

# Package ‘Capr’

March 14, 2023

**Title** Cohort Definition Application Programming in R

**Version** 2.0.0

## **Description**

The Capr package provides a programming language for defining OHDSI cohort definitions in R. The functions in Capr allow for the programmatic creation of OHDSI concept sets and cohorts that can be serialized to Atlas/CIRCE-BE compatible json files or to OHDSI-SQL. Capr functions can be used to create, save, and load component parts to a cohort definition allowing R programmers to easily reuse cohort logic. Capr provides tools to create a large number of OHDSI cohorts programmatically while also helping bridge the gap between human readable descriptions of clinical phenotypes and their computational implementation.

**License** Apache License 2.0

**URL** <https://ohdsi.github.io/Capr>, <https://github.com/OHDSI/Capr>

**BugReports** <https://github.com/OHDSI/Capr/issues>

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Depends** R (>= 3.5.0),

**Imports** magrittr (>= 1.5.0),

jsonlite,  
methods,  
purrr,  
rlang,  
dplyr,  
checkmate,  
tibble,  
readr,  
stringr,  
glue,  
cli,  
digest,  
fs,  
lubridate,  
DBI,  
DatabaseConnector,  
SqlRender

**Suggests** testthat (>= 3.0.0),  
 knitr,  
 rmarkdown,  
 CohortGenerator,  
 CirceR,  
 Eunomia

**VignetteBuilder** knitr

**Additional\_repositories** <https://OHDSI.github.io/drat>

**Config/testthat/edition** 3

**Collate** 'Capr.R'  
 'conceptSet.R'  
 'attributes-concept.R'  
 'attributes-logic.R'  
 'query.R'  
 'window.R'  
 'criteria.R'  
 'attributes-nested.R'  
 'attributes-op.R'  
 'exit.R'  
 'cohort.R'  
 'collectCodesetId.R'  
 'utils.R'

## R topics documented:

==	4
age	4
as.data.frame,ConceptSet-method	5
as.json	5
atLeast	6
atMost	6
attrition	7
bt	7
CensoringCriteria-class	8
censoringEvents	8
cohort	8
Concept-class	9
conceptAttribute-class	9
ConceptSet-class	9
ConceptSetItem-class	10
condition	10
conditionEra	11
continuousObservation	11
Criteria-class	11
cs	12
daysOfSupply	13
death	13
drug	14
drugEra	14
drugExit	15

DrugExposureExit-class	15
drugQuantity	16
drugRefills	16
duringInterval	17
endDate	17
Endpoint-class	18
entry	18
eq	19
era	19
EventAperture-class	20
eventEnds	20
eventStarts	21
EventWindow-class	21
exactly	22
exit	22
firstOccurrence	22
FixedDurationExit-class	23
fixedExit	23
generateCaprTemplate	23
getConceptSetCall	24
getConceptSetDetails	24
Group-class	25
gt	25
gte	26
logicAttribute-class	26
lt	27
lte	27
male	28
measurement	28
nbt	29
nestedAttribute-class	29
nestedWithAll	30
nestedWithAny	30
nestedWithAtLeast	30
nestedWithAtMost	31
observation	31
observationExit	31
ObservationExit-class	32
ObservationWindow-class	32
Occurrence-class	32
opAttributeDate-class	33
opAttributeInteger-class	33
opAttributeNumeric-class	33
procedure	34
Query-class	34
rangeHigh	34
rangeLow	35
readConceptSet	35
startDate	36
toCirce	36
unit	36
valueAsNumber	37

visit . . . . .	37
withAll . . . . .	38
withAny . . . . .	38
withAtLeast . . . . .	38
withAtMost . . . . .	39
writeCohort . . . . .	39
writeConceptSet . . . . .	40
<b>Index</b>	<b>41</b>

---

<code>==</code>	<i>FUnction checks if two concept set class objects are equivalent</i>
-----------------	--

---

**Description**

FUnction checks if two concept set class objects are equivalent

**Usage**

```
## S4 method for signature 'ConceptSet,ConceptSet'
e1 == e2
```

**Arguments**

e1, e2                    a ConceptSet Class object

---

<code>age</code>	<i>Function to create age attribute</i>
------------------	---

---

**Description**

Function to create age attribute

**Usage**

```
age(op)
```

**Arguments**

op                    an opAttribute object that is either numeric or integer that defines the logical operation used to determine eligible patient age

---

`as.data.frame, ConceptSet-method`*Coerce a concept set expression to a dataframe*

---

**Description**

Coerce a concept set expression to a dataframe

**Usage**

```
## S4 method for signature 'ConceptSet'  
as.data.frame(x)
```

**Arguments**

x                      A Caper Concept Set

**Value**

A tibble (dataframe) with columns: concept\_id, includeDescendants, isExcluded, includeMapped.

