Package ‘ClinicalCharacteristics’

July 14, 2025

Title Runs Clinical Characteristics for OHDSI Studies

Version 1.0.0

Description A tool to characterize cohorts using a table shell approach.

License Apache License (>= 2)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports cli,
crayon,
fs,
purrr,
SqlRender,
snakecase,
Capr,
DatabaseConnector,
dplyr,
glue,
readr,
tibble,
tidyr,
here,
lubridate,
reactable,
methods

Additional_repositories `https://OHDSI.github.io/drat`

Suggests knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

`addDefaultEthnicityLineItems` *Convenience function to add default ethnicity line items*

Description

Convenience function to add default ethnicity line items

Usage

```
addDefaultEthnicityLineItems()
```

Value

a list of line items for default ethnicity categories (hispanic, not hispanic, not reported)

`addDefaultGenderLineItems` *Convenience function to add male and female line items for demographic characterization*

Description

Convenience function to add male and female line items for demographic characterization

Usage

```
addDefaultGenderLineItems()
```

Value

a list of two line items for male and female gender

`adherentPresenceStat` *Adherent Presence Stat*

Description

Create a presence stat where only occurrence during the observation period are valid and the denominator are those who only adhere to the observation period

Usage

```
adherentPresenceStat()
```

Value

A presence stat object

`age10yrGrp` *Create a breaks Strategy object for age into 10 year groups*

Description

Create a breaks Strategy object for age into 10 year groups

Usage

```
age10yrGrp()
```

Value

A BreaksStrategy object with defaults assumptions for 10 year age groups

`age5yrGrp` *Create a breaks Strategy object for age into 5 year groups*

Description

Create a breaks Strategy object for age into 5 year groups

Usage

```
age5yrGrp()
```

Value

A BreaksStrategy object with defaults assumptions for 5 year age groups

ageCharBreaks *Create a age statistic with breaks*

Description

Create a age statistic with breaks

Usage

```
ageCharBreaks(breaks)
```

Arguments

`breaks`	a breaksStrategy object dictating how to classify counts into categories

Value

A DemographicAge Statistic class object with breaks

ageCharCts *Create a age statistic as continuous*

Description

Create a age statistic as continuous

Usage

```
ageCharCts()
```

Value

A DemographicAge Statistic class object as continuous

anyCountBreaksStat *Any Count with Breaks*

Description

Create a count stat with breaks where any occurrence is valid.

Usage

```
anyCountBreaksStat(breaks)
```

Arguments

`breaks`	a breaksStrategy object dictating how to classify counts into categories. If null then this defaults to a continuous distribution

Value

A stat object breaks

`anyCountCtsStat` *Any Count Continuous*

Description

Create a count stat where any occurrence is valid.

Usage

```
anyCountCtsStat()
```

Value

A stat object continuousDistribution

`anyPresenceStat` *Any Presence Stat*

Description

Create a presence stat where any occurrence is valid

Usage

```
anyPresenceStat()
```

Value

A presence stat object

`anyScore` *Any Score*

Description

Create score statistic

Usage

```
anyScore(weight)
```

Value

A stat object for a scoreTransformation

`Breaks` *Breaks Statistic*

Description

A statistic that converts a continuous value to a categorical value by grouping the number of events into discrete buckets.

Super class

`ClinicalCharacteristics::Statistic` -> `Breaks`

Methods

Public methods:

- `Breaks$new()`
- `Breaks$clone()`

Method `new()`:

Usage:

```
Breaks$new(personLine, breaks)
```

Arguments:

`personLine` the means of converting occurrences to a single event per patient. For presence this could be any, observed or adherent

`breaks` a breaks strategy object to categorize results

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
Breaks$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`BreaksStrategy` *BreaksStrategy*

Description

An R6 class to define a BreaksStrategy object

Active bindings

`name` the name of the breaks strategy

`type` the type of breaks strategy. Could be 'value' or 'concept'

`labels` A character vector used to label each break interval

`breaks` a vector with cut points

Methods

Public methods:

- `BreaksStrategy$new()`
- `BreaksStrategy$makeCaseWhenSql()`
- `BreaksStrategy$clone()`

Method `new()`:

Usage:

```
BreaksStrategy$new(name, labels, breaks, type)
```

Arguments:

`name` the name of the breaks strategy

`labels` a character vector indicating how to label each break interval

`breaks` a vector with cut points

`type` the type of breaks strategy. Could be 'value' or 'concept'

Method `makeCaseWhenSql()`: Generate SQL code for a CASE WHEN statement based on the break strategy

Usage:

```
BreaksStrategy$makeCaseWhenSql(ordinalId)
```

Arguments:

`ordinalId` the order identifier of the line item in the table shell

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
BreaksStrategy$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`BuildOptions` *BuildOptions*

Description

An R6 class to define build options for the tableShell

Active bindings

`codesetTempTable` table name for codeset table

`sourceCodesetTempTable` table name for source codeset table

`timeWindowTempTable` table name for time windows

`targetCohortTempTable` table name for target cohorts

`tsMetaTempTable` table name for table shell meta

`conceptSetOccurrenceTempTable` table name for concept set occurrence table

`cohortOccurrenceTempTable` table name for cohort occurrence table

`patientLevelDataTempTable` table name for patient level data

`patientLevelTableShellTempTable` table name for patient level data table merged with ts meta

`categoricalSummaryTempTable` table name for categorical summary table

`continuousSummaryTempTable` table name for continuous summary table

`cohortAnalysisType` toggle to choose if using cohort era or start date

Methods

Public methods:

- `BuildOptions$new()`
- `BuildOptions$clone()`

Method `new()`**:**

Usage:

```
BuildOptions$new(
  codesetTempTable = NULL,
  sourceCodesetTempTable = NULL,
  timeWindowTempTable = NULL,
  targetCohortTempTable = NULL,
  tsMetaTempTable = NULL,
  conceptSetOccurrenceTempTable = NULL,
  cohortOccurrenceTempTable = NULL,
  patientLevelDataTempTable = NULL,
  patientLevelTableShellTempTable = NULL,
  categoricalSummaryTempTable = NULL,
  continuousSummaryTempTable = NULL,
  cohortAnalysisType = NULL
)
```

