Package ‘AnnotationFilter’

May 29, 2024

Title Facilities for Filtering Bioconductor Annotation Resources
Version 1.28.0
URL https://github.com/Bioconductor/AnnotationFilter
BugReports https://github.com/Bioconductor/AnnotationFilter/issues
Description This package provides class and other infrastructure to implement filters for manipulating Bioconductor annotation resources. The filters will be used by ensembl db, Organism.dplyr, and other packages.
Depends R (>= 3.4.0)
Imports utils, methods, GenomicRanges, lazyeval
Suggests BiocStyle, knitr, testthat, RSQLite, org.Hs.eg.db, rmarkdown
VignetteBuilder knitr
License Artistic-2.0
biocViews Annotation, Infrastructure, Software
Encoding UTF-8
LazyData true
RoxygenNote 6.0.1
Collate 'AllGenerics.R' 'AnnotationFilter.R' 'AnnotationFilterList.R'
   'translate-utils.R'
git_url https://git.bioconductor.org/packages/AnnotationFilter
git_branch RELEASE_3_19
git_last_commit 61709ad
git_last_commit_date 2024-04-30
Repository Bioconductor 3.19
Date/Publication 2024-05-29
Author Martin Morgan [aut],
   Johannes Rainer [aut],
   Joachim Bargsten [ctb],
   Daniel Van Twisk [ctb],
   Bioconductor Package Maintainer [cre]
Maintainer Bioconductor Package Maintainer <maintainer@bioconductor.org>
Contents

AnnotationFilter ................................................................. 2
AnnotationFilterList .......................................................... 5
GenenameFilter ................................................................. 7

Description

The filters extending the base AnnotationFilter class represent a simple filtering concept for annotation resources. Each filter object is thought to filter on a single (database) table column using the provided values and the defined condition.

Filter instances created using the constructor functions (e.g. GeneIdFilter).
supportedFilters() lists all defined filters. It returns a two column data.frame with the filter class name and its default field. Packages using AnnotationFilter should implement the supportedFilters for their annotation resource object (e.g. for object = "EnsDb" in the ensembldb package) to list all supported filters for the specific resource.

condition() get the condition value for the filter object.
value() get the value for the filter object.
field() get the field for the filter object.
not() get the not for the filter object.
feature() get the feature for the GRangesFilter object.

Converts an AnnotationFilter object to a character(1) giving an equation that can be used as input to a dplyr filter.

AnnotationFilter translates a filter expression such as ~ gene_id == "BCL2" into a filter object extending the AnnotationFilter class (in the example a GeneIdFilter object) or an AnnotationFilterList if the expression contains multiple conditions (see examples below). Filter expressions have to be written in the form ~ <field> <condition> <value>, with <field> being the default field of the filter class (use the supportedFilter function to list all fields and filter classes), <condition> the logical expression and <value> the value for the filter.

Usage

CdsStartFilter(value, condition = "==", not = FALSE)
CdsEndFilter(value, condition = "==", not = FALSE)
ExonIdFilter(value, condition = "==", not = FALSE)
ExonNameFilter(value, condition = "==", not = FALSE)
ExonRankFilter(value, condition = "==", not = FALSE)
ExonStartFilter(value, condition = "==", not = FALSE)
ExonEndFilter(value, condition = "==", not = FALSE)
GeneIdFilter(value, condition = "==", not = FALSE)
GeneNameFilter(value, condition = "==", not = FALSE)
GeneBiotypeFilter(value, condition = "==", not = FALSE)
GeneStartFilter(value, condition = "==", not = FALSE)
GeneEndFilter(value, condition = "==", not = FALSE)
EntrezFilter(value, condition = "==", not = FALSE)
SymbolFilter(value, condition = "==", not = FALSE)
TxIdFilter(value, condition = "==", not = FALSE)
TxNameFilter(value, condition = "==", not = FALSE)
TxBiotypeFilter(value, condition = "==", not = FALSE)
TxStartFilter(value, condition = "==", not = FALSE)
TxEndFilter(value, condition = "==", not = FALSE)
ProteinIdFilter(value, condition = "==", not = FALSE)
UniprotFilter(value, condition = "==", not = FALSE)
SeqNameFilter(value, condition = "==", not = FALSE)
SeqStrandFilter(value, condition = "==", not = FALSE)