---

`as.json`*Coerce Capr object to json*

---

**Description**

Coerce Capr object to json

**Usage**

```
as.json(x, pretty = TRUE, ...)  
  
## S4 method for signature 'Cohort'  
as.json(x, pretty = TRUE, ...)
```

**Arguments**

x                      the capr object  
pretty                a toggle to make the json look nice, part of jsonlite  
...                    additional arguments passes to jsonlite::toJSON

---

atLeast	<i>Function to enumerate an minimal count of occurrences</i>
---------	--

---

### Description

Function to enumerate an minimal count of occurrences

### Usage

```
atLeast(x, query, aperture = duringInterval(eventStarts(-Inf, Inf)))
```

### Arguments

x	the integer counting the number of occurrences
query	a query object that provides context to the clinical event of interest
aperture	an eventAperture object that shows the temporal span where the event is to be observed relative to the index event

---

atMost	<i>Function to enumerate a maximum count of occurrences</i>
--------	---

---

### Description

Function to enumerate a maximum count of occurrences

### Usage

```
atMost(x, query, aperture = duringInterval(eventStarts(-Inf, Inf)))
```

### Arguments

x	the integer counting the number of occurrences
query	a query object that provides context to the clinical event of interest
aperture	an eventAperture object that shows the temporal span where the event is to be observed relative to the index event

---

attrition	<i>Create a cohort attrition object</i>
-----------	---

---

### Description

Create a cohort attrition object

### Usage

```
attrition(..., expressionLimit = c("First", "All", "Last"))
```

### Arguments

...	Capr groups
expressionLimit	how to limit initial events per person either First, All, or Last

---

bt	<i>Between operator</i>
----	-------------------------

---

### Description

function that builds an opAttribute based on between logic

### Usage

```
bt(x, y)

## S4 method for signature 'integer'
bt(x, y)

## S4 method for signature 'numeric'
bt(x, y)

## S4 method for signature 'Date'
bt(x, y)
```

### Arguments

x	the left side bound of the between logic This can either be an integer, numeric, or Date data type. Different data types will return the appropriate opAttribute type
y	the right side bound of the between logic. This can either be an integer, numeric, or Date data type. Different data types will return the appropriate opAttribute type

---

CensoringCriteria-class

*An S4 class identifying a censoring criteria for the cohort*


---

### Description

The censoring criteria specifies events where the person exits the cohort. These events are based on a query class object and users can specify multiple queries in the censoring criteria.

### Slots

`criteria` a list of Capr query class objects that specify the events that would lead a person to exit the cohort.

---

censoringEvents

*Constructor for a set of censoring events*


---

### Description

Constructor for a set of censoring events

### Usage

```
censoringEvents(...)
```

### Arguments

... a list of Capr query objects that are used as censoring events

---

cohort

*Function that creates a cohort object*


---

### Description

Function that creates a cohort object

### Usage

```
cohort(entry, attrition = NULL, exit = NULL, era = NULL)
```

### Arguments

<code>entry</code>	the index event of the cohort
<code>attrition</code>	rules that restrict the cohort further, developing attrition
<code>exit</code>	the event where the person exits the cohort
<code>era</code>	Cohort era (collapse) logic created with the 'cohortEra' function



---

Concept-class	<i>An S4 class for a single OMOP Concept</i>
---------------	--

---

**Description**

A concept class contains all the information about the concept from the OMOP vocabulary.

**Slots**

concept\_id the OMOP/OHDSI concept ID  
 concept\_name the name of the concept  
 standard\_concept whether the concept is standard 'S', classification 'C', or non-standard NA  
 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 (e.g. Drug, Condition, Procedure, etc)  
 vocabulary\_id the name of the vocabulary  
 concept\_class\_id type of concept class

---

conceptAttribute-class	<i>An S4 class for a concept attribute</i>
------------------------	--

---

**Description**

An S4 class for a concept attribute

**Slots**

name the name of the attribute  
 conceptSet a list representing the concepts for the attribute

---

ConceptSet-class	<i>An S4 class for ConceptSetExpresion</i>
------------------	--

---

**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	<i>An S4 class for ConceptSetItem</i>
----------------------	---------------------------------------

---

**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

---

condition	<i>Query the condition domain</i>
-----------	-----------------------------------

---

**Description**

Query the condition domain

**Usage**

condition(conceptSet, ...)

**Arguments**

- conceptSet A condition concept set
- ... optional attributes

**Value**

A Capr Query

---

conditionEra	<i>Query the condition era domain</i>
--------------	---------------------------------------

---

**Description**

Query the condition era domain

**Usage**

```
conditionEra(conceptSet, ...)
```

**Arguments**

conceptSet	A condition concept set
...	optional attributes

**Value**

A Capr Query

---

continuousObservation	<i>A function to construct the observationWindow</i>
-----------------------	--

---

**Description**

A function to construct the observationWindow

**Usage**

```
continuousObservation(priorDays = 0L, postDays = 0L)
```

**Arguments**

priorDays	minimum number of observation days prior to the cohort index. Default 0 days
postDays	minimum number of observation days post cohort index. Default 0 days

---

Criteria-class	<i>An S4 for a criteria</i>
----------------	-----------------------------

---

**Description**

a criteria is a temporal observation of a clinical event relative to the index event

**Slots**

occurrence	an occurrence object specifying how many events must occur to consider the event as part of the cohort definition
query	a query object that provides context to the clinical event of interest
aperture	an eventAperture object that shows the temporal span where the event is to be observed relative to the index event

---

cs	Create a concept set
----	----------------------

---

### Description

cs is used to create concept set expressions.

‘exclude’ is meant to be used inside ‘cs’ when creating a new concept set.

‘mapped’ is meant to be used inside ‘cs’ when creating a new concept set.