Arguments:

`codesetTempTable` the name of the codeset table used in execution. Defaults as a temp table #codeset

`sourceCodesetTempTable` the name of the source codeset table used in execution

`timeWindowTempTable` the name of the time Window table used in execution. Defaults as a temp table #time_windows

`targetCohortTempTable` the name of the target cohort table used in execution. Defaults as a temp table #target_cohorts

`tsMetaTempTable` the name of the table shell meta table used in execution. Defaults as a temp table #ts_meta

`conceptSetOccurrenceTempTable` the name of the concept set occurrence table used in execution. Defaults as a temp table #concept_set_occ

`cohortOccurrenceTempTable` the name of the cohort occurrence table used in execution. Defaults as a temp table #cohort_occ

`patientLevelDataTempTable` the name of the patient level data table used in execution. Note this does not contain info of the table shell. Defaults as a temp table #patient_data

`patientLevelTableShellTempTable` the name of the patient level data table with additional meta info used in execution. Defaults as a temp table #pat_ts_tab

`categoricalSummaryTempTable` the name of the categorical summary table used in execution. Defaults as a temp table #categorical_table

`continuousSummaryTempTable` the name of the continuous summary table used in execution. Defaults as a temp table #continuous_table

`cohortAnalysisType` a toggle specifying if in a cohort Char whether to use the cohort era ('era') or just the start date ('startDate')

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
BuildOptions$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`cohortFollowupTime` *Create a cohort follow up time characteristic*

Description

Create a cohort follow up time characteristic

Usage

```
cohortFollowupTime()
```

Value

A DemographicCohortTime Statistic class object

`CohortInfo` *CohortInfoe*

Description

An R6 class to define a Cohort Info object. CohortInfo objects do not maintain any execution settings, just the id and name

Methods

Public methods:

- `CohortInfo$new()`
- `CohortInfo$getId()`
- `CohortInfo$getName()`
- `CohortInfo$cohortDetails()`
- `CohortInfo$clone()`

Method `new()`**:**

Usage:

```
CohortInfo$new(id, name)
```

Arguments:

`id` the cohort definition id

`name` the name of the cohort definition

Method `getId()`**:** get the cohort id

Usage:

```
CohortInfo$getId()
```

Method `getName()`**:** get the cohort name

Usage:

```
CohortInfo$getName()
```

Method `cohortDetails()`**:** print the cohort details

Usage:

```
CohortInfo$cohortDetails()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CohortInfo$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CohortLineItem` *CohortLineItem*

Description

An R6 class to define a CohortLineItem

Super class

`ClinicalCharacteristics::LineItem` -> `CohortLineItem`

Methods

Public methods:

- `CohortLineItem$new()`
- `CohortLineItem$clone()`

Method `new()`**:**

Usage:

```
CohortLineItem$new(
  sectionLabel,
  domainTable,
  covariateCohort,
  timeInterval,
  statistic
)
```

Arguments:

`sectionLabel` a label for the table shell section

`domainTable` the domain table in the cdm

`covariateCohort` a CohortInfo class with cohorts for covariates

`timeInterval` a time interval class object to determine the time frame to consider the analytic

`statistic` a Statistic Class object used to determine what type of analytic should be done for the line item

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CohortLineItem$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ConceptSetGroupLineItem` *ConceptSetGroupLineItem*

Description

An R6 class to define a ConceptSetGroupLineItem

Super class

`ClinicalCharacteristics::LineItem` -> `ConceptSetGroupLineItem`

Methods

Public methods:

- `ConceptSetGroupLineItem$new()`
- `ConceptSetGroupLineItem$grabConceptSet()`
- `ConceptSetGroupLineItem$clone()`

Method `new()`**:**

Usage:

```
ConceptSetGroupLineItem$new(
  sectionLabel,
  groupLabel,
  conceptSets,
  domainTables,
  timeInterval,
  statistic
)
```

Arguments:

`sectionLabel` a label for the table shell section

`groupLabel` a label for the group

`conceptSets` a group of concept sets

`domainTables` the domain tables in the cdm

`timeInterval` a time interval class object to determine the time frame to consider the analytic

`statistic` a Statistic Class object used to determine what type of analytic should be done for the line item

Method `grabConceptSet()`**:** retrieve the concept sets

Usage:

```
ConceptSetGroupLineItem$grabConceptSet()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ConceptSetGroupLineItem$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ConceptSetLineItem` *ConceptSetLineItem*

Description

An R6 class to define a ConceptSetLineItem

Super class

`ClinicalCharacteristics::LineItem` -> `ConceptSetLineItem`

Methods

Public methods:

- `ConceptSetLineItem$new()`
- `ConceptSetLineItem$grabConceptSet()`
- `ConceptSetLineItem$clone()`

Method `new()`**:**

Usage:

```
ConceptSetLineItem$new(
  sectionLabel,
  domainTable,
  conceptSet,
  timeInterval,
  statistic
)
```

Arguments:

`sectionLabel` a label for the table shell section

`domainTable` the domain table in the cdm

`conceptSet` a concept set class from Capr

`timeInterval` a time interval class object to determine the time frame to consider the analytic

`statistic` a Statistic Class object used to determine what type of analytic should be done for the line item grabConceptSet

Method `grabConceptSet()`**:** helper to pull concept Capr class items

Usage:

```
ConceptSetLineItem$grabConceptSet()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ConceptSetLineItem$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ContinuousDistribution` *Continuous Distribution Statistic*

Description

A statistic that summarizes the number of occurrences as continuous value using mean, standard deviation and order statistics

Super class

`ClinicalCharacteristics::Statistic` -> `ContinuousDistribution`

Methods

Public methods:

- `ContinuousDistribution$new()`
- `ContinuousDistribution$clone()`

Method `new()`**:**

Usage:

```
ContinuousDistribution$new(personLine)
```

Arguments:

`personLine` the means of converting occurrences to a single event per patient. For presence this could be any, observed or adherent

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ContinuousDistribution$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`createCohortInfo` *Create a CohortInfo object for a cohort and set its attributes*

Description

Create a CohortInfo object for a cohort and set its attributes

Usage

