Package 'Capr'

January 19, 2022

Title Cohort definition Application Programming in R

Version 1.0.1

Description The CAPR package develops cohort definitions to implement across an OMOP mapped dbms. This pacakge allows allows for the programmatic creation of OMOP cohorts that compile to the CIRCE-BE engine. CAPR utilizes s4 to construct component parts to the cohort definition (i.e. Primary Criteria, Inclusion Rules, Additional Criteria, Censoring Criteria, and End Strategy) and then packs them together into a Cohort Definition class. The Cohort Definition can be rendered into a CIRCE-BE object that will generate ohdsiSQL to query against an OMOP dbms. CAPR adds component parts to the OMOP cohort definition in order to combine Concept Set Expressions with its definition logic in the same position, facilitating the transition between scientific description and computational implmentation.

License Apache License 2.0

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Depends R (>= 3.5.0),
CirceR (>= 1.0.0),
DatabaseConnector (>= 2.4.2),
magrittr (>= 1.5.0)

Imports jsonlite,
RJSONIO,
methods,
purrr,
rlang,
uuid,
SqlRender,
dplyr,
checkmate,
tibble,
withr,
readr,
utils,
stringr,
glue,

cli

Suggests testthat (>= 3.0.0),
knitr,
rmarkdown

Remotes ohdsi/CirceR

Collate 'Capr.R'
'LowLevelClasses.R'
'LowLevelUtilityFn.R'
'LowLevelBuildLangFn.R'
'LowLevelCoercionFn.R'
'LowLevelCreateFn.R'
'LowLevelLoadFn.R'
'LowLevelSaveFn.R'
'UserAttributeEdit.R'
'UserCommands.R'
'UserConceptLookupFn.R'
'UserCreateAttributeFn.R'
'UserCreateDomainFn.R'
'UserCreateFn.R'
'UserEditFn.R'
'utils.R'

VignetteBuilder knitr

Config/testthat/edition 3

R topics documented:

`addAttributeToQuery` *Function to add Attribute to Query*

Description

This function edits a expression type class

Usage

```
addAttributeToQuery(query, attribute)
```

Arguments

`query`	identify the query object to edit
`attribute`	the attribute to add to the query

Value

the edited query component

`as.AttributeLoad` *A coercion function to convert to a CAPR attribute*

Description

This function takes a saved CAPR attribute json and returns an attribute CAPR R object

Usage

```
as.AttributeLoad(x)
```

Arguments

`x`	the object to coerce

Value

a attribute class object

`as.Circe,Window-method`

Coersive function from S4 to S3

Description

To serialize between json and R, an S3 list object is required. CAPR creates an organized s4 object that maintains components of the cohort definition. CIRCE needs to be in an S3 structure in R before serializing to json. These functions maintain consistency between the s3 and s4 data structures

Usage

```
## S4 method for signature 'Window'
as.Circe(x)

## S4 method for signature 'Timeline'
as.Circe(x)

## S4 method for signature 'Occurrence'
as.Circe(x)

## S4 method for signature 'ObservationWindow'
as.Circe(x)

## S4 method for signature 'Limit'
as.Circe(x)

## S4 method for signature 'ExpressionType'
as.Circe(x)

## S4 method for signature 'Concept'
as.Circe(x)

## S4 method for signature 'ConceptSetItem'
as.Circe(x)

## S4 method for signature 'ConceptSetExpression'
as.Circe(x)

## S4 method for signature 'OpAttribute'
as.Circe(x)

## S4 method for signature 'SourceConceptAttribute'
as.Circe(x)

## S4 method for signature 'ConceptAttribute'
as.Circe(x)
```

```
## S4 method for signature 'LogicAttribute'
as.Circe(x)

## S4 method for signature 'CorrelatedCriteriaAttribute'
as.Circe(x)

## S4 method for signature 'Query'
as.Circe(x)

## S4 method for signature 'Count'
as.Circe(x)

## S4 method for signature 'Group'
as.Circe(x)

## S4 method for signature 'DateOffsetEndStrategy'
as.Circe(x)

## S4 method for signature 'CustomEraEndStrategy'
as.Circe(x)

## S4 method for signature 'CollapseSettings'
as.Circe(x)

## S4 method for signature 'CensorWindow'
as.Circe(x)

## S4 method for signature 'Component'
as.Circe(x)
```

Arguments

x	a component class object in s4

Value

the object converted back to s3 that can be used for json seralization

---

`as.CohortEra` *A coercion function to convert to a CAPR CohortEra*

Description

A coercion function to convert to a CAPR CohortEra

Usage

```
as.CohortEra(x)
```

Arguments

x	the object to coerce

Value

a cohortEra class object

---

as.ComponentLoad *A coercion function to convert to a CAPR component*

Description

This function takes a saved CAPR component json and returns component CAPR R object

Usage

```
as.ComponentLoad(x)
```

Arguments

x	the object to coerce

Value

a component class object

---

as.Concept *A coercion function to convert to a CAPR concept*

Description

This function takes a data frame containing information about a concept and converts it into the Concept class

Usage

```
as.Concept(x)
```

Arguments

x	the object to coerce

Value

a concept class object

as.ConceptLoad *A coercion function to load to a CAPR concept*

Description

This function takes a data frame containing information about a concept and converts it into the Concept class

Usage

```
as.ConceptLoad(x)
```

Arguments

x	the object to coerce

Value

a concept class object

as.ConceptSetExpression *A coercion function to convert to a CAPR conceptSetExpression*

Description

A coercion function to convert to a CAPR conceptSetExpression

Usage

```
as.ConceptSetExpression(x)
```

Arguments

x	the object to coerce

Value

a concept set expression class object

as.ConceptSetItem *A coercion function to convert to a CAPR conceptSetItem*

Description

This function takes a list and converts it into the Concept set Item class

Usage

```
as.ConceptSetItem(x)
```

Arguments

x	the object to coerce

Value

a conceptSetItem class object

as.CountLoad *A coercion function to convert to a CAPR count*

Description

This function takes a saved CAPR count json and returns count CAPR R object

Usage

```
as.CountLoad(x)
```

Arguments

x	the object to coerce

Value

a count class object

as.EndStrategyLoad *A coercion function to convert to a CAPR EndStrategy*

Description

This function takes a saved CAPR EndStrategy json and returns EndStrategy CAPR R object

Usage

```
as.EndStrategyLoad(x)
```

Arguments

x	the object to coerce

Value

a EndStrategy class object

as.ExpressionType *A coercion function to convert to a CAPR expression type*

Description

A coercion function to convert to a CAPR expression type

Usage

```
as.ExpressionType(x)
```

Arguments

x	the object to coerce

Value

an expressionType class object

as.GroupLoad *A coercion function to convert to a CAPR group*

Description

This function takes a saved CAPR group json and returns group CAPR R object

Usage

```
as.GroupLoad(x)
```

Arguments

x	the object to coerce

Value

a group class object

as.Limit *A coercion function to convert to a CAPR limit*

Description

A coercion function to convert to a CAPR limit

Usage

```
as.Limit(x)
```

Arguments

x	the object to coerce

Value

a limit class object

as.MetaData *A coercion function to convert to a CAPR metaData*

Description

A coercion function to convert to a CAPR metaData

Usage

```
as.MetaData(x)
```

Arguments

x	the object to coerce

Value

a meta data class object

as.ObservationWindow *A coercion function to convert to a CAPR ObservationWindow*

Description

A coercion function to convert to a CAPR ObservationWindow

Usage

```
as.ObservationWindow(x)
```

Arguments

x	the object to coerce

Value

an observation window class object

as.Occurrence *A coercion function to convert to a CAPR Occurrence*

Description

A coercion function to convert to a CAPR Occurrence

Usage

```
as.Occurrence(x)
```

Arguments

x	the object to coerce

Value

a occurrence class object

as.QueryLoad *A coercion function to convert to a CAPR query*

Description

This function takes a saved CAPR query json and returns query CAPR R object

Usage

```
as.QueryLoad(x)
```

Arguments

x	the object to coerce

Value

a query class object

as.Timeline *A coercion function to convert to a CAPR timeline*

Description

A coercion function to convert to a CAPR timeline

Usage

```
as.Timeline(x)
```

Arguments

x	the object to coerce

Value

a timeline class object

as.Window *A coercion function to convert to a CAPR window*

Description

A coercion function to convert to a CAPR window

Usage

```
as.Window(x)
```

Arguments

x	the object to coerce

Value

a window class object

CensorWindow-class *An S4 class for CensorWindow*

Description

A class showing dates that indicate the range of entries the are captured in the cohort

Slots

StartDate	the left side of truncation for the study observation
EndDate	the right side of truncation for the study observation

checkConceptField *Function to get concept fields from concept set expression in object*

Description

Function to get concept fields from concept set expression in object

Usage

```
checkConceptField(x, field)
```

Arguments

x	the object to check
field	the concept field to check

Value

a list or vector of concept fields

checkConceptIds *Function to get concept ids from concept set expression in object*

Description

Function to get concept ids from concept set expression in object

Usage

```
checkConceptIds(x)
```

Arguments

x	the object to check

Value

a list or vector of concept id integers

CohortDefinition-class *An S4 class for a Circe Cohort Definition*

Description

A cohort definition contains information about how to quantify a clinical phenotype. The ultimate purpose of Capr is to allow the creation and manipulation of Circe cohort definitions in R making CohortDefinition its most important class.

Slots

`CohortDetails` a cohortDetails object providing meta information about the cohort

`PrimaryCriteria` a component class containing the primary criteria

`AdditionalCriteria` a component class containing the additional criteria

`InclusionRules` a component class containing the Inclusion Rules

`EndStrategy` a component class containing the End Strategy

`CensoringCriteria` a component class containing the censoring criteria

`CohortEra` a component class containing the cohort era

CohortDetails-class *Show Contents of a Component*

Description

This function prints the contents of a component. Note 1/27/21 attributes and some other s4 classes need to be implemented

Details

param showFullConceptSetExpressions T/F options to include full details of concept expressions An S4 class providing metadata for a CohortDefinition

The cohort details do not affect the cohort definition and are for improving human readability only.