‘descendants’ is meant to be used inside ‘cs’ when creating a new concept set.

### Usage

```
cs(..., name = "", id = NULL)
```

```
exclude(...)
```

```
mapped(...)
```

```
descendants(...)
```

### Arguments

...	One or more numeric vectors that can be coerced to integers, or Calls to helper functions "exclude", "descendants", or "mapped".
name	A name for the concept set
id	An id for the concept set

### Value

A Capr Concept Set Object

A list of Capr concepts

A list of Capr concepts

A list of Capr concepts

### Functions

- `exclude()`: exclude concepts
- `mapped()`: Include mapped concepts
- `descendants()`: Include descendants

### Examples

```
## Not run:
cs(1, 2)
cs(1, c(1, 10, 2))
cs(1, seq(2, 10, 2))
cs(1, 2, 3, exclude(4, 5))
cs(1, 2, 3, exclude(4, 5), mapped(6, 7))
cs(1, 2, 3, exclude(4, 5), mapped(6, 7), descendants(8, 9))
```

```
cs(descendants(1, 2, 3), exclude(descendants(8, 9)))  
  
## End(Not run)
```

---

daysOfSupply	<i>Function to create days supply attribute</i>
--------------	---

---

**Description**

This function is used only for a drug query. days supply is a column in the drug exposure table of the cdm. This attribute allows a subquery to find drugs that satisfy certain values determined by the op logic.

**Usage**

```
daysOfSupply(op)
```

**Arguments**

op	an opAttribute object that is either numeric or integer that defines the logical operation used to determine eligible number of days of supply
----	--

---

death	<i>Query the condition era domain</i>
-------	---------------------------------------

---

**Description**

Query the condition era domain

**Usage**

```
death(conceptSet = NULL, ...)
```

**Arguments**

conceptSet	A condition concept set
...	optional attributes

**Value**

A Capr Query

---

drug	<i>Query the drug domain</i>
------	------------------------------

---

**Description**

Query the drug domain

**Usage**

drug(conceptSet, ...)

**Arguments**

conceptSet	A drug concept set
...	optional attributes

**Value**

A Capr Query

---

drugEra	<i>Query the drug era domain</i>
---------	----------------------------------

---

**Description**

Query the drug era domain

**Usage**

drugEra(conceptSet, ...)

**Arguments**

conceptSet	A drug ingredient concept set
...	optional attributes

**Value**

A Capr Query

---

drugExit	<i>Function to create an exit based on exit based on the end of a continuous drug exposure</i>
----------	--

---

## Description

Function to create an exit based on exit based on the end of a continuous drug exposure

## Usage

```
drugExit(
  conceptSet,
  persistenceWindow = 0L,
  surveillanceWindow = 0L,
  daysSupplyOverride = NULL
)
```

## Arguments

conceptSet	the concept set of the drug exposure used to identify the exit
persistenceWindow	allow for a maximum of days between exposure records when inferring the era of persistence exposure
surveillanceWindow	add days to the end of the era of persistence exposure as an additional period of surveillance prior to cohort exit
daysSupplyOverride	force drug exposure days supply to a set number of days

---

DrugExposureExit-class

*An S4 class for a cohort exit based on continuous exposure persistence.*

---

## Description

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 specified 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.

**Slots**

conceptSet the concept set of the drug exposure used to identify the exit

persistenceWindow allow for a maximum of days between exposure records when inferring the era of persistence exposure

surveillanceWindow add days to the end of the era of persistence exposure as an additional period of surveillance prior to cohort exit

daysSupplyOverride force drug exposure days supply to a set number of days

count an integer specifying the number of occurrences for a criteria

---

drugQuantity	<i>Function to create quantity attribute</i>
--------------	--

---

**Description**

This function is used only for a drug query. quantity is a column in the drug exposure table of the cdm. This attribute allows a subquery to find drugs that satisfy certain values determined by the op logic.

**Usage**

drugQuantity(op)

**Arguments**

op	an opAttribute object that is either numeric or integer that defines the logical operation used to determine eligible quantity
----	--

---

drugRefills	<i>Function to create refills attribute</i>
-------------	---

---

**Description**

This function is used only for a drug query. refills is a column in the drug exposure table of the cdm. This attribute allows a subquery to find drugs that satisfy certain values determined by the op logic.

**Usage**

drugRefills(op)

**Arguments**

op	an opAttribute object that is either numeric or integer that defines the logical operation used to determine eligible number of refills
----	---



---

duringInterval	<i>Function that creates an eventAperture an opening where an event can occur relative to the index event</i>
----------------	---

---

### Description

Function that creates an eventAperture an opening where an event can occur relative to the index event

### Usage

```
duringInterval(
  startWindow,
  endWindow = NULL,
  restrictVisit = FALSE,
  ignoreObservationPeriod = FALSE
)
```

### Arguments

startWindow	the starting window where an event can occur
endWindow	the end window of where an event can occur. This parameter is optional
restrictVisit	a logical toggle specifying whether the event should occur on the same visit
ignoreObservationPeriod	a logical toggle specifying whether we can consider events outside the observation period

---

endDate	<i>Function that creates a end date attribute</i>
---------	---

---

### Description

Function that creates a end date attribute

### Usage

```
endDate(op)
```

### Arguments

op	an opAttribute object must be a date that defines the logical operation used to determine eligible end dates
----	--