## S4 method for signature 'AnnotationFilter'
condition(object)

## S4 method for signature 'AnnotationFilter'
value(object)

## S4 method for signature 'AnnotationFilter'
field(object)

## S4 method for signature 'AnnotationFilter'
not(object)

GRangesFilter(value, feature = "gene", type = c("any", "start", "end", "within", "equal"))

feature(object)

## S4 method for signature 'AnnotationFilter,missing'
convertFilter(object)

## S4 method for signature 'missing'
supportedFilters(object)

AnnotationFilter(expr)

**Arguments**

- **object**
  - An `AnnotationFilter` object.

- **value**
  - character(), integer(), or GRanges() value for the filter

- **feature**
  - character(1) defining on what feature the GRangesFilter should be applied. Choices could be "gene", "tx" or "exon".
type  character(1) indicating how overlaps are to be filtered. See `findOverlaps` in the `IRanges` package for a description of this argument.

expr  A filter expression, written as a formula, to be converted to an `AnnotationFilter` or `AnnotationFilterList` class. See below for examples.

condition  character(1) defining the condition to be used in the filter. For `IntegerFilter` or `DoubleFilter`, one of `"==", "!=", ">", "<", ">=", "<="`. For `CharacterFilter`, one of `"==", "!=", "startsWith", "endsWith" or "contains"`. Default condition is `"=="`.

not  logical(1) whether the `AnnotationFilter` is negated. TRUE indicates is negated (!). FALSE indicates not negated. Default not is FALSE.

Details

By default filters are only available for tables containing the field on which the filter acts (i.e. that contain a column with the name matching the value of the field slot of the object). See the vignette for a description to use filters for databases in which the database table column name differs from the default field of the filter.

Filter expressions for the `AnnotationFilter` class have to be written as formulas, i.e. starting with a `~`.

Value

The constructor function return an object extending `AnnotationFilter`. For the return value of the other methods see the methods’ descriptions.

character(1) that can be used as input to a `dplyr` filter.

`AnnotationFilter` returns an `AnnotationFilter` or an `AnnotationFilterList`.

Note

Translation of nested filter expressions using the `AnnotationFilter` function is not yet supported.

See Also

`AnnotationFilterList` for combining `AnnotationFilter` objects.

Examples

```r
## filter by GRanges
GRangesFilter(\code{GenomicRanges::GRanges("chr10:87869000-87876000")})
## Create a SymbolFilter to filter on a gene's symbol.
sf <- SymbolFilter("BCL2")
sf

## Create a GeneStartFilter to filter based on the genes' chromosomal start
## coordinates
gsf <- GeneStartFilter(10000, condition = ">")
gsf

filter <- SymbolFilter("ADA", ">=")
```
result <- convertFilter(filter)
result
supportedFilters()

## Convert a filter expression based on a gene ID to a GeneIdFilter
gnf <- AnnotationFilter(~ gene_id == "BCL2")
gnf

## Same conversion but for two gene IDs.
gnf <- AnnotationFilter(~ gene_id %in% c("BCL2", "BCL2L11"))
gnf

## Converting an expression that combines multiple filters. As a result we
## get an AnnotationFilterList containing the corresponding filters.
## Be aware that nesting of expressions/filters does not work.
flt <- AnnotationFilter(~ gene_id %in% c("BCL2", "BCL2L11") &
                        tx_biotype == "nonsense_mediated_decay" |
                        seq_name == "Y")
flt

### Description

The AnnotationFilterList allows to combine filter objects extending the AnnotationFilter class to construct more complex queries. Consecutive filter objects in the AnnotationFilterList can be combined by a logical and (&) or or (|). The AnnotationFilterList extends list, individual elements can thus be accessed with [[].

value() get a list with the AnnotationFilter objects. Use [[ to access individual filters.

logicOp() gets the logical operators separating successive AnnotationFilter.

not() gets the logical operators separating successive AnnotationFilter.