Slots

`Name` a name for the cohort

`Description` a text field providing an information on the cohort and what it is intended

`Author` who created the cohort

`cdmVersionRange` the range of cdm versions

`CollapseSettings-class` *An S4 class for Collapse Settings*

Description

A class providing information that identifies the padding for cohort eras

Slots

`Type` boolean operator for the number of items in group to include. all, any, at most and at least

`Count` the number of criteria's needed for restriction. If Type is ALL or ANY this value is NA

`compileCohortDefinition` *Convert cohort definition object to CIRCE and run through circe compiler*

Description

This function converts a Cohort Definition class object to a CIRCE expression, creates the json and compiles the circe json to create ohdisql to run queries against a dbms containing OMOP cdm data

Usage

```
compileCohortDefinition(CohortDefinition, generateOptions = NULL)
```

Arguments

`CohortDefinition` input cohort Definition class object

`generateOptions` the options for building the ohdisql using CirceR::createGenerateOptions If generateOptions is left NULL, then this function will give a lite return of just the json to be activated. with circe R.

Value

If the generate options is supplied this function returns a three tiered list containing the the circe json, a text read and ohisql. If an error occurs the ohdisql slot will be NA and the user should review the circe cohort definition for potential errors. If the generateOptions is not supplied it will just return the json

Component-class *An S4 class for a cohort definition component*

Description

This class is an flexible container used to store the component parts of cohort definition allowing us to maintain information in smaller parts that remain relevant in isolation. The structure of a Circe cohort definition relies on a concept set table that stores information for queries. In each cohort component an internal reference id is used to maintain consistency between the expression of the cohort criteria and the actionable concepts. The component container bundles the concept set expression and the criteria expression into one object that is saveable and inheritable. Smaller classes are stored within the container and when they are converted into a superior class the component container is modified but the previous information is kept in tact. A component consists of 4 parts: MetaData stores the name, description and the ComponentType. The ComponentType identifies what kind of component one is using. Next the criteriaExpression stores any information about the deployment of the medical concept. This includes queries, counts, groups, attributes and other structures that detail the information of the specific component class. The limit is a slot that specifies the limit of entry for person events, e.g. the first event, all events, or last event for the criteriaExpression. Finally the ConceptSetExpression slot holds the concepts relevant to the criteria expression and their unique identifies. A Component object can be saved as a json file or loaded back into its s4 class. In some cases components can be nested inside other components TODO Explain the possible nesting structures that can exist. Question: why does metaData get its own class but other slots do not?

Slots

`MetaData` meta information about the object

`CriteriaExpression` a list of criteria that is in the object

`Limit` a list containing any limits

`ConceptSetExpression` a list containing any concept sets

componentType,Component-method *Function to find the Component Class*

Description

Function to find the Component Class

Usage

```
## S4 method for signature 'Component'
componentType(x)
```

Arguments

x	the component to check

Value

a character string with the component class

Concept-class *An S4 class for a ConceptSet*

Description

A concept class contains all the information about the concept from the OMOP voabulary

Slots

`CONCEPT_ID` the id of the concept
`CONCEPT_NAME` the name of the concept
`STANDARD_CONCEPT` whether the cncept is standard, single letter
`STANDARD_CONCEPT_CAPTION` whether the concept is standard full phrase
`INVALID_REASON` Whether the concept is invalid single letter
`INVALID_REASON_CAPTION` whether the concept is invalid standard phrase
`CONCEPT_CODE` the original code of the concept from its vocabulary
`DOMAIN_ID` the domain of the concept
`VOCABULARY_ID` the name of the vocabulary
`CONCEPT_CLASS_ID` type of concept class

ConceptAttribute-class *An S4 class for Concept Attribute*

Description

A concept attribute, using concepts to identify the attribute like a gender or race etc

Slots

`Name` the name of the attribute
`Concepts` a list containing the concepts used to identify the attribute

---

ConceptSetExpression-class *An S4 class for ConceptSetExpresion*

Description

A class for the concept set expressions bundles multiple concepts with mapping

Slots

`id` an id for the concept set expression to identify within a component
`Name` the name of the concept set expression
`Expression` a list containing expressions. expressions include multiple conceptSetItem objects

ConceptSetItem-class *An S4 class for ConceptSetItem*

Description

A class that provides information on the mapping of the concept

Slots

`Concept` a concept class object

`isExcluded` toggle if want to exclude the concept

`includeDescendants` toggle if want to include descendants

`includeMapped` toggle if want to include map

convertAdditionalCriteriaToCIRCE *Convert Additional Criteria Component to CIRCE*

Description

Convert Additional Criteria Component to CIRCE

Usage

```
convertAdditionalCriteriaToCIRCE(x)
```

Arguments

x	the component to convert

Value

a circe converted component

convertCensoringCriteriaToCIRCE *Convert Censoring Criteria Component to CIRCE*

Description

Convert Censoring Criteria Component to CIRCE

Usage

```
convertCensoringCriteriaToCIRCE(x)
```

Arguments

x	the component to convert

Value

a circe converted component

convertCohortDefinitionToCIRCE *Function to update cohort definition to CIRCE*

Description

Function to update cohort definition to CIRCE

Usage

```
convertCohortDefinitionToCIRCE(x)
```

Arguments

x	the cohort definition to convert to circe

Value

a circe object in R

convertCohortEraToCIRCE *Convert CohortEra Component to CIRCE*

Description

Convert CohortEra Component to CIRCE

Usage

```
convertCohortEraToCIRCE(x)
```

Arguments

x	the component to convert

Value

a circe converted component

convertEndStrategyToCIRCE *Convert End Strategy Component to CIRCE*

Description

Convert End Strategy Component to CIRCE

Usage

```
convertEndStrategyToCIRCE(x)
```

Arguments

x	the component to convert

Value

a circe converted component

convertInclusionRulesToCIRCE *Convert Inclusion Rules Component to CIRCE*

Description

Convert Inclusion Rules Component to CIRCE

Usage

```
convertInclusionRulesToCIRCE(x)
```

Arguments

x	the component to convert

Value

a circe converted component

convertPrimaryCriteriaToCIRCE *Convert Primary Criteria Component to CIRCE*

Description

Convert Primary Criteria Component to CIRCE

Usage

```
convertPrimaryCriteriaToCIRCE(x)
```

Arguments

x	the component to convert

Value

a circe converted component

convertRuleToCIRCE *Convert single rule (group) Component to CIRCE*

Description

Convert single rule (group) Component to CIRCE

Usage

```
convertRuleToCIRCE(x)
```

Arguments

x	the component to convert

Value

a circe converted component

CorrelatedCriteriaAttribute-class *An S4 class for CorrelatedCriteriaAttribute*

Description

A group attribute that is nested within a query.

Slots

`Name` name of the attribute

`Group` a group class object for the attribute

Count-class *An S4 class for a Count*

Description

A count class provides a number of occurrences of the query and the timeline that it happens

Slots

`Criteria` a query class object

`Timeline` a timeline class object

`Occurrence` an occurrence class object

createAdditionalCriteria

Function creates an Additional Criteria

Description

Function creates an Additional Criteria from a component class group

Usage

```
createAdditionalCriteria(Name, Contents = NULL, Limit, Description = NULL)
```

Arguments

Name	a character string naming the group object, this is required for the object. One should make the name descriptive of what the group is trying to identify.
Contents	a single component of group class that describes the additional criteria. If the Contents are empty then the additional criteria is only decribed by the qualified limit
Limit	how to limit initial events per person
Description	a character string describing the count object, this is optional so default is null

Value

new additional criteria component.

createAgeAtEndAttribute

create AgeAtEnd Attribute

Description

This function creates an Operator attribute for person AgeAtEnd. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createAgeAtEndAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	an integer for the AgeAtEnd
Extent	an integer for the AgeAtEnd only used if the op is bt or !bt

Value

a component of attribute class

createAgeAtStartAttribute *create AgeAtStart Attribute*

Description

This function creates an Operator attribute for person AgeAtStart. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createAgeAtStartAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	an integer for the AgeAtStart
Extent	an integer for the AgeAtStart only used if the op is bt or !bt

Value

a component of attribute class

createAgeAttribute *create Age Attribute*

Description

This function creates an Operator attribute for person age. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createAgeAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	an integer for the age
Extent	an integer for the age only used if the op is bt or !bt

Value

a component of attribute class

createAttributeCall *Get attributes from cohort expression and prepare R language*

Description

This function creates attributes within the queries and turns them into R language which will then create them as a CAPR object

Usage

```
createAttributeCall(x, objNm)
```

Arguments

x	the circe cohort definition
objNm	the naming convention to assign the object

Value

r language to generate the concept set expressions of the cohort

createCensoringCriteria *Function creates a Censoring Criteria*

Description

Function creates a Censoring Criteria from a list of queries

Usage

```
createCensoringCriteria(Name, ComponentList, Description = NULL)
```

Arguments

Name	a character string naming the inclusion rules, this is required for the object. One should make the name descriptive of what the group is trying to identify.
ComponentList	a list of component class queries to be inserted into the censoring criteria.
Description	a character string describing the count object, this is optional so default is null

Value

new censoring criteria component.

createCohortDefinition

Create Cohort Definition class object

Description

This function creates a Cohort Definition class object from multiple component parts. A cohort definition contains at a minimum a primary criteria class. The cohort definition can further contain a inclusion rules, additional criteria, censoring criteria and end strategy classes to provide more details on cohort restriction and cohort exit. Other components may also be manipulated but since they do not rely on a concept set expressions, they can be manipulated in separate methods. The cohort definition class differs from the circe expression in that it does not have a separate space for concept set expressions, which are bundled within the component.