---

Endpoint-class	<i>An S4 class for an Endpoint</i>
----------------	------------------------------------

---

**Description**

this determines the time in days relative to an index either before or after

**Slots**

days either a character string all or an integer for the number of days

coeff a character string either before or after

---

entry	<i>Create a cohort entry criteria</i>
-------	---------------------------------------

---

**Description**

Create a cohort entry criteria

**Usage**

```
entry(
  ...,
  observationWindow = continuousObservation(0L, 0L),
  primaryCriteriaLimit = c("First", "All", "Last"),
  additionalCriteria = NULL,
  qualifiedLimit = c("First", "All", "Last")
)
```

**Arguments**

...	Capr Queries
observationWindow	a time specifying the minimal time a person is observed
primaryCriteriaLimit	Which primary criteria matches should be considered for inclusion? "First", "Last" or "All"
additionalCriteria	a Capr group that adds restriction to the entry event
qualifiedLimit	Which criteria matches should be considered for inclusion? "First", "Last" or "All"

**Value**

A cohort entry Capr object

---

eq	<i>Equal to operator</i>
----	--------------------------

---

### Description

function that builds an opAttribute based on equal to logic

### Usage

```
eq(x)

## S4 method for signature 'integer'
eq(x)

## S4 method for signature 'numeric'
eq(x)

## S4 method for signature 'Date'
eq(x)
```

### Arguments

x	the value to used as a bound in the op logic. This can either be an integer, numeric, or Date data type. Different data types will return the appropriate opAttribute type
---	--

---

era	<i>Create a Cohort Era class object</i>
-----	---

---

### 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

```
era(eraDays = 0L, studyStartDate = NULL, studyEndDate = NULL)
```

### Arguments

eraDays	a numeric that specifies the number of days for the era padding
studyStartDate	a date string that specifies the starting date of registration
studyEndDate	a date string that specifies the end date of registration

---

EventAperture-class	<i>An S4 class for Aperture</i>
---------------------	---------------------------------

---

### Description

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

### Slots

startWindow a EventWindow class object identifying the start window

endWindow a EventWindow class object identifying 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

---

eventEnds	<i>Function creates an event window where the event ends</i>
-----------	--

---

### Description

Function creates an event window where the event ends

### Usage

```
eventEnds(a, b, index = c("startDate", "endDate"))
```

### Arguments

a the left side of the event window

b the right side of the event window

index specifying what part of the index we start looking for events either at the index start date or index enddate

---

```
eventStarts      #' A function to offset the number of days relative to index #' @param
                  days a number specifying the number of days to offset from in-
                  dex where #' an event may be observed. In this function a neg-
                  ative number means days before index #' and a postive number
                  means days after index. #' @export offset <- function(days) co-
                  eff <- dplyr::if_else(sign(days) == 1, "after", "before", "before")
                  new("Endpoint", days = as.integer(abs(days)), coeff = coeff)
```

---

## Description

```
#' Function looking at all time before an event #' @export allDaysBefore <- function() new("Endpoint",
days = "all", coeff = "before")
```

## Usage

```
eventStarts(a, b, index = c("startDate", "endDate"))
```

## Arguments

a	the left side of the event window
b	the right side of the event window
index	specifying what part of the index we start looking for events either at the index start date or index enddate

## Details

```
#' Function looking at all time after an event #' @export allDaysAfter <- function() new("Endpoint",
days = "all", coeff = "after")
```

Function creates an event window where the event starts

---

EventWindow-class	<i>An S4 class for a Window</i>
-------------------	---------------------------------

---

## Description

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

## Slots

event	a character string either start or end. Identifies the point of reference for the window
start	an endpoint object containing the days and coefficient for the start of the window
end	an endpoint object containing the days and coefficient for the end of the window
index	A character string either start or end. Identifies where the index is relative to the window

---

exactly	<i>Function to enumerate an exact count of occurrences</i>
---------	--

---

### Description

Function to enumerate an exact count of occurrences

### Usage

```
exactly(x, query, aperture = duringInterval(eventStarts(-Inf, Inf)))
```

### Arguments

x	the integer counting the number of occurrences
query	a query object that provides context to the clinical event of interest
aperture	an eventAperture object that shows the temporal span where the event is to be observed relative to the index event

---

exit	<i>Function that creates a cohort exit object</i>
------	---

---

### Description

Function that creates a cohort exit object

### Usage

```
exit(endStrategy, censor = NULL)
```

### Arguments

endStrategy	the endStrategy object to specify for the exit
censor	the censoring criteria to specify for the exit

---

firstOccurrence	<i>Add first occurrence attribute</i>
-----------------	---------------------------------------

---

### Description

Add first occurrence attribute

### Usage

```
firstOccurrence()
```

### Value

An attribute that can be used in a query function

---

FixedDurationExit-class

*An S4 class for a cohort exit based on fixed duration persistence.*


---

### Description

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.

