Package 'Organism.dplyr'

October 18, 2017

Title dplyr-based Access to Bioconductor Annotation Resources

Version 1.2.2

Description This package provides an alternative interface to Bioconductor 'annotation' resources, in particular the gene identifier mapping functionality of the 'org' packages (e.g., org.Hs.eg.db) and the genome coordinate functionality of the 'TxDb' packages (e.g., TxDb.Hsapiens.UCSC.hg38.knownGene).

Depends R (>= 3.4), dplyr (>= 0.7.0)

Imports RSQLite, S4Vectors, GenomeInfoDb, IRanges, GenomicRanges, GenomicFeatures, AnnotationDbi, methods, tools, utils, BiocFileCache, dbplyr, DBI

Suggests org.Hs.eg.db, TxDb.Hsapiens.UCSC.hg38.knownGene, org.Mm.eg.db, TxDb.Mmusculus.UCSC.mm10.ensGene, testthat, knitr, rmarkdown, BiocStyle, ggplot2

License Artistic-2.0

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Collate src.R filter.R extractors.R select.R utils.R

VignetteBuilder knitr

biocViews Annotation, Sequencing, GenomeAnnotation

NeedsCompilation no

Author Martin Morgan [aut, cre], Yubo Cheng [ctb]

Maintainer Martin Morgan <martin.morgan@roswellpark.org>

R topics documented:

`BasicFilter-class` *Filtering src_organism objects*

Description

These functions create filters to be used by the "select" interface to `src_organism` objects.

Usage

```
AccnumFilter(value, condition = "==")
AliasFilter(value, condition = "==")
CdsChromFilter(value, condition = "==")
CdsIdFilter(value, condition = "==")
CdsNameFilter(value, condition = "==")
CdsStrandFilter(value, condition = "==")
EnsemblFilter(value, condition = "==")
EnsemblprotFilter(value, condition = "==")
EnsembltransFilter(value, condition = "==")
EntrezFilter(value, condition = "==")
EnzymeFilter(value, condition = "==")
EvidenceFilter(value, condition = "==")
EvidenceallFilter(value, condition = "==")
ExonChromFilter(value, condition = "==")
ExonIdFilter(value, condition = "==")
ExonNameFilter(value, condition = "==")
ExonRankFilter(value, condition = "==")
ExonStrandFilter(value, condition = "==")
FlybaseFilter(value, condition = "==")
FlybaseCgFilter(value, condition = "==")
FlybaseProtFilter(value, condition = "==")
GeneChromFilter(value, condition = "==")
GeneStrandFilter(value, condition = "==")
GenenameFilter(value, condition = "==")
GoFilter(value, condition = "==")
GoallFilter(value, condition = "==")
IpiFilter(value, condition = "==")
MapFilter(value, condition = "==")
MgiFilter(value, condition = "==")
OmimFilter(value, condition = "==")
OntologyFilter(value, condition = "==")
OntologyallFilter(value, condition = "==")
PfamFilter(value, condition = "==")
PmidFilter(value, condition = "==")
PrositeFilter(value, condition = "==")
RefseqFilter(value, condition = "==")
SymbolFilter(value, condition = "==")
TxChromFilter(value, condition = "==")
TxIdFilter(value, condition = "==")
TxNameFilter(value, condition = "==")
TxStrandFilter(value, condition = "==")
TxTypeFilter(value, condition = "==")
```

```
UnigeneFilter(value, condition = "==")
UniprotFilter(value, condition = "==")
WormbaseFilter(value, condition = "==")
ZfinFilter(value, condition = "==")
CdsStartFilter(value, condition = "==")
CdsEndFilter(value, condition = "==")
ExonStartFilter(value, condition = "==")
ExonEndFilter(value, condition = "==")
GeneStartFilter(value, condition = "==")
GeneEndFilter(value, condition = "==")
TxStartFilter(value, condition = "==")
TxEndFilter(value, condition = "==")

GRangesFilter(value)

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

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

supportedFilters()
```

Arguments

value	Value of the filter. For GRangesFilter value should be a GRanges object.
object	A BasicFilter or GRangesFilter object
condition	The condition to be used in filter for genomic extractors, one of "==", "!=", "startsWith", "endsWith", ">", "<", ">=", "<=". For character values "==", "!=", "startsWith" and "endsWith" are allowed, for numeric values (CdsStartFilter, CdsEndFilter, ExonStartFilter, ExonEndFilter, GeneStartFilter, GeneEndFilter, TxStartFilter and TxEndFilter), "==", "!=", ">", ">=", "<" and "<=". Default condition is "==".