Usage

```
createCohortDefinition(
  Name,
  Description = NA_character_,
  Author = NA_character_,
  cdmVersionRange = ">=5.0.0",
  PrimaryCriteria,
  AdditionalCriteria = NULL,
  InclusionRules = NULL,
  EndStrategy = NULL,
  CensoringCriteria = NULL,
  CohortEra = NULL
)
```

Arguments

Name	make a name for the cohort to add to the cohort details
Description	add a description detail to cohort details, optional
Author	add an author name to cohort details, optional
cdmVersionRange	add a cdm version range typically >= 5.0.0, please specify if not v5
PrimaryCriteria	add primary criteria object
AdditionalCriteria	add additional criteria object. if null then will create an additional criteria with qualified limit
InclusionRules	add inclusion rules object. if null will create empty inclusion rules with expression limit
EndStrategy	add end strategy object. if null will add end of continuous era strategy
CensoringCriteria	add censoring criteria object. if null will add empty censoring criteria
CohortEra	add cohort era object. if null will add collapse settings with 0 day pad and no censor window

Value

cohort definition class object with defined inputs. This can now be compiled into ohdisql and converted to json

---

`createCohortEra` *Create a Cohort Era class object*

Description

The Cohort Era depicts the time span of the cohort. The Censor Window includes the date window for which we register events. The Collapse Settings identify the era padding between events before exiting a cohort.

Usage

```
createCohortEra(EraPadDays = 0L, LeftCensorDate = NULL, RightCensorDate = NULL)
```

Arguments

`EraPadDays`	a numeric that specifies the number of days for the era padding
`LeftCensorDate`	a date string that specifies the starting date of registration
`RightCensorDate`	a date string that specifies the end date of registration

Value

a cohort era component

`createComponent` *createComponent*

Description

createComponent

Usage

```
createComponent(
  Name,
  Description = NULL,
  ComponentType = c("ConceptSetExpression", "Group", "Query", "Count", "Attribute",
    "PrimaryCriteria", "AdditionalCriteria", "InclusionRules", "EndStrategy",
    "CensoringCriteria", "CohortEra", "Empty"),
  CriteriaExpression = NULL,
  Limit = NULL,
  ConceptSetExpression = NULL
)
```

Arguments

Name	a name
Description	a description default null
ComponentType	match an arg from vector
CriteriaExpression	include anything for the criteria can be null
Limit	determine limit
ConceptSetExpression	add anny concept set expressions

---

createConceptAttribute *createConceptAttribue*

Description

createConceptAttribue

Usage

```
createConceptAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE,
  name
)
```

Arguments

conceptIds	the list of ids to lookup, need OMOP vocabulary connection
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.

mapToStandard	whether to map concept ids to standard or leave as is default is TRUE
name	name of the ttribute name

`createConceptMapping` *Function to help user develop the concept mapping*

Description

This function creates a concept mapping list that is used to establish the concept set item for each member of the concept set expression. This function is evolving.

Usage

```
createConceptMapping(
  n,
  includeDescendants = NULL,
  isExcluded = NULL,
  includeMapped = NULL
)
```

Arguments

n	the length of the concept set expression
includeDescendants	a logic vector of length n that contains the toggle for whether the concept should include descendants. If the parameter is left null then will return all FALSE
isExcluded	a logic vector of length n that contains the toggle for whether the concept should be excluded. If the parameter is left null then will return all FALSE
includeMapped	a logic vector of length n that contains the toggle for whether the concept should include mapped concepts. If the parameter is left null then will return all FALSE

Value

This function returns a list for concept mapping for the concept set expression

---

`createConceptSet` *Create Concept Set list*

Description

This function takes a data frame of OMOP concepts, establishes the mapping logic and bundles them together as a concept set item. With this function, toggling the mapping options sets the logic for all concepts in the concept set expression. If the user wants to set a custom mapping for each concept in the expression the user should use createConceptSetExpressionCustom. This is an evolving function.

Usage

```
createConceptSet(
  conceptSet,
  includeDescendants = TRUE,
  isExcluded = FALSE,
  includeMapped = FALSE
)
```

Arguments

conceptSet	a dataframe containing the concepts one would like to add to the concept set. The data frame of concepts can be queried using the lookup concept functions (requires a connection to an OMOP CDM).
includeDescendants	logic toggle where default true includes descendant concepts to the defined concept
isExcluded	logic toggle when true excludes the defined concept when attached to a concept set expression
includeMapped	logic toggle when true includes mapped concepts to the defined concept

Value

This function returns a concept set item object

---

createConceptSetExpression

Create Concept Set Expression

Description

This function takes a data frame of OMOP concepts, establishes the mapping logic and bundles them together as a concept set expression. A new concept expression created in R sets a guid for the concept id. This unique identifier is used to link the concept set expressions to its implementation within the cohort definition (typically as a query). With this function, toggling the mapping options sets the logic for all concepts in the concept set expression. If the user wants to set a custom mapping for each concept in the expression the user should use createConceptSetExpressionCustom. This is an evolving function.

Usage

```
createConceptSetExpression(
  conceptSet,
  Name,
  includeDescendants = TRUE,
  isExcluded = FALSE,
  includeMapped = FALSE
)
```

Arguments

conceptSet	a dataframe containing the concepts one would like to add to the concept set. The data frame of concepts can be queried using the lookup concept functions (requires a connection to an OMOP CDM).
Name	a name for the concept set expression.
includeDescendants	logic toggle where default true includes descendant concepts to the defined concept
isExcluded	logic toggle when true excludes the defined concept when attached to a concept set expression
includeMapped	logic toggle when true includes mapped concepts to the defined concept

Value

This function returns a component class object which contains the concept set expression

---

createConceptSetExpressionCustom

Create a Custom Concept Set Expression

Description

This function takes a data frame of OMOP concepts, establishes the mapping logic and bundles them together as a concept set expression. A new concept expression created in R sets a guid for the concept id. This unique identifier is used to link the concept set expressions to its implementation within the cohort definition (typically as a query). With this function, the user can pre-define a full list of mapping for each concept set item in the concept set expression. This is an evolving function

Usage

```
createConceptSetExpressionCustom(conceptSet, Name, conceptMapping = NULL)
```

Arguments

conceptSet	a dataframe containing the concepts one would like to add to the concept set. The data frame of concepts can be queried using the lookup concept functions (requires a connection to an OMOP CDM).
Name	a name for the concept set expression.
conceptMapping	a list of mapping for each concept set item. The list will contain whether the concept should includeDescendants, isExcluded or includeMapped. If the concept Mapping is left null then by default only the includeDescendants mapping will be true for all. others will remain false.

Value

This function returns a component class object which contains the concept set expression

createConditionEra *create ConditionEra for create Query*

Description

This function creates a query based on ConditionEra. Input pertinent conceptSetExpression and attirbuteList

Usage

```
createConditionEra(conceptSetExpression = NULL, attributeList = NULL)
```

Arguments

conceptSetExpression	place a component class concept set expression for domain. The concept set expressions must be adhere to the domain of the query
attributeList	a list of attributes to add to the query, if no attributes used then leave null

Value

a component of query class

createConditionOccurrence *create ConditionOccurrence for create Query*

Description

This function creates a query based on ConditionOccurrence. Input pertinent conceptSetExpression and attirbuteList

Usage

```
createConditionOccurrence(conceptSetExpression = NULL, attributeList = NULL)
```

Arguments

conceptSetExpression	place a component class concept set expression for domain. The concept set expressions must be adhere to the domain of the query
attributeList	a list of attributes to add to the query, if no attributes used then leave null

Value

a component of query class

createConditionSourceConceptAttribute
create condition source concept

Description

create condition source concept

Usage

```
createConditionSourceConceptAttribute(ConceptSetExpression)
```

Arguments

ConceptSetExpression	the concept set expression we wish to deploy as a source concept attribute This concept set expression should contain source codes, which may be non-standard.

Value

a source concept attribute component

createConditionTypeExcludeAttribute
create exclude attribute for condition type

Description

This function creates a attribute for exclusion

Usage

```
createConditionTypeExcludeAttribute(logic = FALSE)
```

Arguments

logic	toggle FALSE to not exclude

Value

a component of attribute class

createCorrelatedCriteriaAttribute *Function to create an attribute for a correlated criteria*

Description

Function to create an attribute for a correlated criteria

Usage

```
createCorrelatedCriteriaAttribute(Group)
```

Arguments

Group	a group object to add

Value

a correlated criteria attribute component

createCount *Function creates a count object*

Description

This function creates a count object of the cohort definition. The count object is used to express a query over a number of occurrences within a timeline relative to the initial event. A count comes from the number of times the applied query must be counted in the candidate patient timeline for them to be a suitable occurrence of a clinical construct.

Usage

```
createCount(
  Query,
  Logic = c("at_least", "at_most", "exactly"),
  Count,
  isDistinct = FALSE,
  Timeline,
  Name = NULL,
  Description = NULL
)
```

Arguments

Query	a component that is of query class
Logic	how to express the count i.e. exactly, at_least, at_most
Count	how many times the query occurs to be eligible
isDistinct	a logic toggle where if TRUE only counts distinct occurrences

Timeline	a timeline class object orienting the time points of recording in reference to the initial event
Name	a character string naming the count object, this is optional so default is null
Description	a character string describing the count object, this is optional so default is null

Value

This function returns a component class object which contains the count object and attached concept set expressions

---

`createCountCall` *Get counts from cohort expression and prepare R language*

Description

Get counts from cohort expression and prepare R language

Usage

```
createCountCall(x, nm)
```

Arguments

x	the circe cohort definition
nm	the naming convention to assign the object

Value

r language to generate the counts of the cohort

`createCustomEraEndStrategy`
Function creates an end strategy from a custom era

Description

This function creates a custom era end strategy. From the ATLAS page: Specify a concept set that contains one or more drugs. A drug era will be derived from all drug exposure events for any of the drugs within the concept set, using the specified persistence window as a maximum allowable gap in days between successive exposure events and adding a specified surveillance window to the final exposure event. If no exposure event end date is provided, then an exposure event end date is inferred to be event start date + days supply in cases when days supply is available or event start date + 1 day otherwise. This event persistence assures that the cohort end date will be no greater than the drug era end date.