### Slots

index specification of event date to offset

offsetDays an integer specifying the number of days to offset from the event date

---

fixedExit

*Function to create an exit based on exit based on the end of a continuous drug exposure*


---

### Description

Function to create an exit based on exit based on the end of a continuous drug exposure

### Usage

```
fixedExit(index = c("startDate", "endDate"), offsetDays)
```

### Arguments

index specification of event date to offset. Can be either startDate or endDate

offsetDays an integer specifying the number of days to offset from the event date

---

generateCaprTemplate

*Generate a Capr cohort using a template*


---

### Description

Generate a Capr cohort using a template

### Usage

```
generateCaprTemplate(file, .capr)
```

### Arguments

file the input file of a concept set

.capr a function that creates a capr cohort

---

getConceptSetCall	<i>Create the Capr code to build a concept set</i>
-------------------	--

---

### Description

Create the Capr code to build a concept set

### Usage

```
getConceptSetCall(x, name = x@Name)
```

### Arguments

x	A concept set
name	the name of the concept set

### Value

The Capr code required to build the concept set

---

getConceptSetDetails	<i>Fill in Concept Set details using a vocab</i>
----------------------	--

---

### Description

Concept sets created in R using the 'cs' function do not contain details like "CONCEPT\_NAME", "DOMAIN\_ID", etc. If an OMOP CDM vocabulary is available then these details can be filled in by the the 'getConceptSetDetails' function.

### Usage

```
getConceptSetDetails(x, con, vocabularyDatabaseSchema = NULL)
```

### Arguments

x	A concept set created by 'cs'
con	A connection to an OMOP CDM database
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'.

### Value

A modified version of the input concept set with concept details filled in.



**Examples**

```
## Not run:
# create a concept set
vocabularyDatabaseSchema = "cdm5"
anemia <- cs(descendants(439777,4013073,4013074))

# fill in the details from an OMOP CDM
library(DatabaseConnector)
con <- connect(dbms = "postgresql", user = "postgres", password = "", server = "localhost/covid")
anemia <- getConceptSetDetails(condition_anemia, con, vocabularyDatabaseSchema = "cdm5")

## End(Not run)
```

---

Group-class	<i>An S4 class for a group</i>
-------------	--------------------------------

---

**Description**

a group is the combination of multiple criteria or sub groups

**Slots**

occurrence an occurrence object specifying how many events must occur to consider the event as part of the cohort definition

criteria a list of criteria that are grouped together

group a list of sub-groups to consider

---

gt	<i>Greater than operator</i>
----	------------------------------

---

**Description**

function that builds an opAttribute based on greater than logic

**Usage**

```
gt(x)

## S4 method for signature 'integer'
gt(x)

## S4 method for signature 'numeric'
gt(x)

## S4 method for signature 'Date'
gt(x)
```

**Arguments**

x	the value to used as a bound in the op logic. This can either be an integer, numeric, or Date data type. Different data types will return the appropriate opAttribute type
---	--

---

gte	<i>Greater than or equal to operator</i>
-----	--

---

### Description

function that builds an opAttribute based on greater than or equal to logic

### Usage

```
gte(x)

## S4 method for signature 'integer'
gte(x)

## S4 method for signature 'numeric'
gte(x)

## S4 method for signature 'Date'
gte(x)
```

### Arguments

x	the value to used as a bound in the op logic. This can either be an integer, numeric, or Date data type. Different data types will return the appropriate opAttribute type
---	--

---

logicAttribute-class	<i>An S4 class for a logical attribute</i>
----------------------	--

---

### Description

with a logic attribute if it is specified than we assume it is true

### Slots

name	the name of the attribute
------	---------------------------

---

lt	<i>Less than operator</i>
----	---------------------------

---

**Description**

function that builds an opAttribute based on less than logic

**Usage**

```
lt(x)

## S4 method for signature 'integer'
lt(x)

## S4 method for signature 'numeric'
lt(x)

## S4 method for signature 'Date'
lt(x)
```

**Arguments**

x	the value to used as a bound in the op logic. This can either be an integer, numeric, or Date data type. Different data types will return the appropriate opAttribute type
---	--

---

lte	<i>Less than or equal to operator</i>
-----	---------------------------------------

---

**Description**

function that builds an opAttribute based on less than or equal to than logic

**Usage**

```
lte(x)

## S4 method for signature 'integer'
lte(x)

## S4 method for signature 'numeric'
lte(x)

## S4 method for signature 'Date'
lte(x)
```

**Arguments**

x	the value to used as a bound in the op logic. This can either be an integer, numeric, or Date data type. Different data types will return the appropriate opAttribute type
---	--

---

male	<i>Add male attribute to a query</i>
------	--------------------------------------

---

### Description

Add male attribute to a query  
 Add female attribute to a query

### Usage

```
male()

female()
```

### Value

An attribute that can be used in a query function  
 An attribute that can be used in a query function

### Functions

- `male()`: male demographic attribute
- `female()`: female demographic attribute

### Examples

```
## Not run:
# Create a cohort of males with Type 1 diabetes
t1dm <- cs(descendants(201254,435216,40484648))
t1dm_males <- cohort(condition(t1dm, male()))

## End(Not run)
## Not run:
# Create a cohort of males with Type 1 diabetes
t1dm <- cs(descendants(201254,435216,40484648))
t1dm_females <- cohort(condition(t1dm, female()))

## End(Not run)
```

---

measurement	<i>Query the measurement domain</i>
-------------	-------------------------------------

