable has format of COHORT table: COHORT_DEFINITION_ID, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.

`outcomeIds`
: A list of ids used to define outcomes. If outcomeTable = CONDITION_OCCURRENCE, the list is a set of ancestor CONCEPT_IDs, and all occurrences of all descendant concepts will be selected. If outcomeTable <> CONDITION_OCCURRENCE, the list contains records found in COHORT_DEFINITION_ID field.

`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 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.

`exposureIds`
: A list of identifiers to define the exposures of interest. If exposureTable = DRUG_ERA, exposureIds should be CONCEPT_ID. If exposureTable <> DRUG_ERA, exposureIds is used to select the cohort_concept_id in the cohort-like table. If no exposureIds are provided, all drugs or cohorts in the exposureTable are included as exposures.

`useCustomCovariates`
: Create covariates from a custom table?

`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.

`customCovariateIds`
: A list of cohort definition IDS identifying the records in the customCovariateTable to use for building custom covariates.

`useNestingCohort`
: Should the study be nested in a cohort (e.g. people with a specific indication)? If not, the study will be nested in the general population.

`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.

`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.

`studyStartDate` A calendar date specifying the minimum date where data is used. Date format is 'yyyymmdd'.

`studyEndDate` A calendar date specifying the maximum date where data is used. Date format is 'yyyymmdd'.

`cdmVersion` Define the OMOP CDM version used: currently support "4" and "5".

`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.

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.

Value

An SccsData object.

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

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 an age effect?*

Description

Does the model contain an age 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 an age effect?*

Description

Does the model contain an age 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.

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`.

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.

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 count of events over calendar time.

Description

Plot the count of 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.

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,
  oracleTempSchema = NULL,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  exposureDatabaseSchema = cdmDatabaseSchema,
  exposureTable = "drug_era",
  outcomeDatabaseSchema = cdmDatabaseSchema,
  outcomeTable = "condition_era",
  customCovariateDatabaseSchema = cdmDatabaseSchema,
  customCovariateTable = "cohort",
  nestingCohortDatabaseSchema = cdmDatabaseSchema,
  nestingCohortTable = "cohort",
  cdmVersion = 5,
  outputFolder = "./SccsOutput",
  sccsAnalysisList,
  exposureOutcomeList,
  combineDataFetchAcrossOutcomes = TRUE,
  getDbSccsDataThreads = 1,
  createStudyPopulationThreads = 1,
  createSccsIntervalDataThreads = 1,
  fitSccsModelThreads = 1,
  cvThreads = 1,
  analysesToExclude = NULL
)
```

Arguments

`connectionDetails`
: An R object of type `ConnectionDetails` created using the function `DatabaseConnector::createCo`

`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'.

`oracleTempSchema`
: DEPRECATED: use `tempEmulationSchema` instead.

`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. 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_OCCURRENCE or CONDITION_ERA, then expectation is outcomeTable has format of COHORT table: COHORT_DEFINITION_ID, SUBJECT_ID, COHORT_START_DATE, COHORT_END_DATE.

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.

cdmVersion
: Define the OMOP CDM version used: currently support "4" and "5".

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

sccsAnalysisList
: A list of objects of sccsAnalysis as created using the createSccsAnalysis() function.

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

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.

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.

createSccsIntervalDataThreads
: 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`.

analysesToExclude
Analyses to exclude. See the Analyses to Exclude section for details.

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)` (if all analyses specify an outcome model should be fitted). When you provide several analyses it will determine whether any of the analyses have anything in common, and will take advantage of this fact. For example, if we specify several analyses that only differ in the way the outcome model is fitted, then this function will extract the data and fit the propensity model only once, and re-use this in all the analysis.

Analyses to Exclude:

Normally, `runSccsAnalyses` will run all combinations of exposure-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
- 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`.

---

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

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`	The `sccsAnalysis` list to be written to file
`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 generated using `getDbSccsData()`.
`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 generated using `createSccsIntervalData()`.
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`

Value

An object of type `sccsData`.

`summarizeSccsAnalyses` *Create a summary report of the analyses*

Description

Create a summary report of the analyses

Usage

```
summarizeSccsAnalyses(referenceTable, outputFolder)
```

Arguments

`referenceTable`	A tibble as created by the runSccsAnalyses function.
`outputFolder`	Name of the folder where all the outputs have been written to.

Value

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

Index