Usage

```
createCustomEraEndStrategy(ConceptSetExpression, gapDays, offset)
```

Arguments

ConceptSetExpression	a component of concept set expression class that contains information on the drug concets to use to define the end strategy
gapDays	the maximum allowable days between successive exposures.
offset	an integer value specifying padding to the cohort exit.

Value

This function returns a component class object which contains the end strategy object

---

createDatabaseConnectionLang

Create details for database connection

Description

This function will create the database connection in the script

Usage

```
createDatabaseConnectionLang(
  connectionDetails = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL
)
```

Arguments

connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL which will create dummy credentials
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.

Value

r language to generate the connection to dbms. Be cautious to not expose credentials

createDateOffsetEndStrategy

Function creates a date offset end strategy

Description

This function creates a date offset end strategy. From the ATLAS page: the event end date is derived from adding a number of days to the event's start or end date. If an offset is added to the event's start date, all cohort episodes will have the same fixed duration (subject to further censoring). If an offset is added to the event's end date, persons in the cohort may have varying cohort duration times due to the varying event durations (such as eras of persistent drug exposure or visit length of stay). This event persistence assures that the cohort end date will be no greater than the selected index event date, plus the days offset.

Usage

```
createDateOffsetEndStrategy(
  offset,
  eventDateOffset = c("StartDate", "EndDate")
)
```

Arguments

offset	an integer value specifying padding to the cohort exit.
eventDateOffset	an input only for DateOffset specifying whether to add an offset to the start or end of an event (i.e. StartDate, EndDate)

Value

This function returns a component class object which contains the end strategy object

createDaysSupplyAttribute

create DaysSupply Attribute

Description

This function creates an Operator attribute for person DaysSupply. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createDaysSupplyAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	an integer for the DaysSupply
Extent	an integer for the DaysSupply only used if the op is bt or !bt

Value

a component of attribute class

---

createDeath *create Death for create Query*

Description

This function creates a query based on Death. Input pertinent conceptSetExpression and attributeList

Usage

```
createDeath(conceptSetExpression = NULL, attributeList = NULL)
```

Arguments

conceptSetExpression	place a component class concept set expression for domain. The concept set expressions must be adhere to the domain of the query
attributeList	a list of attributes to add to the query, if no attributes used then leave null

Value

a component of query class

---

createDeathSourceConceptAttribute *create Death source concept*

Description

create Death source concept

Usage

```
createDeathSourceConceptAttribute(ConceptSetExpression)
```

Arguments

ConceptSetExpression	the concept set expression we wish to deploy as a source concept attribute This concept set expression should contain source codes, which may be non-standard.

Value

a source concept attribute component

`createDeathTypeAttribute` *create DeathType as a concept Attribute*

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createDeathTypeAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

`conceptIds`	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
`connectionDetails`	An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.
`connection`	An object of type `connection` as created using the `connect` function in the DatabaseConnector package. Can be left NULL if `connectionDetails` is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
`vocabularyDatabaseSchema`	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
`tempEmulationSchema`	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
`mapToStandard`	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

createDeathTypeExcludeAttribute *create exclude attribute for death type*

Description

This function creates a attribute for exclusion

Usage

```
createDeathTypeExcludeAttribute(logic = FALSE)
```

Arguments

logic	toggle FALSE to not exclude

Value

a component of attribute class

createDeviceExposure *create DeviceExposure for create Query*

Description

This function creates a query based on DeviceExposure. Input pertinent conceptSetExpression and attirbuteList

Usage

```
createDeviceExposure(conceptSetExpression = NULL, attributeList = NULL)
```

Arguments

conceptSetExpression	place a component class concept set expression for domain. The concept set expressions must be adhere to the domain of the query
attributeList	a list of attributes to add to the query, if no attributes used then leave null

Value

a component of query class

createDeviceSourceConceptAttribute *create Device source concept*

Description

create Device source concept

Usage

```
createDeviceSourceConceptAttribute(ConceptSetExpression)
```

Arguments

ConceptSetExpression	the concept set expression we wish to deploy as a source concept attribute This concept set expression should contain source codes, which may be non-standard.

Value

a source concept attribute component

createDeviceTypeAttribute *create DeviceType as a concept Attribute*

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createDeviceTypeAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

conceptIds	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
mapToStandard	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

---

createDoseEra *create DoseEra for create Query*

Description

This function creates a query based on DoseEra. Input pertinent conceptSetExpression and attributeList

Usage

```
createDoseEra(conceptSetExpression = NULL, attributeList = NULL)
```

Arguments

conceptSetExpression	place a component class concept set expression for domain. The concept set expressions must be adhere to the domain of the query
attributeList	a list of attributes to add to the query, if no attributes used then leave null

Value

a component of query class

createDoseUnitAttribute *create DoseUnit as a concept Attribute*

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createDoseUnitAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

conceptIds	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
mapToStandard	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

createDrugEra *create DrugEra for create Query*

Description

This function creates a query based on DrugEra. Input pertinent conceptSetExpression and attirbuteList

Usage

```
createDrugEra(conceptSetExpression = NULL, attributeList = NULL)
```

Arguments

conceptSetExpression	place a component class concept set expression for domain. The concept set expressions must be adhere to the domain of the query
attributeList	a list of attributes to add to the query, if no attributes used then leave null

Value

a component of query class

createDrugExposure *create DrugExposure for create Query*

Description

This function creates a query based on DrugExposure. Input pertinent conceptSetExpression and attirbuteList

Usage

```
createDrugExposure(conceptSetExpression = NULL, attributeList = NULL)
```

Arguments

conceptSetExpression	place a component class concept set expression for domain. The concept set expressions must be adhere to the domain of the query
attributeList	a list of attributes to add to the query, if no attributes used then leave null

Value

a component of query class

`createDrugSourceConceptAttribute` *create Drug source concept*

Description

create Drug source concept

Usage

```
createDrugSourceConceptAttribute(ConceptSetExpression)
```

Arguments

`ConceptSetExpression`	the concept set expression we wish to deploy as a source concept attribute This concept set expression should contain source codes, which may be non-standard.

Value

a source concept attribute component

`createDrugTypeAttribute` *create DrugType as a concept Attribute*

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createDrugTypeAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

conceptIds	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
mapToStandard	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

---

createDrugTypeExcludeAttribute

create exclude attribute for drug type

Description

This function creates a attribute for exclusion

Usage

```
createDrugTypeExcludeAttribute(logic = FALSE)
```

Arguments

logic	toggle FALSE to not exclude

Value

a component of attribute class

createEffectiveDrugDoseAttribute *create EffectiveDrugDose Attribute*

Description

This function creates an Operator attribute for person EffectiveDrugDose. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createEffectiveDrugDoseAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	an integer for the EffectiveDrugDose
Extent	an integer for the EffectiveDrugDose only used if the op is bt or !bt

Value

a component of attribute class

createEmptyComponent *Create an Empty Component*

Description

Create an Empty Component

Usage

```
createEmptyComponent()
```

Value

an empty component

createEraEndDateAttribute

create era End Date Attribute

Description

This function creates an Operator attribute for the era end date. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createEraEndDateAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	a character string of the date
Extent	a character string of the extent only used if the op is bt or !bt

Value

a componet of attribute class

createEraLengthAttribute

create EraLength Attribute

Description

This function creates an Operator attribute for person EraLength. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createEraLengthAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	an integer for the EraLength
Extent	an integer for the EraLength only used if the op is bt or !bt

Value

a component of attribute class

createEraStartDateAttribute *create Era start Date Attribute*

Description

This function creates an Operator attribute for the era start date. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createEraStartDateAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	a character string of the date
Extent	a character string of the extent only used if the op is bt or !bt

Value

a componet of attribute class

createFirstAttribute *create First Occurrence Attribute*

Description

This function creates a attribute for first occurrence

Usage

```
createFirstAttribute(logic = TRUE)
```

Arguments

logic	toggle TRUE for first occurence

Value

a component of attribute class

createGapDaysAttribute *create GapDays Attribute*

Description

This function creates an Operator attribute for person GapDays. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createGapDaysAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	an integer for the GapDays
Extent	an integer for the GapDays only used if the op is bt or !bt

Value

a component of attribute class

createGenderAttribute *create gender as a concept Attribute*

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createGenderAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

conceptIds	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
mapToStandard	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

---

createGroup *Function creates a group object*

Description

This function creates a group object of the cohort definition. The group object binds multiple queries, counts, attributes and other groups to create one component. For entry into the cohort the patient must have a valid instance of all aspects of the group. Groups are used in additional criteria, inclusion rules and correlated criteria. One can attach a list of counts as a criteria list, a list of demographic criteria (select attributes) or a list of sub groups.