---

### Description

Query the measurement domain

### Usage

```
measurement(conceptSet, ...)
```

**Arguments**

conceptSet	A measurement concept set
...	optional attributes

**Value**

A Capr Query

---

nbt	<i>Not between operator</i>
-----	-----------------------------

---

**Description**

function that builds an opAttribute based on not between logic

**Usage**

```
nbt(x, y)

## S4 method for signature 'integer'
nbt(x, y)

## S4 method for signature 'numeric'
nbt(x, y)

## S4 method for signature 'Date'
nbt(x, y)
```

**Arguments**

x	the left side bound of the between logic This can either be an integer, numeric, or Date data type. Different data types will return the appropriate opAttribute type
y	the right side bound of the between logic. This can either be an integer, numeric, or Date data type. Different data types will return the appropriate opAttribute type

---

nestedAttribute-class	<i>An S4 class for a nested attribute</i>
-----------------------	---

---

**Description**

An S4 class for a nested attribute

**Slots**

name	the name of the attribute
conceptSet	a list representing the concepts for the attribute

---

nestedWithAll	<i>Function to construct a nested group where all criteria and groups must be satisfied</i>
---------------	---

---

**Description**

Function to construct a nested group where all criteria and groups must be satisfied

**Usage**

```
nestedWithAll(...)
```

**Arguments**

...	a set of criteria or groups
-----	-----------------------------

---

nestedWithAny	<i>Function to construct a nested group where any criteria and groups may be satisfied</i>
---------------	--

---

**Description**

Function to construct a nested group where any criteria and groups may be satisfied

**Usage**

```
nestedWithAny(...)
```

**Arguments**

...	a set of criteria or groups
-----	-----------------------------

---

nestedWithAtLeast	<i>Function to construct a nested group where at least some of the criteria or groups must be satisfied</i>
-------------------	---

---

**Description**

Function to construct a nested group where at least some of the criteria or groups must be satisfied

**Usage**

```
nestedWithAtLeast(x, ...)
```

**Arguments**

x	an integer specifying the number of criteria or groups that must be satisfied
...	a set of criteria or groups

---

nestedWithAtMost	<i>Function to construct a nested group where at most some of the criteria or groups must be satisfied</i>
------------------	--

---

**Description**

Function to construct a nested group where at most some of the criteria or groups must be satisfied

**Usage**

```
nestedWithAtMost(x, ...)
```

**Arguments**

x	an integer specifying the number of criteria or groups that must be satisfied
...	a set of criteria or groups

---

observation	<i>Query the observation domain</i>
-------------	-------------------------------------

---

**Description**

Query the observation domain

**Usage**

```
observation(conceptSet, ...)
```

**Arguments**

conceptSet	A condition concept set
...	optional attributes

**Value**

A Capr Query

---

observationExit	<i>Function to create an exit based on continuous observation</i>
-----------------	---

---

**Description**

Function to create an exit based on continuous observation

**Usage**

```
observationExit()
```

---

ObservationExit-class    *An S4 class for a cohort exit based on end of continuous observation.*

---

### Description

The event persists until the end of continuous observation of the persons

### Slots

index    specification of event date to offset

offsetDays    an integer specifying the number of days to offset from the event date

---

ObservationWindow-class

*An S4 class for an ObservationWindow*

---

### Description

this determines the minimal observation time before and after index for all persons in the cohort

### Slots

priorDays    minimum number of days prior to the cohort index

postDays    minimum number of days post cohort index

---

Occurrence-class

*An S4 class for an occurrence.*

---

### Description

This determines how many events need to occur to count the criteria in the cohort definition (relative to the index event)

### Slots

type    a character string determine the logic for counting occurrences. Can be all, any, exactly, atLeast, or atMost

count    an integer specifying the number of occurrences for a criteria



---

opAttributeDate-class    *An S4 class for a op attribute that is a date*

---

### Description

An S4 class for a op attribute that is a date

### Slots

name   the name of the attribute  
 op   the operator one of: gt,lt,gte,lte,eq,bt,!bt  
 value   a value serving as the single limit or lower limit in a bt.  
 extent   a value serving as the upper limit in a bt, otherwise this is empty

---

opAttributeInteger-class  
                                   *An S4 class for a op attribute that is an integer*

---

### Description

An S4 class for a op attribute that is an integer

### Slots

name   the name of the attribute  
 op   the operator one of: gt,lt,gte,lte,eq,bt,!bt  
 value   a value serving as the single limit or lower limit in a bt.  
 extent   a value serving as the upper limit in a bt, otherwise this is empty

---

opAttributeNumeric-class  
                                   *An S4 class for a op attribute that is a numeric*

---

### Description

An S4 class for a op attribute that is a numeric

### Slots

name   the name of the attribute  
 op   the operator one of: gt,lt,gte,lte,eq,bt,!bt  
 value   a value serving as the single limit or lower limit in a bt  
 extent   a value serving as the upper limit in a bt, otherwise this is empty

procedure	<i>Query the procedure domain</i>
<b>Description</b>	
Query the procedure domain	
<b>Usage</b>	
procedure(conceptSet, ...)	
<b>Arguments</b>	
conceptSet	A procedure concept set
...	optional attributes
<b>Value</b>	
A Capr Query	
Query-class	<i>An S4 class for a Circe Query</i>