Details

All filters except GRangesFilter() takes value(s) from corresponding fields in the data base. For example, AccnumFilter() takes values of accession number(s), which come from field accnum. See keytypes() and keys() for possible values.

GRangesFilter() takes a GRanges object as filter, and returns genomic extractors (genes, transcripts, etc.) that are partially overlapping with the region.

supportedFilters() lists all available filters for src_organism object.

Value

A Filter object showing class, value and condition of the filter

Author(s)

Yubo Cheng.

See Also

src_organism for creating a src_organism object.

transcripts_tbl for generic functions to extract genomic features from a src_organism object.

select,src_organism-method for "select" interface on src_organism objects.

Examples

```
src <- src_organism(dbpath=hg38light())
keytypes(src)
head(keys(src, "ensembl"))

## filter by ensembl
EnsemblFilter("ENSG00000171862")

## filter by gene symbol start with "BRAC"
SymbolFilter("BRCA", "startsWith")

## filter by GRanges
GRangesFilter(GenomicRanges::GRanges("chr10:87869000-87876000"))

## filter by transcript start position
TxStartFilter(87863438, ">")
```

---

hg38light *Utilities used in examples, vignettes, and tests*

Description

These functions are primarily for illustrating functionality. hg38light() and mm10light() provide access to trimmed-down versions of Organism.dplyr data based derived from the TxDb.Hsapiens.UCSC.hg38.knownGene and TxDb.Mmusculus.UCSC.mm10.ensGene data bases.

Usage

```
hg38light()

mm10light()
```

Value

character(1) file path to the trimmed-down data base

Examples

```
hg38light()
mm10light()
```

keytypes,src_organism-method
Using the "select" interface on src_organism objects

Description

select, columns and keys can be used together to extract data from a src_organism object.

Usage

```
## S4 method for signature 'src_organism'
keytypes(x)

## S4 method for signature 'src_organism'
columns(x)

## S4 method for signature 'src_organism'
keys(x, keytype, ...)

select_tbl(x, keys, columns, keytype)

## S4 method for signature 'src_organism'
select(x, keys, columns, keytype)

## S4 method for signature 'src_organism'
mapIds(x, keys, column, keytype, ..., multiVals)
```

Arguments

x	a src_organism object
keytype	specifies the kind of keys that will be returned. By default keys will return the keys for schema of the src_organism object.
...	other arguments. These include: pattern: the pattern to match. column: the column to search on. fuzzy: TRUE or FALSE value. Use fuzzy matching? (this is used with pattern)
keys	the keys to select records for from the database. All possible keys are returned by using the keys method.
columns	the columns or kinds of things that can be retrieved from the database. As with keys, all possible columns are returned by using the columns method.
column	character(1) the column to search on, can only have a single element for the value
multiVals	What should mapIds do when there are multiple values that could be returned. Options include: first: when there are multiple matches only the 1st thing that comes back will be returned. This is the default behavior. list: return a list object to the end user

filter: remove all elements that contain multiple matches and will therefore return a shorter vector than what came in whenever some of the keys match more than one value

asNA: return an NA value whenever there are multiple matches

CharacterList: returns a SimpleCharacterList object

FUN: can also supply a function to the multiVals argument for custom behaviors. The function must take a single argument and return a single value. This function will be applied to all the elements and will serve a 'rule' that for which thing to keep when there is more than one element. So for example this example function will always grab the last element in each result: `last <- function(x){x[[length(x)]]}`

Details

`keytypes()`: discover which keytypes can be passed to keytype argument of methods `select` or `keys`.

`keys()`: returns keys for the `src_organism` object. By default it returns the primary keys for the database, and returns the keys from that keytype when the keytype argument is used.