Converts an AnnotationFilterList object to a character(1) giving an equation that can be used as input to a dplyr filter.

### Usage

AnnotationFilterList(..., logicOp = character(), logOp = character(),
                     not = FALSE, .groupingFlag = FALSE)

## S4 method for signature 'AnnotationFilterList'
value(object)

## S4 method for signature 'AnnotationFilterList'
logicOp(object)
## S4 method for signature 'AnnotationFilterList'
not(object)

## S4 method for signature 'AnnotationFilterList'
distributeNegation(object,
  .prior_negation = FALSE)

## S4 method for signature 'AnnotationFilterList,missing'
convertFilter(object)

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

### Arguments

...  individual `AnnotationFilter` objects or a mixture of `AnnotationFilter` and `AnnotationFilterList` objects.

logicOp  character of length equal to the number of submitted `AnnotationFilter` objects - 1. Each value representing the logical operation to combine consecutive filters, i.e. the first element being the logical operation to combine the first and second `AnnotationFilter`, the second element being the logical operation to combine the second and third `AnnotationFilter` and so on. Allowed values are "&" and "|". The function assumes a logical *and* between all elements by default.


not  logical of length one. Indicates whether the grouping of `AnnotationFilters` are to be negated.

.groupingFlag  Flag designated for internal use only.

object  An object of class `AnnotationFilterList`.

.prior_negation  logical(1) unused argument.

### Value

`AnnotationFilterList` returns an `AnnotationFilterList`.

`value()` returns a list with `AnnotationFilter` objects.

`logicOp()` returns a character() vector of “&” or “|” symbols.

`not()` returns a character() vector of “&” or “|” symbols.

`AnnotationFilterList` object with DeMorgan’s law applied to it such that it is equal to the original `AnnotationFilterList` object but all ‘!’s are distributed out of the `AnnotationFilterList` object and to the nested `AnnotationFilter` objects.

`character(1)` that can be used as input to a dplyr filter.

### Note

The `AnnotationFilterList` does not support containing empty elements, hence all elements of length == 0 are removed in the constructor function.
GenenameFilter

Description

The GenenameFilter class and functions are deprecated. Please use the GeneNameFilter() instead.

Usage

GenenameFilter(value, condition = "==", not = FALSE)
GenenameFilter

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>character() value for the filter</td>
</tr>
<tr>
<td>condition</td>
<td>character(1) defining the condition to be used in the filter. One of &quot;==&quot;, &quot;!=&quot;, &quot;startsWith&quot;, &quot;endsWith&quot; or &quot;contains&quot;. Default condition is &quot;==&quot;.</td>
</tr>
<tr>
<td>not</td>
<td>logical(1) whether the AnnotationFilter is negated. TRUE indicates is negated (!). FALSE indicates not negated. Default not is FALSE.</td>
</tr>
</tbody>
</table>