<b>Description</b>	
A query is a medical concept that can be extracted from a database through a 'where' clause in a SQL statement. This includes concepts.	
<b>Slots</b>	
domain	The domain to search (e.g. "Condition", "Drug", "Measurement", etc)
conceptSet	The Concept set describing the observation to serach for
attributes	a list of attributes that modify the query (e.g. 'male()', 'female()', 'age(gte(65))')
rangeHigh	<i>Function to create rangeHigh attribute</i>

<b>Description</b>	
This function is used only for measurement query. range_high is a column in the measurement table of the cdm. This attribute allows a subquery to find measurements that satisfy certain values determined by the op logic.	
<b>Usage</b>	
rangeHigh(op)	
<b>Arguments</b>	
op	an opAttribute object that is either numeric or integer that defines the logical operation used to determine eligible range high

rangeLow

*Function to create rangeLow attribute***Description**

This function is used only for measurement query. range\_low is a column in the measurement table of the cdm. This attribute allows a subquery to find measurements that satisfy certain values determined by the op logic.

**Usage**

```
rangeLow(op)
```

**Arguments**

op	an opAttribute object that is either numeric or integer that defines the logical operation used to determine eligible range low
----	---

readConceptSet

*Read a concept set json or csv into R***Description**

Concept sets can be serialized to json or csv file formats. 'readConceptSet' reads the files into R as Capr concepts sets.

**Usage**

```
readConceptSet(path, name, id = NULL)
```

**Arguments**

path	Name of concept set file to read in csv or json format. (e.g. "concepts.json")
name	the name of the concept set
id	the id for the concept set (keep?)

**Examples**

```
## Not run:
anemia <- readConceptSet('anemia.json')
anemia <- readConceptSet('anemia.csv')

## End(Not run)
```

---

startDate	<i>Function that creates a start date attribute</i>
-----------	---

---

**Description**

Function that creates a start date attribute

**Usage**

```
startDate(op)
```

**Arguments**

op	an opAttribute object must be a date that defines the logical operation used to determine eligible start dates
----	--

---

toCirce	<i>Function to coerce cohort to circe</i>
---------	---

---

**Description**

Function to coerce cohort to circe

**Usage**

```
toCirce(cd)
```

**Arguments**

cd	the Capr cohort class
----	-----------------------

---

unit	<i>Add unit attribute to a query</i>
------	--------------------------------------

---

**Description**

Add unit attribute to a query

**Usage**

```
unit(x)
```

**Arguments**

x	A single character identifier for a unit or a concept set that identifies units
---	---

**Value**

An attribute that can be used in a query function

---

valueAsNumber	<i>Function to create valueAsNumber attribute</i>
---------------	---

---

**Description**

This function is used only for measurement query. valueAsNumber is a column in the measurement table of the cdm. This attribute allows a subquery to find measurements that satisfy certain values determined by the op logic.

**Usage**

```
valueAsNumber(op)
```

**Arguments**

op	an opAttribute object that is either numeric or integer that defines the logical operation used to determine eligible patient age
----	---

---

visit	<i>Query the visit occurrence domain</i>
-------	--

---

**Description**

Query the visit occurrence domain

**Usage**

```
visit(conceptSet, ...)
```

**Arguments**

conceptSet	A condition concept set
...	optional attributes

**Value**

A Capr Query

---

withAll	<i>Function to construct a group where all criteria and groups must be satisfied</i>
---------	--

---

**Description**

Function to construct a group where all criteria and groups must be satisfied

**Usage**

```
withAll(...)
```

**Arguments**

...	a set of criteria or groups
-----	-----------------------------

---

withAny	<i>Function to construct a group where any criteria and groups may be satisfied</i>
---------	---

---

**Description**

Function to construct a group where any criteria and groups may be satisfied

**Usage**

```
withAny(...)
```

**Arguments**

...	a set of criteria or groups
-----	-----------------------------

---

withAtLeast	<i>Function to construct a group where at least some of the criteria or groups must be satisfied</i>
-------------	--

---

**Description**

Function to construct a group where at least some of the criteria or groups must be satisfied

**Usage**

```
withAtLeast(x, ...)
```

**Arguments**

x	an integer specifying the number of criteria or groups that must be satisfied
...	a set of criteria or groups

---

withAtMost	<i>Function to construct a group where at most some of the criteria or groups must be satisfied</i>
------------	---

---

### Description

Function to construct a group where at most some of the criteria or groups must be satisfied

### Usage

```
withAtMost(x, ...)
```

### Arguments

x	an integer specifying the number of criteria or groups that must be satisfied
...	a set of criteria or groups

---

writeCohort	<i>Write Cohort json file</i>
-------------	-------------------------------

---

### Description

Write Cohort json file

### Usage

```
writeCohort(x, path)
```

### Arguments

x	A Capr cohort
path	The name of the file to create

### Examples

```
## Not run:
cs1 <- cs(descendants(exclude(436665),440383,442306,4175329))
cs1 <- getConceptSetDetails(cs1)
x <- cohort(condition(cs1))
writeCohort(x, "cohortDefinition.json")

## End(Not run)
```

---

writeConceptSet	<i>Save a concept set as a json file</i>
-----------------	--

---

**Description**

The resulting concept Set JSON file can be imported into Atlas.

