Using extended cohort attributes in Capr

Contents

0.1 Introduction

There are several attributes that are available in circe-be and found in ATLAS. In future releases, `Capr` will create a dedicated function signature to each attribute (subject to user feedback). In order to extend the usage of attributes that are not personalized, a few low level functions were made available to access a greater set of attributes. The functions are: `createOpAttribute` and `createConceptAttribute`. OpAttributes are for attributes that contain an operator and concept attribute include concept ids that are not part of the cohort concept set list. To view all possible attribute in Capr available for each domain, a helper function `listAttributeOptions` can be used. Below are some of the available attribute per attribute class.

```
library(Capr)
AttributeOptions
#> $Op
#>  [1] "Age"                 "OccurrenceStartDate" "OccurrenceEnd"       "AgeAtEnd"            "AgeAtS
#>  [6] "PeriodLength"        "ValueAsNumber"       "RangeLow"            "RangeHigh"           "RangeL
#> [11] "RangeHighRatio"      "EraStartDate"        "EraEndDate"          "OccurrenceCount"     "EraLen
#> [16] "Refills"             "Quantity"            "DaysSupply"          "EffectiveDrugDose"   "VisitL
#>
#> $Concept
#>  [1] "ConditionType"     "Gender"            "VisitType"         "DrugType"          "RouteConcepts"
#>  [7] "ProviderSepcialty" "PlaceOfService"    "ProcedureType"     "Modifier"          "ObservationTyp
#> [13] "Qualifier"         "Unit"              "MeasurementType"   "Operator"          "DeathType"
#>
#> $Logical
#>  [1] "First"                  "DrugTypeExclude"        "ConditionTypeExclude"   "VisitTypeExclude"
#>  [6] "ObservationTypeExclude" "MeasurementTypeExclude" "Abnormal"               "DeathTypeExclude"
#>
#> $SourceConcept
#> [1] "VisitSourceConcept"       "DrugSourceConcept"        "ConditionSourceConcept"   "ProcedureSourc
#> [5] "ObservationSourceConcept" "MeasurementSourceConcept" "DeathSourceConcept"       "DeviceSourceCo
#>
#> $TextFilter
#> [1] "ValueAsString"  "StopReason"     "UniqueDeviceId"
```

Not all of these attributes are Op or Concept, there are others like source and logical which already have been personalized and do not need use of low level functions. An example of a logical is `createFirstAttribute` and an example of a source concept is `createDeviceSourceConceptAttribute`. Correlated Criteria is

another attribute that is already user facing. Currently text filler attributes are not supported but will be extended in a future release.

0.2 Op Attribute

The Op Attribute uses an operator to bound either a single value or a value and extent. Operators include less than, less than or equal to, greater than, greater than or equal to, equal to, between and not between. Capr allows four type of entries for operators: a symbol, text, short, and indicies. Capr will automatically map the operator to the short hand text which is used in circe. Two operators (between and not between) require two inputs a value and an extent. No other operator requires an extent, which should be NULL by default. The value and extent can either be integers or character date strings. If you wish to use a data Op attribute than the character date string must be of form yyyy-mm-dd, otherwise it will be invalid in the circe compiler. In future releases, a separate date attribute may be created.

symb	text	short	idx
<	less than	lt	1
<=	less than or equal to	lte	2
>	greater than	gt	3
>=	greater than or equal to	gte	4
==	equal to	eq	5
–	between	bt	6
!-	not between	!bt	7

An example of using a Op Attribute can be seen in the chunk below:

```
opExample <- createOpAttribute(Name = "RangeHigh", Op = ">=", Value = 5L)
str(opExample)
#> Formal class 'Component' [package "Capr"] with 4 slots
#>   ..@ MetaData            :Formal class 'MetaData' [package "Capr"] with 3 slots
#>   .. .. ..@ ComponentType: chr "Attribute"
#>   .. .. ..@ Name         : chr "OpAttribute"
#>   .. .. ..@ Description  : chr NA
#>   ..@ CriteriaExpression  :List of 1
#>   .. ..$ :Formal class 'OpAttribute' [package "Capr"] with 3 slots
#>   .. .. .. ..@ Name    : chr "RangeHigh"
#>   .. .. .. ..@ Op      : chr "gte"
#>   .. .. .. ..@ Contents:List of 2
#>   .. .. .. .. ..$ Value : int 5
#>   .. .. .. .. ..$ Extent: int NA
#>   ..@ Limit               : list()
#>   ..@ ConceptSetExpression: list()
```

For a date an example is in the chunk below:

```
opExample2 <- createOpAttribute(Name = "OccurrenceStartDate", Op = "greater than", Value = "2018-12-31")
str(opExample2)
#> Formal class 'Component' [package "Capr"] with 4 slots
#>   ..@ MetaData            :Formal class 'MetaData' [package "Capr"] with 3 slots
#>   .. .. ..@ ComponentType: chr "Attribute"
#>   .. .. ..@ Name         : chr "OpAttribute"
#>   .. .. ..@ Description  : chr NA
```

```
#>   ..@ CriteriaExpression  :List of 1
#>   .. ..$ :Formal class 'OpAttribute' [package "Capr"] with 3 slots
#>   .. .. .. ..@ Name    : chr "OccurrenceStartDate"
#>   .. .. .. ..@ Op      : chr "gt"
#>   .. .. .. ..@ Contents:List of 2
#>   .. .. .. .. ..$ Value : chr "2018-12-31"
#>   .. .. .. .. ..$ Extent: chr NA
#>   ..@ Limit               : list()
#>   ..@ ConceptSetExpression: list()
```

0.3 Concept Attribute

A concept attribute looks up concept ids without including them as concept sets in the cohort definitions. Examples of a concept attribute is Gender. Male (ID: 8507) and Female (ID: 8532) have standard concept ids that are often used as attributes. We can apply the lookup tools to assist us in querying concept ids from the OMOP vocabulary. An example of creating a concept attribute is shown below:

```
createConceptAttribute(conceptIds = 8507, name = "Gender")
```