```
createCohortInfo(id, name)
```

Arguments

`id`	The ID of the cohort
`name`	The name of the cohort

Value

A CohortInfo object

`createCohortLineItem` *Create a cohort line item and set its attributes*

Description

Create a cohort line item and set its attributes

Usage

```
createCohortLineItem(
  sectionLabel = NA_character_,
  covariateCohort,
  cohortTable,
  timeInterval,
  statistic
)
```

Arguments

`sectionLabel`	(OPTIONAL) The name of the line item (if not provided, the name will be set to the cohort name from the CohortInfo object)
`timeInterval`	The TimeIntervalClass object used for the line item
`statistic`	The Statistic object to be used to evaluate the line item
`cohort`	A CohortInfo object

Value

A CohortLineItem object

`createCohortLineItemBatch` *Create a batch of cohort line items from a list of CohortInfo objects.*

Description

The name of each line item will be set to the name of its cohort from the CohortInfo object.

Usage

```
createCohortLineItemBatch(
  sectionLabel,
  covariateCohorts,
  cohortTable,
  statistic,
  timeIntervals
)
```

Arguments

`sectionLabel`	The name of the cohort batch
`statistic`	The Statistic object to be used to evaluate the line items
`timeIntervals`	A list of TimeIntervalClass objects
`cohorts`	A list of CohortInfo objects

Value

A list of CohortLineItem objects

`createConceptSetGroupLineItem` *Create a concept set group item and set its attributes*

Description

Create a concept set group item and set its attributes

Usage

```
createConceptSetGroupLineItem(
  sectionLabel = NA_character_,
  groupLabel,
  conceptSets,
  domainTables,
  timeInterval,
  statistic
)
```

Arguments

`sectionLabel`	(OPTIONAL) The name of the line item (if not provided, the name will be the same as the group label)
`groupLabel`	the label of the group
`conceptSets`	A list of Capr concept set object
`domainTables`	a vector of domains corresponding to the concept set
`timeInterval`	The TimeIntervalClass object used for the line item
`statistic`	The Statistic object to be used to evaluate the line item

Value

A CohortLineItem object

createConceptSetLineItem

Create a concept set line item and set its attributes

Description

Create a concept set line item and set its attributes

Usage

```
createConceptSetLineItem(
  sectionLabel = NA_character_,
  domain,
  conceptSet,
  timeInterval,
  statistic
)
```

Arguments

`sectionLabel`	(OPTIONAL) The name of the line item (if not provided, the name will be set to the Capr concept set name)
`domain`	The domain of the concept set (must be one of 'Condition', 'Drug', 'Procedure', 'Observation', 'Measurement', 'Device')
`conceptSet`	The Capr concept set object
`timeInterval`	The Time Interval object used for the line item
`statistic`	The Statistic object to be used to evaluate the line item
`sourceConceptSet`	(OPTIONAL) A Capr concept set of source concept IDs to use to limit the concept set
`typeConceptIds`	(OPTIONAL) A list of type concept IDs to use to limit the concept set
`visitOccurrenceConceptIds`	(OPTIONAL) A list of visit occurrence concept IDs to use to limit the concept set

Value

A ConceptSetLineItem object

`createConceptSetLineItemBatch` *Create a batch of concept set line items from a list of Capr concept sets.*

Description

The name of each line item will be set to the name of its Capr concept set. All line items will use the same statistic, domain, type concepts, and visit concepts. It is not possible to specify source concept IDs.

Usage

```
createConceptSetLineItemBatch(
  sectionLabel,
  domain,
  conceptSets,
  timeIntervals,
  statistic
)
```

Arguments

`sectionLabel`	The name of the concept set batch
`domain`	The domain of the concept sets (must be one of 'Condition', 'Drug', 'Procedure', 'Observation', 'Measurement', 'Device')
`conceptSets`	A list of concept set Capr objects
`timeIntervals`	A list of TimeIntervalClass objects
`statistic`	The Statistic object to be used to evaluate the line items
`typeConceptIds`	(OPTIONAL) A list of type concept IDs to use to limit the concept set
`visitOccurrenceConceptIds`	(OPTIONAL) A list of visit occurrence concept IDs to use to limit the concept set

Value

A list of ConceptSetLineItem objects

`createDemographicLineItem` *Create a demographic line item and set its attributes*

Description

Create a demographic line item and set its attributes

Usage

```
createDemographicLineItem(statistic)
```

Arguments

`statistic`	The Statistic object to be used to evaluate the line item

Value

A DemographicLineItem object

`createExecutionSettings` *Create an ExecutionSettings object and set its attributes*

Description

Create an ExecutionSettings object and set its attributes

Usage

```
createExecutionSettings(
  connectionDetails,
  connection = NULL,
  cdmDatabaseSchema,
  workDatabaseSchema,
  tempEmulationSchema,
  cohortTable,
  cdmSourceName
)
```

Arguments

`connectionDetails`	A DatabaseConnector connectionDetails object (optional if connection is specified)
`connection`	A DatabaseConnector connection object (optional if connectionDetails is specified)
`cdmDatabaseSchema`	The schema of the OMOP CDM database
`workDatabaseSchema`	The schema to which results will be written
`tempEmulationSchema`	Some database platforms like Oracle and Snowflake do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
`cohortTable`	The name of the table where the cohort(s) are stored
`cdmSourceName`	A human-readable name for the OMOP CDM source

Value

An ExecutionSettings object

createSourceConceptSetLineItem

Create a source concept set line item and set its attributes

Description

Create a source concept set line item and set its attributes

Usage