Value

The constructor function return a GenenameFilter.
Index

<table>
<thead>
<tr>
<th>AnnotationFilter</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnnotationFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>AnnotationFilterList</td>
<td>2, 4, 5</td>
</tr>
<tr>
<td>AnnotationFilterList-class</td>
<td>5</td>
</tr>
<tr>
<td>CdsEndFilter</td>
<td>2</td>
</tr>
<tr>
<td>CdsEndFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>CdsStartFilter</td>
<td>2</td>
</tr>
<tr>
<td>CdsStartFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>CharacterFilter</td>
<td>2</td>
</tr>
<tr>
<td>condition</td>
<td>2</td>
</tr>
<tr>
<td>condition,AnnotationFilter-method</td>
<td>2</td>
</tr>
<tr>
<td>convertFilter</td>
<td>2</td>
</tr>
<tr>
<td>convertFilter,AnnotationFilterList</td>
<td>5</td>
</tr>
<tr>
<td>convertFilter,AnnotationFilter,missing-method</td>
<td>2</td>
</tr>
<tr>
<td>convertFilter,AnnotationFilterList,missing-method</td>
<td>5</td>
</tr>
<tr>
<td>distributeNegation</td>
<td>5</td>
</tr>
<tr>
<td>distributeNegation,AnnotationFilterList-method</td>
<td>5</td>
</tr>
<tr>
<td>DoubleFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>EntrezFilter</td>
<td>2</td>
</tr>
<tr>
<td>EntrezFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>ExonEndFilter</td>
<td>2</td>
</tr>
<tr>
<td>ExonEndFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>ExonIdFilter</td>
<td>2</td>
</tr>
<tr>
<td>ExonIdFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>ExonNameFilter</td>
<td>2</td>
</tr>
<tr>
<td>ExonNameFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>GRangesFilter</td>
<td>2</td>
</tr>
<tr>
<td>GRangesFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>ExonRankFilter</td>
<td>2</td>
</tr>
<tr>
<td>ExonRankFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>ExonStartFilter</td>
<td>2</td>
</tr>
<tr>
<td>ExonStartFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>feature</td>
<td>2</td>
</tr>
<tr>
<td>field</td>
<td>2</td>
</tr>
<tr>
<td>field,AnnotationFilter-method</td>
<td>2</td>
</tr>
<tr>
<td>GeneBiotypeFilter</td>
<td>2</td>
</tr>
<tr>
<td>GeneBiotypeFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>GeneEndFilter</td>
<td>2</td>
</tr>
<tr>
<td>GeneEndFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>GeneIdFilter</td>
<td>2</td>
</tr>
<tr>
<td>GeneIdFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>GenenameFilter</td>
<td>7</td>
</tr>
<tr>
<td>GenenameFilter-class</td>
<td>7</td>
</tr>
<tr>
<td>GeneNameFilter</td>
<td>7</td>
</tr>
<tr>
<td>GeneNameFilter-class</td>
<td>7</td>
</tr>
<tr>
<td>IntegerFilter-class</td>
<td>2</td>
</tr>
<tr>
<td>logicOp</td>
<td>5</td>
</tr>
<tr>
<td>logicOp,AnnotationFilterList-method</td>
<td>5</td>
</tr>
</tbody>
</table>
not (AnnotationFilterList), 5
not, AnnotationFilter-method (AnnotationFilter), 2
not, AnnotationFilterList-method (AnnotationFilterList), 5

ProteinIdFilter (AnnotationFilter), 2
ProteinIdFilter-class (AnnotationFilter), 2

SeqNameFilter (AnnotationFilter), 2
SeqNameFilter-class (AnnotationFilter), 2

SeqStrandFilter (AnnotationFilter), 2
SeqStrandFilter-class (AnnotationFilter), 2
show, AnnotationFilter-method (AnnotationFilter), 2
show, AnnotationFilterList-method (AnnotationFilterList), 5
show, CharacterFilter-method (AnnotationFilter), 2
show, DoubleFilter-method (AnnotationFilter), 2
show, GRangesFilter-method (AnnotationFilter), 2
show, IntegerFilter-method (AnnotationFilter), 2
supportedFilters, 7
supportedFilters, missing-method (AnnotationFilter), 2
SymbolFilter (AnnotationFilter), 2
SymbolFilter-class (AnnotationFilter), 2

TxBiotypeFilter (AnnotationFilter), 2
TxBiotypeFilter-class (AnnotationFilter), 2
TxEndFilter (AnnotationFilter), 2
TxEndFilter-class (AnnotationFilter), 2
TxIdFilter (AnnotationFilter), 2
TxIdFilter-class (AnnotationFilter), 2
TxNameFilter (AnnotationFilter), 2
TxNameFilter-class (AnnotationFilter), 2
TxStartFilter (AnnotationFilter), 2
TxStartFilter-class (AnnotationFilter), 2

UniprotFilter (AnnotationFilter), 2
UniprotFilter-class (AnnotationFilter), 2

value (AnnotationFilter), 2
value, AnnotationFilter-method (AnnotationFilter), 2
value, AnnotationFilterList-method (AnnotationFilterList), 5