Usage

```
createGroup(
  Name,
  type = c("ALL", "ANY", "AT_LEAST", "AT_MOST"),
  count = NULL,
  criteriaList = NULL,
  demographicCriteriaList = NULL,
  Groups = NULL,
  Description = NULL
)
```

Arguments

Name	a character string naming the group object, this is required for the object. One should make the name descriptive of what the group is trying to identify.
type	a character string expressing the combination of qualifiying criterias for restriction. Valid options are ALL meaning all aspects of the group must be true to enter cohort, ANY meaning at least 1 aspect of the group must be true, AT_LEAST meaning at least a certain count of the group must be true of AT_MOST meaning at most a certain count must be true of the group. The type entry must be in all capital letters
count	the count of criterias needed for restriction. The count only applies if the type if AT_LEAST or AT_MOST. Otherwise this parameter remains NULL
criteriaList	a list of component class count objects to be added. May be left empty, but at least one of criteriaList, demographicCriteriaList and Groups must be filled. The input must be a list of components
demographicCriteriaList	a list of select component class attributes to be added. May be left empty, but at least one of criteriaList, demographicCriteriaList and Groups must be filled. The input must be a list of components
Groups	a list of component class groups to be added. May be left empty, but at least one of criteriaList, demographicCriteriaList and Groups must be filled. The input must be a list of components
Description	a character string describing the count object, this is optional so default is null

Value

This function returns a component class object which contains the group object and attached concept set expressions

---

createGroupCall *Get groups from cohort expression and prepare R language*

Description

This function creates groups from cohort and turns them into R language which will then create them as a CAPR objects

Usage

```
createGroupCall(x, nm, assignName = NULL)
```

Arguments

x	the circe cohort definition
nm	the naming convention for sub-objects
assignName	the naming convention to assign the object

Value

r language to generate the groups of the cohort

createInclusionRules *Function creates an Inclusion Rule*

Description

Function creates a Inclusion Rule from a list of groups, each specifying a unique rule

Usage

```
createInclusionRules(Name, Contents, Limit, Description = NULL)
```

Arguments

Name	a character string naming the inclusion rules, this is required for the object. One should make the name descriptive of what the group is trying to identify.
Contents	a list of component class groups to be inserted into the inclusion rules. Each group in the list is a separate rule.
Limit	how to limit initial events per person
Description	a character string describing the count object, this is optional so default is null

Value

new inclusion rules component.

createLogicalAttribute *createLogicalAttribue*

Description

createLogicalAttribue

Usage

```
createLogicalAttribute(name, logic = TRUE)
```

Arguments

name	is the name of the attribute
logic	whether the logic is true or false, default is true

createMeasurement *create Measurement for create Query*

Description

This function creates a query based on Measurement. Input pertinent conceptSetExpression and attirbuteList

Usage

```
createMeasurement(conceptSetExpression = NULL, attributeList = NULL)
```

Arguments

conceptSetExpression	place a component class concept set expression for domain. The concept set expressions must be adhere to the domain of the query
attributeList	a list of attributes to add to the query, if no attributes used then leave null

Value

a component of query class

createMeasurementSourceConceptAttribute *create measurement source concept*

Description

create measurement source concept

Usage

```
createMeasurementSourceConceptAttribute(ConceptSetExpression)
```

Arguments

ConceptSetExpression	the concept set expression we wish to deploy as a source concept attribute This concept set expression should contain source codes, which may be non-standard.

Value

a source concept attribute component

createMeasurementTypeAttribute
create MeasurementType as a concept Attribute

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createMeasurementTypeAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

conceptIds	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
mapToStandard	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

createMeasurementTypeExcludeAttribute

create exclude attribute for measurement type

Description

This function creates a attribute for exclusion

Usage

```
createMeasurementTypeExcludeAttribute(logic = FALSE)
```

Arguments

logic	toggle FALSE to not exclude

Value

a component of attribute class

createModifierAttribute

create Modifier as a concept Attribute

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createModifierAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

conceptIds	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.

connection An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.

vocabularyDatabaseSchema
Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.

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

mapToStandard a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

---

createObservation *create Observation for create Query*

Description

This function creates a query based on Observation. Input pertinent conceptSetExpression and attirbuteList

Usage

```
createObservation(conceptSetExpression = NULL, attributeList = NULL)
```

Arguments

conceptSetExpression
place a component class concept set expression for domain. The concept set expressions must be adhere to the domain of the query

attributeList a list of attributes to add to the query, if no attributes used then leave null

Value

a component of query class

createObservationPeriod

create ObservationPeriod for create Query

Description

This function creates a query based on ObservationPeriod. Input pertinent conceptSetExpression and attirbuteList

Usage

```
createObservationPeriod(conceptSetExpression = NULL, attributeList = NULL)
```

Arguments

conceptSetExpression	place a component class concept set expression for domain. The concept set expressions must be adhere to the domain of the query
attributeList	a list of attributes to add to the query, if no attributes used then leave null

Value

a component of query class

createObservationSourceConceptAttribute

create observation source concept

Description

create observation source concept

Usage

```
createObservationSourceConceptAttribute(ConceptSetExpression)
```

Arguments

ConceptSetExpression	the concept set expression we wish to deploy as a source concept attribute This concept set expression should contain source codes, which may be non-standard.

Value

a source concept attribute component

`createObservationTypeAttribute` *create ObservationType as a concept Attribute*

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createObservationTypeAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

`conceptIds`	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
`connectionDetails`	An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.
`connection`	An object of type `connection` as created using the `connect` function in the DatabaseConnector package. Can be left NULL if `connectionDetails` is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
`vocabularyDatabaseSchema`	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
`tempEmulationSchema`	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
`mapToStandard`	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

createObservationTypeExcludeAttribute

create exclude attribute for observation type

Description

This function creates a attribute for exclusion

Usage

```
createObservationTypeExcludeAttribute(logic = FALSE)
```

Arguments

logic	toggle FALSE to not exclude

Value

a component of attribute class

createObservationWindow

Function creates an Observation Window

Description

This function creates an observation window used in a primary criteria. The observation window provides the amount of time before and after the initial event of continuous observation necessary for a person to be eligible to enter the cohort. The minimal observation days would be 0 days of prior observation and 0 days of post observations. This is the default for this function.

Usage

```
createObservationWindow(PriorDays = 0L, PostDays = 0L)
```

Arguments

PriorDays	number of days prior to the initial event of continuous observation
PostDays	number of days of continous observation after index date

Value

This function returns a observation window class object providing prior and post days of observation

createOccurrenceEndDateAttribute

create occurrence End Date Attribute

Description

This function creates an Operator attribute for the occurrence end date. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createOccurrenceEndDateAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	a character string of the date
Extent	a character string of the extent only used if the op is bt or !bt

Value

a componet of attribute class

createOccurrenceStartDateAttribute

create occurrence Start Date Attribute

Description

This function creates an Operator attribute for the occurrence start date. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createOccurrenceStartDateAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	a character string of the date
Extent	a character string of the extent only used if the op is bt or !bt

Value

a component of attribute class

`createOpAttribute` *createOpAttribute*

Description

createOpAttribute

Usage

```
createOpAttribute(Name, Op, Value, Extent = NULL)
```

Arguments

`Name`	a name
`Op`	a type of operator
`Value`	a value either integer or character for dates
`Extent`	only if Op is bt or !bt, otherwise NULL. Value is either integer or character for dates

`createOperatorAttribute`
create Operator as a concept Attribute

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createOperatorAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

`conceptIds`	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
`connectionDetails`	An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.

connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
mapToStandard	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

`createPeriodEndDateAttribute` *create period End Date Attribute*

Description

This function creates an Operator attribute for the period end date. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createPeriodEndDateAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	a character string of the date
Extent	a character string of the extent only used if the op is bt or !bt

Value

a componet of attribute class

createPeriodStartDateAttribute

create period Start Date Attribute

Description

This function creates an Operator attribute for the period start date. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createPeriodStartDateAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	a character string of the date
Extent	a character string of the extent only used if the op is bt or !bt

Value

a componet of attribute class

createPlaceOfServiceAttribute

create PlaceOfService as a concept Attribute

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createPlaceOfServiceAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

conceptIds	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
mapToStandard	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

---

createPrimaryCriteria *Function creates a Primary Criteria*

Description

Function creates a primary criteria from multiple queries. User adds a list of component class queries, identifies the observation window and the criteria limit.

Usage

```
createPrimaryCriteria(
  Name,
  ComponentList,
  ObservationWindow = NULL,
  Limit,
  Description = NULL
)
```

Arguments

Name	a character string naming the group object, this is required for the object. One should make the name descriptive of what the group is trying to identify.
ComponentList	a list of query components to add to the primary criteria. These components include the queries and concept set expression used in the cohort.

`ObservationWindow`
: an observationWindow class object that set the prior and post days of continuous observation for the initial event

`Limit`
: how to limit initial events per person

`Description`
: a character string describing the count object, this is optional so default is null

Value

new primary criteria component.

`createProcedureOccurrence`
create ProcedureOccurrence for create Query

Description

This function creates a query based on ProcedureOccurrence. Input pertinent conceptSetExpression and attirbuteList

Usage

```
createProcedureOccurrence(conceptSetExpression = NULL, attributeList = NULL)
```

Arguments

`conceptSetExpression`
: place a component class concept set expression for domain. The concept set expressions must be adhere to the domain of the query

`attributeList`
: a list of attributes to add to the query, if no attributes used then leave null

Value

a component of query class

`createProcedureSourceConceptAttribute`
create procedure source concept

Description

create procedure source concept

Usage

```
createProcedureSourceConceptAttribute(ConceptSetExpression)
```

Arguments

`ConceptSetExpression`
: the concept set expression we wish to deploy as a source concept attribute This concept set expression should contain source codes, which may be non-standard.

Value

a source concept attribute component

`createProcedureTypeAttribute` *create ProcedureType as a concept Attribute*

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createProcedureTypeAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

`conceptIds`	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
`connectionDetails`	An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.
`connection`	An object of type `connection` as created using the `connect` function in the DatabaseConnector package. Can be left NULL if `connectionDetails` is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
`vocabularyDatabaseSchema`	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
`tempEmulationSchema`	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
`mapToStandard`	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

createProcedureTypeExcludeAttribute

create exclude attribute for procedure type

Description

This function creates a attribute for exclusion

Usage

```
createProcedureTypeExcludeAttribute(logic = FALSE)
```

Arguments

logic	toggle FALSE to not exclude

Value

a component of attribute class

createProviderSpecialtyAttribute

create ProviderSpecialty as a concept Attribute

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createProviderSpecialtyAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

conceptIds	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.

connection An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.

vocabularyDatabaseSchema
Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.