```
createSourceConceptSetLineItem(
  sectionLabel = NA_character_,
  domain,
  sourceConceptSet,
  timeInterval,
  statistic,
  typeConceptIds = c()
)
```

Arguments

`sectionLabel`	(OPTIONAL) The name of the line item (if not provided, the name will be set to the Capr concept set name)
`domain`	The domain of the concept set (must be one of 'Condition', 'Drug', 'Procedure', 'Observation', 'Measurement', 'Device')
`sourceConceptSet`	A SourceConcept R6 object created using the `sourceConceptSet` function
`timeInterval`	The Time Interval object used for the line item
`statistic`	The Statistic object to be used to evaluate the line item
`typeConceptIds`	(OPTIONAL) A list of type concept IDs to use to limit the concept set

Value

A SourceConceptSetLineItem object

createSourceConceptSetLineItemBatch

Create a batch of source concept set line items from a list of SourceConceptSet classes.

Description

Create a batch of source concept set line items from a list of SourceConceptSet classes.

Usage

```
createSourceConceptSetLineItemBatch(
  sectionLabel,
  domain,
  sourceConceptSets,
  timeIntervals,
  statistic,
  typeConceptIds = c()
)
```

Arguments

`sectionLabel`	(OPTIONAL) The name of the line item (if not provided, the name will be set to the Capr concept set name)
`domain`	The domain of the concept set (must be one of 'Condition', 'Drug', 'Procedure', 'Observation', 'Measurement', 'Device')
`timeIntervals`	A list of TimeIntervalClass objects
`statistic`	The Statistic object to be used to evaluate the line item
`typeConceptIds`	(OPTIONAL) A list of type concept IDs to use to limit the concept set
`sourceConceptSet`	A list of SourceConcept R6 object created using the `sourceConceptSet` function

Value

A list of SourceConceptSetLineItem objects

`createTableShell` *Create Table Shell*

Description

Create an empty TableShell object and set its title

Usage

```
createTableShell(title, targetCohorts, lineItems)
```

Arguments

`title`	The title of the TableShell
`targetCohorts`	A list of TargetCohort objects
`lineItems`	A list of lineItem objects

Value

A TableShell object

defaultTableShellBuildOptions

Default build options to generate table shell

Description

Default build options to generate table shell

Usage

```
defaultTableShellBuildOptions(
  codesetTempTable = "#codeset",
  sourceCodesetTempTable = "#source_codeset",
  timeWindowTempTable = "#time_windows",
  targetCohortTempTable = "#target_cohorts",
  tsMetaTempTable = "#ts_meta",
  conceptSetOccurrenceTempTable = "#concept_set_occ",
  cohortOccurrenceTempTable = "#cohort_occ",
  patientLevelDataTempTable = "#patient_data",
  patientLevelTableShellTempTable = "#pat_ts_tab",
  categoricalSummaryTempTable = "#categorical_table",
  continuousSummaryTempTable = "#continuous_table",
  cohortAnalysisType = c("era", "startDate")
)
```

Arguments

`codesetTempTable`	the name of the codeset table used in execution. Defaults as a temp table #codeset
`timeWindowTempTable`	the name of the time Window table used in execution. Defaults as a temp table #time_windows
`targetCohortTempTable`	the name of the target cohort table used in execution. Defaults as a temp table #target_cohorts
`tsMetaTempTable`	the name of the table shell meta table used in execution. Defaults as a temp table #ts_meta
`conceptSetOccurrenceTempTable`	the name of the concept set occurrence table used in execution. Defaults as a temp table #concept_set_occ
`cohortOccurrenceTempTable`	the name of the cohort occurrence table used in execution. Defaults as a temp table #cohort_occ
`patientLevelDataTempTable`	the name of the patient level data table used in execution. Note this does not contain info of the table shell. Defaults as a temp table #patient_data
`patientLevelTableShellTempTable`	the name of the patient level data table with additional meta info used in execution. Defaults as a temp table #pat_ts_tab

`categoricalSummaryTempTable`	the name of the categorical summary table used in execution. Defaults as a temp table #categorical_table
`continuousSummaryTempTable`	the name of the continuous summary table used in execution. Defaults as a temp table #continuous_table
`connectionDetails`	A DatabaseConnector connectionDetails object (optional if connection is specified)
`useCohortEra`	a true false toggle specifying if in a cohort Char whether to use the cohort era (TRUE) or just the start date (FALSE)

Value

A BuildOptions object

`defaultYearGrp` *Create a breaks Strategy object for year*

Description

Create a breaks Strategy object for year

Usage

```
defaultYearGrp(startYear = NULL)
```

Arguments

`startYear`	the year to start the year group sequence. By default this is the year 2000

Value

A BreaksStrategy object with defaults assumptions for 5 year age groups

---

`DemographicAge` *Demographic Age Statistic*

Description

A Demographic Statistic that calculates age from the person table

Super class

`ClinicalCharacteristics::Statistic` -> `DemographicAge`

Methods

Public methods:

- `DemographicAge$new()`
- `DemographicAge$getDemoLabel()`
- `DemographicAge$modifyBreaksLabels()`
- `DemographicAge$clone()`

Method `new()`**:**

Usage:

```
DemographicAge$new(statType, aggType, demoCategory, breaks = NULL)
```

Arguments:

`statType`	the type of statistic
`aggType`	the way the metric is reported either categorical or continuous
`demoCategory`	the name of the demographic category
`breaks`	a breaks strategy object to categorize results

Method `getDemoLabel()`**:** retrieve the demographic label

Usage:

```
DemographicAge$getDemoLabel()
```

Method `modifyBreaksLabels()`**:** update the breaks labels within the statistics class

Usage:

```
DemographicAge$modifyBreaksLabels(newLabels)
```

Arguments:

`newLabels`	a character string of new labels for the breaks

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DemographicAge$clone(deep = FALSE)
```

Arguments:

`deep`	Whether to make a deep clone.