`columns()`: discover which kinds of data can be returned for the `src_organism` object.

`select()`: retrieves the data as a `tibble` based on parameters for selected keys columns and keytype arguments. If requested columns that have multiple matches for the keys, 'select()' will return a `tibble` with one row for each possible match.

`mapIds()`: gets the mapped ids (column) for a set of keys that are of a particular keytype. Usually returned as a named character vector.

Value

`keys`, `columns` and `keytypes` each returns a character vector of possible values. `select` returns a `tibble`.

Author(s)

Yubo Cheng.

See Also

`AnnotationDb-class` for more descriptsion of methods `select`, `keytypes`, `keys` and `columns`.

`src_organism` for creating a `src_organism` object.

`transcripts_tbl` for generic functions to extract genomic features from a `src_organism` object.

Examples

```
## Not run: src <- src_organism("TxDb.Hsapiens.UCSC.hg38.knownGene")
src <- src_organism(dbpath=hg38light())

## keytypes
keytypes(src)

## columns
columns(src)

## keys
```

```
keys(src, "entrez")

keytype <- "symbol"
keys <- c("ADA", "NAT2")
columns <- c("entrez", "tx_id", "tx_name","exon_id")

## select
select_tbl(src, keys, columns, keytype)
select(src, keys, columns, keytype)

## mapIds
mapIds(src, keys, column = "tx_name", keytype)
```

---

src_organism *Create a sqlite database from TxDb and corresponding Org packages*

Description

The database provides a convenient way to map between gene, transcript, and protein identifiers.

Usage

```
src_organism(txdb = NULL, dbpath = NULL)

src_ucsc(organism, genome = NULL, id = NULL, dbpath = NULL,
  verbose = TRUE)

supportedOrganisms()

## S3 method for class 'tbl_organism'
select_(.data, ...)

## S3 method for class 'src_organism'
src_tbls(x)

## S3 method for class 'src_organism'
tbl(src, ...)

## S4 method for signature 'src_organism'
seqinfo(x)
```

Arguments

txdb	character(1) naming a TxDb.* package (e.g., TxDb.Hsapiens.UCSC.hg38.knownGene) or TxDb object instantiating the content of a TxDb.* pacakge.
dbpath	path and file name where SQLite file will be accessed or created if not already exists.
organism	organism or common name
genome	genome name
id	choose from "knownGene", "ensGene" and "refGene"

verbose	logical. Should R report extra information on progress? Default is TRUE.
.data	A tbl.
...	Comma separated list of unquoted expressions. You can treat variable names like they are positions. Use positive values to select variables; use negative values to drop variables.
x	A src_organism object
src	A src_organism object

Details

src_organism() and src_ucsc() are meant to be a building block for src_organism, which provides an integrated presentation of identifiers and genomic coordinates.

src_organism() creates a dplyr database integrating org.* and TxDb.* information by given TxDb. And src_ucsc() creates the database by given organism name, genome and/or id.

supportedOrganisms() provides all supported organisms in this package with corresponding OrgDb and TxDb.

Value

src_organism() and src_ucsc() returns a dplyr src_dbi instance representing the data tables.

Author(s)

Yubo Cheng.

See Also

dplyr for details about using dplyr to manipulate data.

transcripts_tbl for generic functions to extract genomic features from a src_organism object.

select,src_organism-method for "select" interface on src_organism objects.

Examples

```
## create human sqlite database with TxDb.Hsapiens.UCSC.hg38.knownGene and
## corresponding org.Hs.eg.db
## Not run: src <- src_organism("TxDb.Hsapiens.UCSC.hg38.knownGene")
src <- src_organism(dbpath=hg38light())

## query using dplyr
inner_join(tbl(src, "id"), tbl(src, "id_go")) %>%
     filter(symbol == "ADA") %>%
     dplyr::select(entrez, ensembl, symbol, go, evidence, ontology)

## create human sqlite database using hg38 genome
## Not run: human <- src_ucsc("human")

## all supported organisms with corresponding OrgDb and TxDb
supportedOrganisms()

## Look at all available tables
src_tbls(src)

## Look at data in table "id"
```

```
tbl(src, "id")

## Look at fields of one table
colnames(tbl(src, "id"))

## seqinfo of src_organism object
seqinfo(src)
```

---

transcripts_tbl *Extract genomic features from src_organism objects*

Description

Generic functions to extract genomic features from an object. This page documents the methods for src_organism objects only.