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

mapToStandard a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

---

createQualifierAttribute

create Qualifier as a concept Attribute

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createQualifierAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

conceptIds a vector of concept ids. Must be connected to an OMOP vocabulary to use function

connectionDetails
An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.

connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
mapToStandard	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

`createQuantityAttribute` *create Quantity Attribute*

Description

This function creates an Operator attribute for person Quantity. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createQuantityAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	an integer for the Quantity
Extent	an integer for the Quantity only used if the op is bt or !bt

Value

a component of attribute class

createQuery *createQuery*

Description

createQuery

Usage

```
createQuery(
  Domain,
  Component = NULL,
  attributeList = NULL,
  Name = NULL,
  Description = NULL
)
```

Arguments

Domain	list the domain from the table we are searching in the query
Component	add the concept set expression we want to query
attributeList	a list of attribute class components to add, if not attributes keep null
Name	is the name of query, optional
Description	an optional description of the query

createQueryCall *Get queries from cohort expression and prepare R language*

Description

This function creates queries and turns them into R language which will then create them as a CAPR object

Usage

```
createQueryCall(x, nm)
```

Arguments

x	the circe cohort definition
nm	the naming convention to assign the object

Value

r language to generate the concept set expressions of the cohort

createRangeHighAttribute

create RangeHigh Attribute

Description

This function creates an Operator attribute for person RangeHigh. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createRangeHighAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	an integer for the RangeHigh
Extent	an integer for the RangeHigh only used if the op is bt or !bt

Value

a component of attribute class

createRangeHighRatioAttribute

create RangeHighRatio Attribute

Description

This function creates an Operator attribute for person RangeHighRatio. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createRangeHighRatioAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	an integer for the RangeHighRatio
Extent	an integer for the RangeHighRatio only used if the op is bt or !bt

Value

a component of attribute class

createRangeLowAttribute

create RangeLow Attribute

Description

This function creates an Operator attribute for person RangeLow. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createRangeLowAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	an integer for the RangeLow
Extent	an integer for the RangeLow only used if the op is bt or !bt

Value

a component of attribute class

createRangeLowRatioAttribute

create RangeLowRatio Attribute

Description

This function creates an Operator attribute for person RangeLowRatio. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createRangeLowRatioAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	an integer for the RangeLowRatio
Extent	an integer for the RangeLowRatio only used if the op is bt or !bt

Value

a component of attribute class

createRefillsAttribute

create Refills Attribute

Description

This function creates an Operator attribute for person Refills. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createRefillsAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	an integer for the Refills
Extent	an integer for the Refills only used if the op is bt or !bt

Value

a component of attribute class

createRouteConceptsAttribute

create RouteConcepts as a concept Attribute

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createRouteConceptsAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

conceptIds	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
mapToStandard	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

---

createSourceConceptAttribute

createSourceConceptAttribute

Description

createSourceConceptAttribute

Usage

```
createSourceConceptAttribute(Domain, ConceptSetExpression)
```

Arguments

Domain	the type of domain for the source concept
ConceptSetExpression	the concept set expression component to add

createTimeline *Set the Timeline in the criteria*

Description

When a criteria object is initialized a default timeline object is also initialized. To change the timeline object we set it to a new information. Inputs include StartWindow, EndWindow, RestrictVisit, and IgnoreObservationPeriod. The StartWindow and EndWindow inputs require a window class object. A new window can be initialized using the createWindow function.

Usage

```
createTimeline(
  StartWindow,
  EndWindow = NULL,
  RestrictVisit = FALSE,
  IgnoreObservationPeriod = FALSE
)
```

Arguments

StartWindow	a window class object that modifies when to begin monitoring for an observation
EndWindow	a window class object that ends the time observing events. This window is not always created so the default is NULL, initializing an empty window
RestrictVisit	a logic toggle where TRUE restricts to the same visit
IgnoreObservationPeriod	a logic toggle where TRUE allows events outside the observation period

Value

a new Timeline class object

createTimelineCall *Function to create a timeline call*

Description

Function to create a timeline call

Usage

```
createTimelineCall(x, objectName)
```

Arguments

x	the circe cohort definition
objectName	the naming convention to assign the object

Value

r language to generate the timelines of the cohort

createUnitAttribute *create Unit as a concept Attribute*

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createUnitAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

conceptIds	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
mapToStandard	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

createValueAsConceptAttribute *create value as a concept Attribute*

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createValueAsConceptAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

conceptIds	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
mapToStandard	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

createValueAsNumberAttribute *create ValueAsNumber Attribute*

Description

This function creates an Operator attribute for person ValueAsNumber. The user selects the type of operator, value which is the minimal bound and extent which is the end point of a between bound. Extent is only used if the op is bt or !bt.

Usage

```
createValueAsNumberAttribute(Op, Value, Extent = NULL)
```

Arguments

Op	defines logic for interpreting the numeric or date value.
Value	an integer for the ValueAsNumber
Extent	an integer for the ValueAsNumber only used if the op is bt or !bt

Value

a component of attribute class

createVisitOccurrence *create VisitOccurrence for create Query*

Description

This function creates a query based on visitOccurrence. Input pertinent conceptSetExpression and attirbuteList

Usage

```
createVisitOccurrence(conceptSetExpression = NULL, attributeList = NULL)
```

Arguments

conceptSetExpression	place a component class concept set expression for domain. The concept set expressions must be adhere to the domain of the query
attributeList	a list of attributes to add to the query, if no attributes used then leave null

Value

a component of query class

`createVisitSourceConceptAttribute` *create Visit source concept*

Description

create Visit source concept

Usage

```
createVisitSourceConceptAttribute(ConceptSetExpression)
```

Arguments

`ConceptSetExpression`	the concept set expression we wish to deploy as a source concept attribute This concept set expression should contain source codes, which may be non-standard.

Value

a source concept attribute component

`createVisitTypeAttribute` *create VisitType as a concept Attribute*

Description

This function creates an attribute out of concept values. input concept ids to actionize them within the attribute. One must clarify if the concept ids should be mapped to standard (default is TRUE) or leave them as is. User needs to be connected to an OMOP vocabulary to use the lookup functions.

Usage

```
createVisitTypeAttribute(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

conceptIds	a vector of concept ids. Must be connected to an OMOP vocabulary to use function
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
mapToStandard	a logical that indicates whether the concept Ids should be mapped to standard concepts

Value

a componet of attribute class

---

createVisitTypeExcludeAttribute

create exclude attribute for visit type

Description

This function creates a attribute for exclusion

Usage

```
createVisitTypeExcludeAttribute(logic = FALSE)
```

Arguments

logic	toggle FALSE to not exclude

Value

a component of attribute class

createWindow *Function to initialize a new window object*

Description

A window depicts the timeline from which events are counted. The window has four components: Start, End, EventStart, and Index Start. First, we determine whether observations are viewed from the start of the event or at the end. By default EventStart is TRUE. Next the start of recording is identied using days and coefficient. The coefficient distinguishes how the days are counted relative to the index date. The end recording is the same as the start, now identifying the end of observation. Finally it is identified whether the index date is relative the start or end of occurrence. A timeline has a start and end window. Usually the end window is not defined. An End Window adds a constraint to the Start Window of a timeline

Usage

```
createWindow(
  StartDays,
  StartCoeff = c("Before", "After"),
  EndDays,
  EndCoeff = c("Before", "After"),
  EventStarts = TRUE,
  IndexStart = TRUE
)
```

Arguments

StartDays	number of days at start of window
StartCoeff	where to begin counting relative to index date: before or after
EndDays	number of days to end window
EndCoeff	where to end counting relative to index date: before or after
EventStarts	if TRUE then this counts from the start of an event otherwise from the end of an event
IndexStart	if TRUE then the index date is the start of event otherwise the end of an event

Value

a new window class object

createWindowCall *Function to create a window object call*

Description

Function to create a window object call

Usage

```
createWindowCall(x)
```

Arguments

x	the circe cohort definition

Value

r language to generate the windows of the cohort

CustomEraEndStrategy-class

An S4 class for CustomEraEndStrategy

Description

An end strategy class specifying the time until the end of drug use for cohort exit

Slots

`DrugCodesetId` the guid of the drug concept set expression to activate in the end strategy

`GapDays` an integer showing the maximum allowable days between successive exposures.

`Offset` an integer value specifying padding to the cohort exit.

DateOffsetEndStrategy-class

An S4 class for DateOffsetEndStrategy

Description

An end strategy class specifying a number of days from the start or end of the initial event until cohort exit

Slots

`DateField` a character string specifying either the StartDate or EndDate of the initial event to begin counting days until cohort exit

`Offset` an integer value specifying padding to the cohort exit.

editConceptSetItem *Function to edit a concept set item*

Description

This function edits a concept set item class

Usage

```
editConceptSetItem(obj, edit, index = NULL, mapping = NULL, newName = NULL)
```

Arguments

obj	the component you wish to edit
edit	the edit to make
index	an index to specify a postion in a list
mapping	a character of includeDescendants, isExcluded or includeMapped to toggle logic
newName	a character string updating the name of the concept set expression

Value

the edit s4 class object

editCount *Function to edit Meta data*

Description

This function edits a meta data class

Usage

```
editCount(obj, edit, slotNms)
```

Arguments

obj	the component you wish to edit
edit	the edit to make
slotNms	a list object where each entry is a slot across multiple objects. The list must be constructed in order and can be done by following the object structure. For example to edit a window in the timeline one must construct a list('Timeline', 'StartWindow','Start'). If one wants to edit the count in an occurrence the list is: list('Occurrence','Count').

Value

the edit s4 class object

editExpressionType *Function to edit Expression type*

Description

This function edits a expression type class

Usage

```
editExpressionType(obj, edit, slotNm)
```

Arguments

obj	the component you wish to edit
edit	the edit to make
slotNm	the slot to edit

Value

the edit s4 class object

editInclusionRules *Function to edit Inclusion Rules*

Description

This function edits a meta data class

Usage

```
editInclusionRules(
  inclusionRules,
  edit,
  detail = c("Name", "Description", "Rule", "Limit"),
  add = FALSE,
  index = NULL
)
```

Arguments

inclusionRules	the inclusion rules component you wish to edit
edit	the edit to make. The edit must conform to the structure of the location detail where the edit is made. See detail for more information
detail	the slot to edit in the inclusion rules. Options are: Name, Description, Rule and Limit. If editing the name or description the edit must be a character string. If editing the limit the edit must be a character string of either All, First or Last. If the detail is a rule, the edit must be a Group type component class. One can use the function componentType to check the type for a component class object.
add	a loggic toggle to say if you are adding a piece to the pc component
index	an index to specify the position in a list that is to be modified. If null defaults to 1

Value

the edit s4 class object

`editLimit` *Function to edit Limit*

Description

This function edits a limit class

Usage