`DemographicCohortTime` *Demographic Cohort Time Statistic*

Description

A Demographic Statistic that calculates the time (in days) in the target cohort

Super class

`ClinicalCharacteristics::Statistic` -> `DemographicCohortTime`

Methods

Public methods:

- `DemographicCohortTime$new()`
- `DemographicCohortTime$getDemoLabel()`
- `DemographicCohortTime$clone()`

Method `new()`**:** initialize cohort time stat

Usage:

```
DemographicCohortTime$new()
```

Method `getDemoLabel()`**:** retrieve the demographic label

Usage:

```
DemographicCohortTime$getDemoLabel()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DemographicCohortTime$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DemographicConcept` *Demographic Concept Statistic*

Description

A Demographic Statistic that considers concepts in person table

Super class

`ClinicalCharacteristics::Statistic` -> `DemographicConcept`

Methods

Public methods:

- `DemographicConcept$new()`
- `DemographicConcept$getConceptColumn()`
- `DemographicConcept$getDemoLabel()`
- `DemographicConcept$getConceptId()`
- `DemographicConcept$clone()`

Method `new()`**:**

Usage:

```
DemographicConcept$new(demoCategory, demoLine, conceptColumn, conceptId)
```

Arguments:

`demoCategory` the category name of the demographic

`demoLine` the line item name of the demographic concept

`conceptColumn` the name of column in the person table to extract demographic concept `conceptId` the concept to search for in the concept column

Method `getConceptColumn()`**:** retrieve the concept column

Usage:

```
DemographicConcept$getConceptColumn()
```

Method `getDemoLabel()`**:** create a label for the demographic concept

Usage:

```
DemographicConcept$getDemoLabel()
```

Method `getConceptId()`**:** retrieve the concept id

Usage:

```
DemographicConcept$getConceptId()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DemographicConcept$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DemographicIndexYear` *Demographic Index Year Statistic*

Description

A Demographic Statistic that retrieves the index year for each patient

Super class

`ClinicalCharacteristics::Statistic` -> `DemographicIndexYear`

Methods

Public methods:

- `DemographicIndexYear$new()`
- `DemographicIndexYear$getDemoLabel()`
- `DemographicIndexYear$modifyBreaksLabels()`
- `DemographicIndexYear$clone()`

Method `new()`**:**

Usage:

```
DemographicIndexYear$new(breaks)
```

Arguments:

`breaks` a breaks strategy object to categorize results

Method `getDemoLabel()`**:** retrieve the demographic label

Usage:

```
DemographicIndexYear$getDemoLabel()
```

Method `modifyBreaksLabels()`**:** update the breaks labels within the statistics class

Usage:

```
DemographicIndexYear$modifyBreaksLabels(newLabels)
```

Arguments:

`newLabels` a character string of new labels for the breaks

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DemographicIndexYear$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DemographicLineItem` *DemographicLineItem*

Description

An R6 class to handle a Demographic line item

Super class

`ClinicalCharacteristics::LineItem` -> `DemographicLineItem`

Methods

Public methods:

- `DemographicLineItem$new()`
- `DemographicLineItem$clone()`

Method `new()`**:**

Usage:

```
DemographicLineItem$new(statistic = statistic)
```

Arguments:

`statistic` a Statistic Class object used to determine what type of analytic should be done for the line item

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DemographicLineItem$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DemographicLocation` *Demographic Location Statistic*

Description

A Demographic Statistic that retrieves and categorizes the location of the persons in the target cohort

Super class

`ClinicalCharacteristics::Statistic` -> `DemographicLocation`

Methods

Public methods:

- `DemographicLocation$new()`
- `DemographicLocation$getDemoLabel()`
- `DemographicLocation$modifyBreaksLabels()`
- `DemographicLocation$clone()`

Method `new()`**:**

Usage:

```
DemographicLocation$new(breaks)
```

Arguments:

`breaks` a breaks strategy object to categorize results

Method `getDemoLabel()`**:** retrieve the demographic label

Usage:

```
DemographicLocation$getDemoLabel()
```

Method `modifyBreaksLabels()`**:** update the breaks labels within the statistics class

Usage:

```
DemographicLocation$modifyBreaksLabels(newLabels)
```

Arguments:

`newLabels` a character string of new labels for the breaks

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DemographicLocation$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DemographicPayerType` *Demographic Payer Statistic*

Description

A Demographic Statistic that retrieves and categorizes the payer type from the payer plan period table

Super class

`ClinicalCharacteristics::Statistic` -> `DemographicPayerType`

Methods

Public methods:

- `DemographicPayerType$new()`
- `DemographicPayerType$getDemoLabel()`
- `DemographicPayerType$modifyBreaksLabels()`
- `DemographicPayerType$clone()`

Method `new()`**:**

Usage:

```
DemographicPayerType$new(breaks)
```

Arguments:

`breaks` a breaks strategy object to categorize results

Method `getDemoLabel()`**:** retrieve the demographic label

Usage:

```
DemographicPayerType$getDemoLabel()
```

Method `modifyBreaksLabels()`**:** update the breaks labels within the statistics class

Usage:

```
DemographicPayerType$modifyBreaksLabels(newLabels)
```

Arguments:

`newLabels` a character string of new labels for the breaks

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DemographicPayerType$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DemographicRace` *Demographic Race Statistic*

Description

A Demographic Statistic that retrieves and categorizes the patient race from the person table

Super class

`ClinicalCharacteristics::Statistic` -> `DemographicRace`

Methods

Public methods:

- `DemographicRace$new()`
- `DemographicRace$getDemoLabel()`
- `DemographicRace$modifyBreaksLabels()`
- `DemographicRace$clone()`

Method `new()`**:**

Usage:

```
DemographicRace$new(breaks)
```

Arguments:

`breaks`	a breaks strategy object to categorize results

Method `getDemoLabel()`**:** retrieve the demographic label

Usage:

```
DemographicRace$getDemoLabel()
```

Method `modifyBreaksLabels()`**:** update the breaks labels within the statistics class

Usage:

```
DemographicRace$modifyBreaksLabels(newLabels)
```

Arguments:

`newLabels`	a character string of new labels for the breaks

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DemographicRace$clone(deep = FALSE)
```

Arguments:

`deep`	Whether to make a deep clone.

`ExecutionSettings` *ExecutionSettings*

Description

An R6 class to define an ExecutionSettings object

Active bindings

`cdmDatabaseSchema` the schema containing the OMOP CDM

`workDatabaseSchema` the schema containing the cohort table

`tempEmulationSchema` the schema needed for temp tables

`cohortTable` the table containing the cohorts

`cdmSourceName` the name of the source data of the cdm

Methods

Public methods:

- `ExecutionSettings$new()`
- `ExecutionSettings$getDbms()`
- `ExecutionSettings$connect()`
- `ExecutionSettings$disconnect()`
- `ExecutionSettings$getConnection()`
- `ExecutionSettings$clone()`

Method `new()`:

Usage:

```
ExecutionSettings$new(
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema = NULL,
  workDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  cohortTable = NULL,
  cdmSourceName = NULL
)
```

Arguments:

`connectionDetails` a connectionDetails object

`connection` a connection to a dbms

`cdmDatabaseSchema` The schema of the OMOP CDM database

`workDatabaseSchema` The schema to which results will be written

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

`cohortTable` The name of the table where the cohort(s) are stored

`cdmSourceName` A human-readable name for the OMOP CDM source

Method `getDbms()`**:** extract the dbms dialect

Usage:

```
ExecutionSettings$getDbms()
```

Method `connect()`**:** connect to dbms

Usage:

```
ExecutionSettings$connect()
```

Method `disconnect()`**:** disconnect from dbms

Usage:

```
ExecutionSettings$disconnect()
```

Method `getConnection()`**:** retrieve the connection object

Usage:

```
ExecutionSettings$getConnection()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ExecutionSettings$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`femaleGender` *Create a female concept stat*

Description

Create a female concept stat

Usage

```
femaleGender()
```

Value

A DemographicConcept Statistic class object indicating a female concept

`generateTableShell` *Function to generate results for the table shell object*

Description

Function to generate results for the table shell object

Usage

```
generateTableShell(tableShell, executionSettings, buildOptions = NULL)
```

Arguments

`tableShell`	The TableShell object to used for generation
`executionSettings`	The ExecutionSettings object used to generate table shell
`buildOptions`	The BuildOptions object used to generate table shell

Value

A list containing a tibble for categorical and continuous results

`indexYear` *Create an index year characteristic*

Description

Create an index year characteristic

Usage

```
indexYear(breaks = NULL)
```

Arguments

`breaks`	a breaksStrategy object dictating how to classify years into categories. By default this will do each year from 2000 to current day.

Value

A DemographicIndexYear Statistic class object

`IntervalRate` *Interval Rate Statistic*

Description

A statistic that calculates the rate of occurrence by taking the number of events per person in the desired interval and dividing by the observed time during the interval. An interval rate can either be monthly or yearly.

Super class

`ClinicalCharacteristics::Statistic` -> `IntervalRate`

Methods

Public methods:

- `IntervalRate$new()`
- `IntervalRate$clone()`

Method `new()`**:**

Usage:

```
IntervalRate$new(interval)
```

Arguments:

`interval` the type of interval to use for the rate. can be either monthly or yearly.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IntervalRate$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`LineItem` *LineItem*

Description

An R6 class to define a LineItem object. A LineItem is a single, explicitly defined characterization to appear in a Section. Derived classes exist off of LineItems

Active bindings

`ordinalId` the order identifier of the line item in the table shell

`sectionLabel` a label for the table shell section

`lineItemLabel` a label for the line item

`valueId` the id for the line item; either a codeset id, a concept id or a -999 to indicate no true id

`valueDescription` the describer for the value id

`domainTable` the domain table in the cdm

`lineItemClass` the type of line item (ie Demographic, ConceptSet, SourceConceptSet, ConceptSetGroup, Cohort)

Methods

Public methods:

- `LineItem$new()`
- `LineItem$getLineItemMeta()`
- `LineItem$getStatistic()`
- `LineItem$clone()`

Method `new()`**:**

Usage:

```
LineItem$new(
  sectionLabel,
  lineItemLabel = NA_character_,
  domainTable,
  lineItemClass,
  valueId = NA_integer_,
  valueDescription = NA_integer_,
  statistic,
  timeInterval = NULL
)
```

Arguments:

`sectionLabel` a label for the table shell section

`lineItemLabel` a label for the line item

`domainTable` the domain table in the cdm

`lineItemClass` the type of line item (ie Demographic, ConceptSet, SourceConceptSet, ConceptSetGroup, Cohort)

`valueId` the id for the line item; either a codeset id, a concept id or a -999 to indicate no true id

`valueDescription` the describer for the value id

`statistic` a Statistic Class object used to determine what type of analytic should be done for the line item

`timeInterval` a time interval class object to determine the time frame to consider the analytic

Method `getLineItemMeta()`**:** retrieve the line item meta information

Usage:

```
LineItem$getLineItemMeta()
```

Method `getStatistic()`**:** retrieve the statistic class object

Usage:

```
LineItem$getStatistic()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
LineItem$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`lineItems` *Combine all lineItems to enter into the tableShell slot*

Description

Combine all lineItems to enter into the tableShell slot

Usage

```
lineItems(...)
```

Arguments

`...`	A list of lineItems created from various calls

Value

a flattened list of lineItems

`lookupSourceConcepts` *Function to look up source concepts in the OMOP Vocabulary*

Description

Function to look up source concepts in the OMOP Vocabulary

Usage

```
lookupSourceConcepts(codes, vocabulary, executionSettings)
```

Arguments

`codes`	a character string of codes to search
`vocabulary`	the vocabulary to use in search of codes
`executionSettings`	The ExecutionSettings object used to connect to the dbms

Value

a tibble of four columns: conceptId, conceptName, conceptCode, vocabularyId

`maleGender` *Create a male concept stat*

Description

Create a male concept stat

Usage

```
maleGender()
```

Value

A DemographicConcept Statistic class object indicating a male concept

`monthlyRate` *Create a monthly interval rate statistic*

Description

This statistic sums the number of occurrences of an event in a timeInterval and divides it by the time (modified by month) to construct a rate per patient. This can then be summarized as a continuous variable

Usage

```
monthlyRate()
```

Value

A stat object of class intervalRate

`newConceptBreaks` *Create a breaks Strategy object for categorizing concepts*

Description

Create a breaks Strategy object for categorizing concepts

Usage

```
newConceptBreaks(name, breaks, labels)
```

Arguments

`name`	the name of the breaks
`breaks`	a vector with cut points to user
`labels`	a character vector indicating how to label the cut-point. Can stay NULL where a default label is given

Value

A BreaksStrategy object of type concept

`newValueBreaks` *Create a breaks Strategy object for categorizing value*

Description

Create a breaks Strategy object for categorizing value

Usage