Usage

```
transcripts_tbl(x, filter = NULL)

## S4 method for signature 'src_organism'
transcripts(x, filter = NULL)

exons_tbl(x, filter = NULL)

## S4 method for signature 'src_organism'
exons(x, filter = NULL)

cds_tbl(x, filter = NULL)

## S4 method for signature 'src_organism'
cds(x, filter = NULL)

genes_tbl(x, filter = NULL)

## S4 method for signature 'src_organism'
genes(x, filter = NULL)

promoters_tbl(x, upstream, downstream, filter = NULL)

## S4 method for signature 'src_organism'
promoters(x, upstream, downstream, filter = NULL)

transcriptsBy_tbl(x, by = c("gene", "exon", "cds"), filter = NULL)

## S4 method for signature 'src_organism'
transcriptsBy(x, by = c("gene", "exon", "cds"),
  filter = NULL)

exonsBy_tbl(x, by = c("tx", "gene"), filter = NULL)
```

```
## S4 method for signature 'src_organism'
exonsBy(x, by = c("tx", "gene"), filter = NULL)

cdsBy_tbl(x, by = c("tx", "gene"), filter = NULL)

## S4 method for signature 'src_organism'
cdsBy(x, by = c("tx", "gene"), filter = NULL)

intronsByTranscript_tbl(x, filter = NULL)

## S4 method for signature 'src_organism'
intronsByTranscript(x, filter = NULL)

fiveUTRsByTranscript_tbl(x, filter = NULL)

## S4 method for signature 'src_organism'
fiveUTRsByTranscript(x, filter = NULL)

threeUTRsByTranscript_tbl(x, filter = NULL)

## S4 method for signature 'src_organism'
threeUTRsByTranscript(x, filter = NULL)
```

Arguments

x	A src_organism object
filter	Either NULL or a named list of vectors to be used to restrict the output. Filter can also be a GRanges object using "GRangesFilter" (see examples).
upstream	For promoters(): An integer(1) value indicating the number of bases upstream from the transcription start site.
downstream	For promoters(): An integer(1) value indicating the number of bases downstream from the transcription start site.
by	One of "gene", "exon", "cds" or "tx". Determines the grouping.

Details

These are the main functions for extracting transcript information from a src_organism object, inherited from transcripts in GenomicFeatures package. Two versions of results are provided: tibble (transcripts_tbl()) and GRanges or GRangesList (transcripts()).

Value

functions with _tbl return a tibble object, other methods return a GRanges or GRangesList object.

Author(s)

Yubo Cheng.

See Also

src_organism for creating a src_organism object.

Examples

```
## Not run: src <- src_ucsc("human")
src <- src_organism(dbpath=hg38light())

## transcript coordinates with filter in tibble format
filters <- list(SymbolFilter(c("A1BG", "CDH2")))
transcripts_tbl(src, filters)

transcripts_tbl(src, list(SymbolFilter("SNORD", "startsWith")))
transcripts_tbl(src, list(GoFilter("GO:0005615")))
transcripts_tbl(src, filter=list(
     SymbolFilter("SNORD", "startsWith"),
     TxStartFilter(25070000, "<")
))
## transcript coordinates with filter in granges format
filters <- list(GRangesFilter(GenomicRanges::GRanges("chr15:1-25070000")))
transcripts(src, filters)

## promoters
promoters(src, upstream=100, downstream=50,
          filter = list(SymbolFilter("ADA")))

## transcriptsBy
transcriptsBy(src, by = "exon", filter = list(SymbolFilter("ADA")))

## exonsBy
exonsBy(src, filter = list(SymbolFilter("ADA")))

## intronsByTranscript
intronsByTranscript(src, filter = list(SymbolFilter("ADA")))

## fiveUTRsByTranscript
fiveUTRsByTranscript(src, filter = list(SymbolFilter("ADA")))
```

Index