```
editLimit(obj, edit = c("All", "First", "Last"))
```

Arguments

`obj`	the component you wish to edit
`edit`	the edit to make either all first or last

Value

the edited s4 class object

`editMetaData` *Function to edit Meta data*

Description

This function edits a meta data class

Usage

```
editMetaData(obj, slotNm, edit)
```

Arguments

`obj`	the component you wish to edit
`slotNm`	the slot to edit
`edit`	the edit to make

Value

the edit s4 class object

editObservationWindow *Function to edit Observation Window*

Description

This function edits a observation window class

Usage

```
editObservationWindow(obj, slotNm, edit)
```

Arguments

obj	the component you wish to edit
slotNm	the slot to edit
edit	the edit to make

Value

the edit s4 class object

editOccurrence *Function to edit an Occurrence*

Description

This function edits an occurrence class

Usage

```
editOccurrence(obj, slotNm, edit)
```

Arguments

obj	the component you wish to edit
slotNm	the slot to edit
edit	the edit to make

Value

the edit s4 class object

`editPrimaryCriteria` *Function to edit Primary Criteria*

Description

This function edits a meta data class

Usage

```
editPrimaryCriteria(
  primaryCriteria,
  detail = c("Name", "Description", "CriteriaList", "Attribute", "PriorDays",
    "PostDays", "ObservationWindow", "PrimaryCriteriaLimit", "ConceptSetItem",
    "ConceptMapping"),
  edit,
  add = FALSE,
  index = NULL,
  mapping = NULL
)
```

Arguments

`primaryCriteria`	the primary criteria component you wish to edit
`detail`	the slot to edit. The options include: Name, Description, CriteriaList, Attribute, PriorDays, PostDays, ObservationWindow, PrimaryCriteriaLimit, ConceptSetItem, ConceptSetMapping. Each slot has a particular edit type.
`edit`	the edit to make. If the detail is Name or Description the edit must be a character string. If the edit is PriorDays, PostDays or ObservationWindow the edit must be an integer, where the ObservationWindow is an edit of two integers to modify both the prior and post days. If the edit is to the PrimaryCriteriaLimit the edit must be a character string of All, First or Last. If the edit is to the conceptSetItem the edit must be a ConceptSetItem class. And if the edit is to the concept set mapping the edit must be a logical (T/F). If the edit is to the CriteriaList it must be a query type component and if it is to the attribute it must be an attribute type component
`add`	a loggic toggle to say if you are adding a piece to the pc component
`index`	an index to specify the position in a vector. This is needed for CriteriaList, Attribute, and edits to the concept sets. The CriteriaList only needs a single index. The others need one index for the position in the list and a second for the position inside the substructure.
`mapping`	an character string specifying the mapping to change. Options are includeDescendants, isExcluded, and includeMapped. This is only required if the detail is ConceptSetMapping

Value

the edit s4 class object

editQuery *Function to edit Query*

Description

This function edits a query class

Usage

```
editQuery(obj, edit, slotNm, index = NULL)
```

Arguments

obj	the component you wish to edit
edit	the edit to make
slotNm	the slot to edit
index	an integer index specifying the location within a list, if not needed leave null

Value

the edit s4 class object

editTimeline *Function to edit Timeline*

Description

This function edits a timeline class

Usage

```
editTimeline(obj, slotNm, edit)
```

Arguments

obj	the component you wish to edit
slotNm	the slot to edit
edit	the edit to make

Value

the edit s4 class object

editWindow *Function to edit a window*

Description

This function edits a window class

Usage

```
editWindow(obj, slotNm, edit)
```

Arguments

obj	the component you wish to edit
slotNm	the slot to edit
edit	the edit to make

Value

the edit s4 class object

EndOfCtsObsEndStrategy-class *An S4 class for EndOfCtsObsEndStrategy*

Description

When the end strategy is not defined the cohort exit is done based on the end of continuous observation. This class is an end strategy type.

Slots

`EndOfContinuousObservation` set as true for end strategy option

ExpressionType-class *An S4 class for Expression type*

Description

An expression type quantifies the number of criteria's needed to set as restriction. Types include: All, Any, at least and at most. If the expression type is at least or at most a count is required to express the type

Slots

`Type` boolean operator for the number of items in group to include. all, any, at most and at least

`Count` the number of criteria's needed for restriction. If Type is ALL or ANY this value is NA

getACCall *Get additional criteria from cohort expression and prepare R language*

Description

Get additional criteria from cohort expression and prepare R language

Usage

```
getACCall(x)
```

Arguments

x	the circe cohort definition

Value

r language to generate the additional criteria of the cohort

getCenCall *Get censoring criteria from cohort expression and prepare R language*

Description

Get censoring criteria from cohort expression and prepare R language

Usage

```
getCenCall(x)
```

Arguments

x	the circe cohort definition

Value

r language to generate the censoring criteria of the cohort

getCohortDefinitionCall *Get call to build cohort definition*

Description

This function generates the cohort definition call and the R language calls needed to build the lower level objects for the cohort definition

Usage

```
getCohortDefinitionCall(x, nm = NULL)
```

Arguments

x	the circe cohort definition
nm	the naming convention to assign the object

Value

r language to generate the cohort

getCohortEraCall *Get cohort era from cohort expression and prepare R language*

Description

Get cohort era from cohort expression and prepare R language

Usage

```
getCohortEraCall(x)
```

Arguments

x	the circe cohort definition

Value

r language to generate the cohort era of the cohort

`getConceptCodeDetails` *Lookup Concepts by OMOP Concept Code using Vocabulary*

Description

This function looks up concepts using the OMOP concept code and vocabulary. Function requires a dbms connection to use

Usage

```
getConceptCodeDetails(
  conceptCode,
  vocabulary,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

`conceptCode`	a character vector of concept codes
`vocabulary`	a single character string with the vocabulary of the codes
`connectionDetails`	An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.
`connection`	An object of type `connection` as created using the `connect` function in the DatabaseConnector package. Can be left NULL if `connectionDetails` is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
`vocabularyDatabaseSchema`	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
`tempEmulationSchema`	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
`mapToStandard`	logic to map to standard OMOP concept

Value

a tibble data frame object with conceptId, conceptName, standardConcept, standardConceptCaption, invalidReason, invalidReasonCaption, conceptCode, domainId, vocabularyId, conceptClassId.

`getConceptIdDetails` *get concept id details*

Description

For one or more concept id, get concept id details by querying the OMOP vocabulary in the database.

Usage

```
getConceptIdDetails(
  conceptIds,
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  mapToStandard = TRUE
)
```

Arguments

`conceptIds`	a vector of concept ids
`connectionDetails`	An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.
`connection`	An object of type `connection` as created using the `connect` function in the DatabaseConnector package. Can be left NULL if `connectionDetails` is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
`vocabularyDatabaseSchema`	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
`tempEmulationSchema`	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
`mapToStandard`	logic to map to standard OMOP concept

Value

a tibble data frame object with conceptId, conceptName, standardConcept, standardConceptCaption, invalidReason, invalidReasonCaption, conceptCode, domainId, vocabularyId, conceptClassId.

getConceptSetCall *Get concept sets from cohort expression and prepare R language*

Description

This function takes the concept sets from the circe cohort definition and generates R functions to create them in the R environment. The data saved is R language to generate the objects. They are evaluated separately

Usage

```
getConceptSetCall(x)
```

Arguments

x	the circe cohort definition

Value

r language to generate the concept set expressions of the cohort

getConceptSetExpression,Component-method *Function to get Concept Set Expressions*

Description

Function to get Concept Set Expressions

Usage

```
## S4 method for signature 'Component'
getConceptSetExpression(x)
```

Arguments

x	the component to check

Value

a list of concept set expressions used in the object

getConceptSetId,ConceptSetExpression-method *Function to find the ConceptSetId*

Description

Function to find the ConceptSetId

Usage

```
## S4 method for signature 'ConceptSetExpression'
getConceptSetId(x)

## S4 method for signature 'Query'
getConceptSetId(x)
```

Arguments

x	the component to check

Value

the id from the conceptset expression

getESCall *Get end strategy from cohort expression and prepare R language*

Description

Get end strategy from cohort expression and prepare R language

Usage

```
getESCall(x)
```

Arguments

x	the circe cohort definition

Value

r language to generate the end strategy of the cohort

getIRSCall — *Get inclusion rules from cohort expression and prepare R language*

Description

Get inclusion rules from cohort expression and prepare R language

Usage

```
getIRSCall(x)
```

Arguments

x	the circe cohort definition

Value

r language to generate the inclusion rules of the cohort

getPCCall — *Get primary criteria from cohort expression and prepare R language*

Description

Get primary criteria from cohort expression and prepare R language

Usage

```
getPCCall(x)
```

Arguments

x	the circe cohort definition

Value

r language to generate the primary criteria of the cohort

Group-class — *An S4 class for Group*

Description

TODO clarify the description of a group. A group that bundles criteria together identifying an event

Slots

`Type` a expression type class Boolean for the number of items to make the group count
`CriteriaList` a list of items (counts and queries) that would identify a medical event
`DemographicCriteriaList` a list of demographic attributes that could identify a population
`Groups` a list of other groups that are contained within a group

Limit-class *An S4 class for Limit*

Description

A class designating a limit of events per person Types include: all first last

Slots

`Type` how to limit events per person: all, first, or last

lineBreak *Print a line break*

Description

Print a line break

Usage

```
lineBreak(t = c(1, 2, 3, 4))
```

Arguments

`t`	A number from 1 to 4 representing the type of line break

Value

Prints a line break. Does not return a value.

listAttributeOptions *List Attribute options*

Description

List Attribute options

Usage

```
listAttributeOptions(domain = NULL)
```

Arguments

`domain`	the attribute options within the domain, default is NULL then all options printed

Value

A dataframe with the list of options for attributes we can use specified per domain.

loadComponent *Function to load component*

Description

This function loads the component from a json file to its s4 componentclass

Usage

```
loadComponent(path)
```

Arguments

path	a path to the file we wish to load

Value

returns a component

LogicAttribute-class *An S4 class for Logic Attribute*

Description

This class creates a logic attribute which says either true or false if the name of the attribute is maintained

Slots

Name a name of the attribute

Logic TRUE or FALSE for this attribute

lookupKeyword *Lookup concept name as a general search*

Description

This function looks up concepts based on the concept name. It can be modified to conduct an exact name search or general search that contains the concept name in the concept.

