Package 'ensembldb'

April 23, 2016

Type Package

Title Utilities to create and use an Ensembl based annotation database

Version 1.2.2

Author Johannes Rainer <johannes.rainer@eurac.edu>,
Tim Triche <tim.triche@usc.edu>

Maintainer Johannes Rainer <johannes.rainer@eurac.edu>

URL https://github.com/jotsetung/ensembldb

BugReports https://github.com/jotsetung/ensembldb/issues

Imports methods, RSQLite, DBI, Biobase, GenomeInfoDb, AnnotationDbi (>= 1.31.19), rtracklayer, S4Vectors, AnnotationHub, Rsamtools, IRanges

Depends BiocGenerics (>= 0.15.10), GenomicRanges, GenomicFeatures

Suggests knitr, BiocStyle, EnsDb.Hsapiens.v75 (>= 0.99.7), RUnit, shiny

VignetteBuilder knitr

Description The package provides functions to create and use transcript centric annotation databases/packages. The annotation for the databases are directly fetched from Ensembl using their Perl API. The functionality and data is similar to that of the TxDb packages from the GenomicFeatures package, but, in addition to retrieve all gene/transcript models and annotations from the database, the ensembldb package provides also a filter framework allowing to retrieve annotations for specific entries like genes encoded on a chromosome region or transcript models of lincRNA genes.

Collate Classes.R dbhelpers.R Methods.R Methods-Filter.R loadEnsDb.R makeEnsemblDbPackage.R EnsDbFromGTF.R runEnsDbApp.R zzz.R

biocViews Genetics, AnnotationData, Sequencing, Coverage

License LGPL

NeedsCompilation no

R topics documented:

`EnsDb-class` *Basic usage of an Ensembl based annotation database*

Description

Get some basic information from an Ensembl based annotation package generated with `makeEnsembldbPackage`.

Usage

```
## S4 method for signature 'EnsDb'
buildQuery(x, columns=c("gene_id", "gene_biotype",
                                    "gene_name"), filter=list(), order.by,
                              order.type="asc", skip.order.check=FALSE)

## S4 method for signature 'EnsDb'
dbconn(x)

EnsDb(x)

## S4 method for signature 'EnsDb'
ensemblVersion(x)

## S4 method for signature 'EnsDb'
lengthOf(x, of="gene", filter=list())

## S4 method for signature 'EnsDb'
listColumns(x, table, skip.keys=TRUE, ...)

## S4 method for signature 'EnsDb'
listGenebiotypes(x, ...)

## S4 method for signature 'EnsDb'
listTxbiotypes(x, ...)

## S4 method for signature 'EnsDb'
listTables(x, ...)
```

```
## S4 method for signature 'EnsDb'
metadata(x, ...)

## S4 method for signature 'EnsDb'
organism(object)

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

Arguments

	(in alphabetic order)
	Additional arguments. Not used.
columns	Columns (attributes) to be retrieved from the database tables. Use the listColumns or listTables method for a list of supported columns.
filter	list of BasicFilter instance(s) to select specific entries from the database (see examples below).
object	For organism: an EnsDb instance.
of	for lengthOf: whether the length of genes or transcripts should be retrieved from the database.
order.by	name of one of the columns above on which the results should be sorted.
order.type	if the results should be ordered ascending (asc, default) or descending (desc).
skip.keys	for listColumns: whether primary and foreign keys (not being e.g. "gene_id" or alike) should be returned or not. By default these will not be returned.
skip.order.check	if paramter order.by should be checked for allowed column names. If TRUE the function checks if the provided order criteria orders on columns present in the database tables.
table	For listColumns: optionally specify the table name for which the columns should be returned.
x	For EnsDb: the file name of the SQLite database. For lengthOf: either an EnsDb or a GRangesList object. For all other methods an EnsDb instance.

Value

For buildQuery A character string with the SQL query.

For connection The SQL connection to the RSQLite database.

For EnsDb An EnsDb instance.

For lengthOf A named integer vector with the length of the genes or transcripts.

For listColumns A character vector with the column names.

For listGenebiotypes A character vector with the biotypes of the genes in the database.

For `listTxbiotypes` A character vector with the biotypes of the transcripts in the database.

For `listTables` A list with the names corresponding to the database table names and the elements being the attribute (column) names of the table.

For `metadata` A `data.frame`.

For `organism` A character string.

For `seqinfo` A `Seqinfo` class.

Objects from the Class

A connection to the respective annotation database is created upon loading of an annotation package created with the `makeEnsembldbPackage` function. In addition, the `EnsDb` constructor specifying the SQLite database file can be called to generate an instance of the object (see `makeEnsemblSQLiteFromTables` for an example).

Slots

`ensdb`**:** Object of class `"DBIConnection"`: the connection to the database.

`tables`**:** named list of database table columns with the names being the database table names. The tables are ordered by their degree, i.e. the number of other tables they can be joined with.

Methods and Functions

buildQuery Helper function building the SQL query to be used to retrieve the wanted information. Usually there is no need to call this method.

dbconn Returns the connection to the internal SQL database.

ensemblVersion Returns the Ensembl version on which the package was built.

lengthOf Retrieve the length of genes or transcripts from the database. The length is the sum of the lengths of all exons of a transcript or a gene. In the latter case the exons are first reduced so that the length corresponds to the part of the genomic sequence covered by the exons.

listColumns Lists all columns of all tables in the database, or, if `table` is specified, of the respective table.

listGenebiotypes Lists all gene biotypes defined in the database.

listTxbiotypes Lists all transcript biotypes defined in the database.

listTables Returns a named list of database table columns (names of the list being the database table names).

metadata Returns a `data.frame` with the metadata information from the database, i.e. informations about the Ensembl version or Genome build the database was build upon.

organism Returns the organism name (e.g. `"homo_sapiens"`).

seqinfo Returns the sequence/chromosome information from the database.

show Displays some informations from the database.

Author(s)

Johannes Rainer

See Also

makeEnsembldbPackage, BasicFilter, exonsBy, genes, transcripts, makeEnsemblSQLiteFromTables

Examples