**Usage**

```
writeConceptSet(x, path, format = "auto", ...)
```

**Arguments**

x	A Capr concept set created by 'cs()'
path	Name of file to write to. (e.g. "concepts.json")
format	the file extension to write
...	additional arguments

**Examples**

```
## Not run:
anemia <- cs(descendants(439777,4013073,4013074))
writeConceptSet(anemia, 'anemia.json')
writeConceptSet(anemia, 'anemia.csv')

## End(Not run)
```



# Index

[==](#), [4](#)  
[==](#), [ConceptSet](#), [ConceptSet-method \(==\)](#), [4](#)  
  
[age](#), [4](#)  
[as.data.frame](#), [ConceptSet-method](#), [5](#)  
[as.json](#), [5](#)  
[as.json](#), [Cohort-method \(as.json\)](#), [5](#)  
[atLeast](#), [6](#)  
[atMost](#), [6](#)  
[attrition](#), [7](#)  
  
[bt](#), [7](#)  
[bt](#), [Date-method \(bt\)](#), [7](#)  
[bt](#), [integer-method \(bt\)](#), [7](#)  
[bt](#), [numeric-method \(bt\)](#), [7](#)  
  
[CensoringCriteria-class](#), [8](#)  
[censoringEvents](#), [8](#)  
[cohort](#), [8](#)  
[Concept-class](#), [9](#)  
[conceptAttribute-class](#), [9](#)  
[ConceptSet-class](#), [9](#)  
[ConceptSetItem-class](#), [10](#)  
[condition](#), [10](#)  
[conditionEra](#), [11](#)  
[continuousObservation](#), [11](#)  
[Criteria-class](#), [11](#)  
[cs](#), [12](#)  
  
[daysOfSupply](#), [13](#)  
[death](#), [13](#)  
[descendants \(cs\)](#), [12](#)  
[drug](#), [14](#)  
[drugEra](#), [14](#)  
[drugExit](#), [15](#)  
[DrugExposureExit-class](#), [15](#)  
[drugQuantity](#), [16](#)  
[drugRefills](#), [16](#)  
[duringInterval](#), [17](#)  
  
[endDate](#), [17](#)  
[Endpoint-class](#), [18](#)  
[entry](#), [18](#)  
[eq](#), [19](#)  
[eq](#), [Date-method \(eq\)](#), [19](#)  
  
[eq](#), [integer-method \(eq\)](#), [19](#)  
[eq](#), [numeric-method \(eq\)](#), [19](#)  
[era](#), [19](#)  
[EventAperture-class](#), [20](#)  
[eventEnds](#), [20](#)  
[eventStarts](#), [21](#)  
[EventWindow-class](#), [21](#)  
[exactly](#), [22](#)  
[exclude \(cs\)](#), [12](#)  
[exit](#), [22](#)  
  
[female \(male\)](#), [28](#)  
[firstOccurrence](#), [22](#)  
[FixedDurationExit-class](#), [23](#)  
[fixedExit](#), [23](#)  
  
[generateCaprTemplate](#), [23](#)  
[getConceptSetCall](#), [24](#)  
[getConceptSetDetails](#), [24](#)  
[Group-class](#), [25](#)  
[gt](#), [25](#)  
[gt](#), [Date-method \(gt\)](#), [25](#)  
[gt](#), [integer-method \(gt\)](#), [25](#)  
[gt](#), [numeric-method \(gt\)](#), [25](#)  
[gte](#), [26](#)  
[gte](#), [Date-method \(gte\)](#), [26](#)  
[gte](#), [integer-method \(gte\)](#), [26](#)  
[gte](#), [numeric-method \(gte\)](#), [26](#)  
  
[logicAttribute-class](#), [26](#)  
[lt](#), [27](#)  
[lt](#), [Date-method \(lt\)](#), [27](#)  
[lt](#), [integer-method \(lt\)](#), [27](#)  
[lt](#), [numeric-method \(lt\)](#), [27](#)  
[lte](#), [27](#)  
[lte](#), [Date-method \(lte\)](#), [27](#)  
[lte](#), [integer-method \(lte\)](#), [27](#)  
[lte](#), [numeric-method \(lte\)](#), [27](#)  
  
[male](#), [28](#)  
[mapped \(cs\)](#), [12](#)  
[measurement](#), [28](#)  
  
[nbt](#), [29](#)  
[nbt](#), [Date-method \(nbt\)](#), [29](#)

nbt, integer-method (nbt), [29](#)  
nbt, numeric-method (nbt), [29](#)  
nestedAttribute-class, [29](#)  
nestedWithAll, [30](#)  
nestedWithAny, [30](#)  
nestedWithAtLeast, [30](#)  
nestedWithAtMost, [31](#)

observation, [31](#)  
observationExit, [31](#)  
ObservationExit-class, [32](#)  
ObservationWindow-class, [32](#)  
Occurrence-class, [32](#)  
opAttributeDate-class, [33](#)  
opAttributeInteger-class, [33](#)  
opAttributeNumeric-class, [33](#)

procedure, [34](#)

Query-class, [34](#)

rangeHigh, [34](#)  
rangeLow, [35](#)  
readConceptSet, [35](#)

startDate, [36](#)

toCirce, [36](#)

unit, [36](#)

valueAsNumber, [37](#)  
visit, [37](#)

withAll, [38](#)  
withAny, [38](#)  
withAtLeast, [38](#)  
withAtMost, [39](#)  
writeCohort, [39](#)  
writeConceptSet, [40](#)