```
newValueBreaks(name, breaks, labels = NULL)
```

Arguments

`name`	the name of the breaks
`breaks`	a vector with cut points to user
`labels`	a character vector indicating how to label the cut-point. Can stay NULL where a default label is given

Value

A BreaksStrategy object of type value

`observedCountBreaksStat` *Observed Count with Breaks*

Description

Create a count stat with breaks where only occurrence during the observation period are valid

Usage

```
observedCountBreaksStat(breaks)
```

Arguments

`breaks`	a breaksStrategy object dictating how to classify counts into categories. If null then this defaults to a continuous distribution

Value

A stat object breaks

`observedCountCtsStat` *Observed Count Continuous*

Description

Create a count stat where only occurrence during the observation period are valid

Usage

```
observedCountCtsStat()
```

Value

A stat object continuousDistribution

`observedPresenceStat` *Observed Presence Stat*

Description

Create a presence stat where only occurrence during the observation period are valid

Usage

```
observedPresenceStat()
```

Value

A presence stat object

`parseCohortInfoFromDf` *Parse cohort info from a data frame*

Description

Parse cohort info from a data frame

Usage

```
parseCohortInfoFromDf(df)
```

Arguments

`df`	The data frame containing the information for the cohorts (id and name)

Value

A list of CohortInfo objects

`payerType` *Create a payer type characteristic*

Description

Create a payer type characteristic

Usage

```
payerType(breaks = NULL)
```

Arguments

`breaks`	a breaksStrategy object dictating how to classify payer types into categories. by default this will use the Source of Payment Typology(SOPT) vocabulary

Value

A DemographicPayerType Statistic class object

`personLocation` *Create a location characteristic*

Description

Create a location characteristic

Usage

```
personLocation(breaks)
```

Arguments

`breaks`	a breaksStrategy object dictating how to classify locations into categories.

Value

A DemographicLocation Statistic class object

`Presence` *Presence Statistic*

Description

A statistic that determines whether at least one clinical event was present during the specified time interval. It is summarized as a categorical value.

Super class

`ClinicalCharacteristics::Statistic` -> `Presence`

Methods

Public methods:

- `Presence$new()`
- `Presence$clone()`

Method `new()`**:**

Usage:

```
Presence$new(personLine)
```

Arguments:

`personLine` the means of converting occurrences to a single event per patient. For presence this could be any, observed or adherent

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Presence$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`quanCharlsonComorbidityScore` *Convenience function to add quan charlson comorbidity score*

Description

The Quan Charlson Comorbidity score is a measure for predicting 10 year survival. It is a modification to the Charlson Score by Quan et al (doi: 10.1097/01.mlr.0000182534.19832.83). The method presented in this packages follows the SNOMED adaption of Quan Charlson tested on OMOP CDM by Fortin et al (doi: 10.1186/s12911-022-02006-1). This function will add the elements needed for each comorbidity line item and the appropriate weights needed to convert the categorization of comorbidities into a score.

Usage

```
quanCharlsonComorbidityScore(timeWindow = NULL)
```

Arguments

`timeWindow`	the interval to assess the comorbidity score, by default baseline it -365 to -1 days

Value

a list of line items for running quan charlson comorbidity score. This will determine the proportion of persons with each comorbidity and the overall score per patient in the cohort

---

`raceCategory` *Create a race characteristic*

Description

Create a race characteristic

Usage

```
raceCategory(breaks = NULL)
```

Arguments

`breaks`	a breaksStrategy object dictating how to classify race into categories. by default this will use custom race categories

Value

A DemographicRace Statistic class object

`reviewTableShellSql` *Function that previews sql script used to generate results for table shell*

Description

Function that previews sql script used to generate results for table shell

Usage

```
reviewTableShellSql(
  tableShell,
  executionSettings,
  buildOptions = NULL,
  saveName = NULL,
  savePath = here::here()
)
```

Arguments

`tableShell`	The TableShell object to used for generation
`executionSettings`	The ExecutionSettings object used to generate table shell
`buildOptions`	The BuildOptions object used to generate table shell
`saveName`	The name of the table shell sql file
`savePath`	the folder location to save the file

Value

A sql file written to a specific location

`saveTableShellResults` *Function that previews sql script used to generate results for table shell*

Description

Function that previews sql script used to generate results for table shell

Usage

```
saveTableShellResults(result, saveName, savePath = here::here())
```

Arguments

`result`	the list output from `generateTableShell` containing a categorical and continuous tibble
`saveName`	The save name of the csv files
`savePath`	the folder location to save the csv files

Value

A sql file written to a specific location

`Score` *Score Statistic*

Description

A statistic that converts a categorical value to a continuous value by modifying the occurrence of an event by a weight and summing across patients.

Super class

`ClinicalCharacteristics::Statistic` -> `Score`

Active bindings

`weight` a numeric value to modify the value of an occurrence

Methods

Public methods:

- `Score$new()`
- `Score$clone()`

Method `new()`**:**

Usage:

```
Score$new(personLine, weight)
```

Arguments:

`personLine` the means of converting occurrences to a single event per patient. For a score currently only enabled for any occurrence

`weight` a numeric value to modify the value of an occurrence

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Score$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`sourceConceptSet` *Function to create a source concept set*

Description

Function to create a source concept set

Usage

```
sourceConceptSet(sourceConceptTable, name)
```

Arguments

`sourceConceptTable`	a dataframe with source concepts from the OMOP vocabulary
`name`	the name of source concept set

Value

a SourceConceptSet R6 class specifing the source concepts in use

SourceConceptSetLineItem *SourceConceptSetLineItem*

Description

An R6 class to define a SourceConceptSetLineItem

Super class

`ClinicalCharacteristics::LineItem` -> `SourceConceptSetLineItem`

Methods

Public methods:

- `SourceConceptSetLineItem$new()`
- `SourceConceptSetLineItem$grabSourceConceptSet()`
- `SourceConceptSetLineItem$clone()`

Method `new()`:

Usage:

```
SourceConceptSetLineItem$new(
  sectionLabel,
  domainTable,
  sourceConceptSet,
  timeInterval,
  statistic
)
```

Arguments:

`sectionLabel` a label for the table shell section

`domainTable` the domain table in the cdm

`sourceConceptSet` a source concept Set

`timeInterval` a time interval class object to determine the time frame to consider the analytic

`statistic` a Statistic Class object used to determine what type of analytic should be done for the line item

Method `grabSourceConceptSet()`: retrieve the source concept set

Usage:

```
SourceConceptSetLineItem$grabSourceConceptSet()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
SourceConceptSetLineItem$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Statistic` *An R6 class to define a Statistic object*

Description

A Statistic is a type of metric to be used for characterization. Specific types of statistics are defined in derived classes

Methods

Public methods:

- `Statistic$new()`
- `Statistic$getStatisticType()`
- `Statistic$getAggregationType()`
- `Statistic$getPersonLineTransformation()`
- `Statistic$getBreaksIfAny()`
- `Statistic$getWeightsIfAny()`
- `Statistic$clone()`

Method `new()`**:**

Usage:

```
Statistic$new(statType, personLine, aggType)
```

Arguments:

`statType` the type of statistic

`personLine` the means of converting occurrences to a single event per patient

`aggType` the way the metric is reported either categorical or continuous

Method `getStatisticType()`**:** retrieve the statistic type

Usage:

```
Statistic$getStatisticType()
```

Method `getAggregationType()`**:** retrieve the aggregation type

Usage:

```
Statistic$getAggregationType()
```

Method `getPersonLineTransformation()`**:** retrieve the person line transformation

Usage:

```
Statistic$getPersonLineTransformation()
```

Method `getBreaksIfAny()`**:** retrieve the breaks object from the statistic object

Usage:

```
Statistic$getBreaksIfAny()
```

Method `getWeightsIfAny()`**:** retrieve the weights object from the statistic object

Usage:

```
Statistic$getWeightsIfAny()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Statistic$clone(deep = FALSE)
```

Arguments:

`deep`	Whether to make a deep clone.

`TableShell` *Table Shell*

Description

An R6 class to define a TableShell object

Methods

Public methods:

- `TableShell$new()`
- `TableShell$getTitle()`
- `TableShell$getTableShellMeta()`
- `TableShell$getTargetCohorts()`
- `TableShell$getLineItems()`
- `TableShell$printJobDetails()`
- `TableShell$buildTableShellSql()`
- `TableShell$outputResults()`
- `TableShell$dropTempTables()`
- `TableShell$clone()`

Method `new()`**:**

Usage:

```
TableShell$new(title, targetCohorts, lineItems)
```

Arguments:

`title`	the title of the table shell
`targetCohorts`	a list of CohortInfo class objects that describe the index cohorts
`lineItems`	a list of line item class objects

Method `getTitle()`**:** get the title of the table shell

Usage:

```
TableShell$getTitle()
```

Method `getTableShellMeta()`**:** get the meta information for the table shell build

Usage:

```
TableShell$getTableShellMeta()
```

Method `getTargetCohorts()`**:** get the target cohorts from the table shell

Usage:

```
TableShell$getTargetCohorts()
```

Method `getLineItems()`**:** get the lineItems from the table shell

Usage:

```
TableShell$getLineItems()
```

Method `printJobDetails()`**:** print the job details of the table shell

Usage:

```
TableShell$printJobDetails()
```

Method `buildTableShellSql()`**:** function creates the table shell sql needed for the execution

Usage:

```
TableShell$buildTableShellSql(executionSettings, buildOptions)
```

Arguments:

`executionSettings` an executionSettings class obj

`buildOptions` a buildOptions class obj

Method `outputResults()`**:** retrieves results from dbms and formats for review

Usage:

```
TableShell$outputResults(executionSettings, buildOptions)
```

Arguments:

`executionSettings` an executionSettings class obj

`buildOptions` a buildOptions class obj

Method `dropTempTables()`**:** drop all temp tables from the tableShell build

Usage:

```
TableShell$dropTempTables(executionSettings, buildOptions)
```

Arguments:

`executionSettings` an executionSettings class obj

`buildOptions` a buildOptions class obj

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TableShell$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`timeInterval` *Create a single time interval*

Description

Create a single time interval

Usage

```
timeInterval(lb, rb)
```

Arguments

`lb`	the left bound of the time interval
`rb`	the right bound of the time interval

Value

A time interval object

`TimeIntervalClass` *TimeIntervalClass*

Description

An R6 class to define a TimeIntervalClass

Methods

Public methods:

- `TimeIntervalClass$new()`
- `TimeIntervalClass$getLb()`
- `TimeIntervalClass$getRb()`
- `TimeIntervalClass$getTimeLabel()`
- `TimeIntervalClass$getTimeInterval()`
- `TimeIntervalClass$clone()`

Method `new()`**:**

Usage:

```
TimeIntervalClass$new(lb, rb)
```

Arguments:

`lb` left bound - the start of the time interval

`rb` right bound - the end of the time interval

Method `getLb()`**:** return the left bound

Usage:

```
TimeIntervalClass$getLb()
```

Method `getRb()`**:** return the right bound

Usage:

```
TimeIntervalClass$getRb()
```

Method `getTimeLabel()`**:** create and return time labels for left and right bounds

Usage:

```
TimeIntervalClass$getTimeLabel()
```

Method `getTimeInterval()`**:** return a tibble with the left and right bounds

Usage:

```
TimeIntervalClass$getTimeInterval()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TimeIntervalClass$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`timeToFirst` *Time To First*

Description

Create a time to stat where any occurrence is valid

Usage

```
timeToFirst()
```

Value

A stat object continuousDistribution

`yearlyRate` *Create a yearly interval rate statistic*

Description

This statistic sums the number of occurrences of an event in a timeInterval and divides it by the time (modified by year) to construct a rate per patient. This can then be summarized as a continuous variable

Usage

```
yearlyRate()
```

Value

A stat object of class intervalRate

Index