Usage

```
lookupKeyword(
  keyword,
  searchType = c("like", "exact", "any"),
  connectionDetails = NULL,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL
)
```

Arguments

keyword	a character string used to search OMOP concepts
searchType	options to aid search. Can use like match, exact match or any match
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.

Value

a tibble data frame object with conceptId, conceptName, standardConcept, standardConceptCaption, invalidReason, invalidReasonCaption, conceptCode, domainId, vocabularyId, conceptClassId.

---

mapOperator *map the operator among options*

Description

map the operator among options

Usage

```
mapOperator(op)
```

Arguments

op	the operator input we want to map

Value

the circe op

`MetaData-class` *An S4 class for Component MetaData TODO confirm possible values for ComponentType. Should Index be included as a slot?*

Description

An S4 class for Component MetaData TODO confirm possible values for ComponentType. Should Index be included as a slot?

Slots

`ComponentType` name of component class (this is formally defined) Possible values are...

`Name` name for component customized by user

`Description` description of the component

`Index` A character string either IndexStartDate or IndexEndDate Identifies where the index is relative to the window

`ObservationWindow-class` *An S4 class for ObservationWindow*

Description

A class designating an amount of time necessary for an initial event to be recorded

Slots

`PriorDays` minimal amount of time before event for it to be recorded

`PostDays` minimal amount of time after an event for it to be recorded

`Occurrence-class` *An S4 class for Occurrence*

Description

The Occurrence class provides logic on the number of criterias that most be true in a person for them to be contained in the expression

Slots

`Type` a character string of either at most, at least, or exactly providing context to the number of occurrences

`Count` an integer value that provides the number of occurrences

`isDistinct` a logic toggle where if TRUE only counts distinct occurrences

OpAttribute-class *An S4 class for an Op Attribute*

Description

An operator attribute meaning it has some value with a boolean operator

Slots

`Name` the name of the attribute

`Op` the operator gt,lt,gte,lte,eq,neq,bt,!bt

`Contents` the contents of the attribute as a list. includes the value and the extent

Query-class *An S4 class for a Query*

Description

TODO clarify description of a Query A query is a medical concept that can be extracted from a database through a 'where' clause in a SQL statement. This includes concepts. (?)

Slots

`Domain` the domain where the concepts can be found

`CodesetId` the id that matches the concept set expression

`Attributes` a list of attributes that modify the query with more information

readInCirce *Function to read in a circe json*

Description

This function reads a circe json an builds the cohort definition in an execution space

Usage

```
readInCirce(
  jsonPath,
  connectionDetails,
  connection = NULL,
  vocabularyDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  returnHash = FALSE
)
```

Arguments

jsonPath	a path to the file we wish to import
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary format resides. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
returnHash	if true returns a has table with all components necessary to build the cohort definition including the cohort definition

Value

returns the cohort definition

---

removeDupCSE *Function that removes duplicate concept set expressions*

Description

Function that removes duplicate concept set expressions

Usage

```
removeDupCSE(cse)
```

Arguments

cse	the list of concept set expressions used in the object

Value

a list of concept set expressions free of duplicates

saveComponent *Function to save component*

Description

This function saves the component as a json file. The component is converted from s4 to s3 to fit the jsonlite function

Usage

```
saveComponent(x, saveName, savePath = getwd())
```

Arguments

x	the component you wish to save
saveName	a name for the function you want to save
savePath	a path to a file to save. Default is the active working directory

Value

no return in r. json file written to a save point

saveState,Concept-method *Save State for components*

Description

These function coerce s4 CAPR objects to s3 so that they are in a json save state

Usage

```
## S4 method for signature 'Concept'
saveState(x)

## S4 method for signature 'ConceptSetItem'
saveState(x)

## S4 method for signature 'ConceptSetExpression'
saveState(x)

## S4 method for signature 'OpAttribute'
saveState(x)

## S4 method for signature 'SourceConceptAttribute'
saveState(x)

## S4 method for signature 'ConceptAttribute'
saveState(x)
```

```
## S4 method for signature 'CorrelatedCriteriaAttribute'
saveState(x)

## S4 method for signature 'LogicAttribute'
saveState(x)

## S4 method for signature 'Window'
saveState(x)

## S4 method for signature 'Timeline'
saveState(x)

## S4 method for signature 'Occurrence'
saveState(x)

## S4 method for signature 'ExpressionType'
saveState(x)

## S4 method for signature 'ObservationWindow'
saveState(x)

## S4 method for signature 'Limit'
saveState(x)

## S4 method for signature 'Query'
saveState(x)

## S4 method for signature 'Count'
saveState(x)

## S4 method for signature 'Group'
saveState(x)

## S4 method for signature 'MetaData'
saveState(x)

## S4 method for signature 'DateOffsetEndStrategy'
saveState(x)

## S4 method for signature 'CustomEraEndStrategy'
saveState(x)

## S4 method for signature 'EndOfCtsObsEndStrategy'
saveState(x)

## S4 method for signature 'CollapseSettings'
saveState(x)

## S4 method for signature 'CensorWindow'
saveState(x)
```

```
## S4 method for signature 'Component'
saveState(x)
```

Arguments

x	a criteria class object in s4

Value

the object converted to s3 to be saved as a json object

---

show,Window-method *Show statements of capr objects*

Description

These functions print the capr object to console in a readable format

Usage

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

## S4 method for signature 'Timeline'
show(object)

## S4 method for signature 'Occurrence'
show(object)

## S4 method for signature 'ObservationWindow'
show(object)

## S4 method for signature 'Group'
show(object)

## S4 method for signature 'Query'
show(object)

## S4 method for signature 'Count'
show(object)

## S4 method for signature 'OpAttribute'
show(object)

## S4 method for signature 'Concept'
show(object)

## S4 method for signature 'ConceptSetItem'
show(object)

## S4 method for signature 'ConceptSetExpression'
```

```
show(object)

## S4 method for signature 'Limit'
show(object)

## S4 method for signature 'MetaData'
show(object)

## S4 method for signature 'Component'
show(object)

## S4 method for signature 'CohortDetails'
show(object)

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

Arguments

object	the object to show

Value

a console print of the object

SourceConceptAttribute-class *An S4 class for SourceConceptAttribute*

Description

An attribute that looks at utilizing the source concepts instead of standard concepts

Slots

Name name of the attribute

SourceCodesetId a source concept id, connection to concept set expression

---

Timeline-class *An S4 class for Timeline*

Description

The timeline class provides context to when the criteria must be observed in a person timeline to pretain to the expression

Slots

`StartWindow` a window class object identifying the start window

`EndWindow` a window class object ifentifying the end window (optional)

`RestrictVisit` a logic toggle where TRUE restricts to the same visit

`IgnoreObservationPeriod` a logic toggle where TRUE allows events outside the observation period

---

`toggleConceptMapping` *Toggle the concept mapping for select positions*

Description

This functions changes the logical object (TRUE or FALSE) to its other state. This helps toggle the concept mapping for a select set in a large list

Usage

```
toggleConceptMapping(
  conceptMapping,
  pos,
  mapping = c("includeDescendants", "isExcluded", "includeMapped")
)
```

Arguments

`conceptMapping`	the conceptMapping object
`pos`	the positions to toggle
`mapping`	select the mapping type to toggle at each position

Value

This function returns a list for concept mapping for the concept set expression

---

`UpdateAndConvert` *A function to update codeset Ids and convert to circe*

Description

A function to update codeset Ids and convert to circe

Usage

```
UpdateAndConvert(x, conceptTable)
```

Arguments

`x`	the object to update and convert
`conceptTable`	a merge table to match guid to codeset id integer

Value

an object with updated codeset id

`UpdateCirceCodesetId,SourceConceptAttribute-method` *Change CodesetId to Integer*

Description

When creating the circe json object, an internal reference system needs to be established for the concept set expressions. This function will update the concept ids from its guid to the ordering of the ids in a merge table. The codeset Ids will be integers starting from 0 in the circe instance.

Usage

```
## S4 method for signature 'SourceConceptAttribute'
UpdateCirceCodesetId(x, conceptTable)

## S4 method for signature 'Query'
UpdateCirceCodesetId(x, conceptTable)

## S4 method for signature 'Count'
UpdateCirceCodesetId(x, conceptTable)

## S4 method for signature 'Group'
UpdateCirceCodesetId(x, conceptTable)

## S4 method for signature 'CustomEraEndStrategy'
UpdateCirceCodesetId(x, conceptTable)
```

Arguments

`x`	a component class object in s4
`conceptTable`	a merge table to match guid to codeset id integer

Value

an object with updated codeset id

`UpdateCodesetIdRule` *Update codeset id for inclusion rule*

Description

Update codeset id for inclusion rule

Usage

```
UpdateCodesetIdRule(x, conceptTable)
```

Arguments

x	the group that need to update codeset Ids
conceptTable	a merge table to match guid to codeset id integer

Value

an object with updated codeset id

Window-class *An S4 class for a Window*

Description

A window class provides details on the end points of the timeline

Slots

Event a character string either EventStarts or EventEnds. Identifies the point of reference for the window

Start a list containing the days and coefficient for the start of the window

End A list containing the days and coefficient for the end of the window

Index A character string either IndexStartDate or IndexEndDate Identifies where the index is relative to the window

---

writeCaprCall *Function to write capr calls from a circe json*

Description

This function writes the CAPR calls used to build the cohort definition defined in the circe json . The ouput is a txt file with executable R language

Usage

```
writeCaprCall(jsonPath, txtPath)
```

Arguments

jsonPath	a path to the file we wish to import
txtPath	a path to the txt file we wish to save

Value

no return but saves the CAPR calls to build a cohort in a txt file

Index