```
library(EnsDb.Hsapiens.v75)

## display some information:
EnsDb.Hsapiens.v75

## show the tables along with its columns
listTables(EnsDb.Hsapiens.v75)

## for what species is this database?
organism(EnsDb.Hsapiens.v75)

## what Ensembl version if the database based on?
ensemblVersion(EnsDb.Hsapiens.v75)

## get some more information from the database
metadata(EnsDb.Hsapiens.v75)

######    buildQuery
##
## join tables gene and transcript and return gene_id and tx_id
buildQuery(EnsDb.Hsapiens.v75, columns=c("gene_id", "tx_id"))


## get all exon_ids and transcript ids of genes encoded on chromosome Y.
buildQuery(EnsDb.Hsapiens.v75, columns=c("exon_id", "tx_id"),
           filter=list(SeqnameFilter( "Y")))

#####    lengthOf
##
## length of a specific gene.
lengthOf(EnsDb.Hsapiens.v75,
         filter=list(GeneidFilter("ENSG00000000003")))

## length of a transcript
lengthOf(EnsDb.Hsapiens.v75, of="tx",
         filter=list(TxidFilter("ENST00000494424")))

## average length of all protein coding genes encoded on chromosomes X
## and Y
mean(lengthOf(EnsDb.Hsapiens.v75, of="gene",
              filter=list(GenebiotypeFilter("protein_coding"),
                  SeqnameFilter(c("X", "Y")))))

## average length of all snoRNAs
mean(lengthOf(EnsDb.Hsapiens.v75, of="gene",
              filter=list(GenebiotypeFilter("snoRNA"),
```

```
                  SeqnameFilter(c("X", "Y")))))

## list all available gene biotypes from the database:
listGenebiotypes(EnsDb.Hsapiens.v75)

## list all available transcript biotypes:
listTxbiotypes(EnsDb.Hsapiens.v75)
```

---

exonsBy *Retrieve annotation data from an Ensembl based package*

Description

Retrieve gene/transcript/exons annotations stored in an Ensembl based database package generated with the makeEnsembldbPackage function.

Usage

```
## S4 method for signature 'EnsDb'
exons(x, columns=listColumns(x,"exon"),
                        filter, order.by, order.type="asc",
                        return.type="GRanges")

## S4 method for signature 'EnsDb'
exonsBy(x, by=c("tx", "gene"),
                          columns=listColumns(x, "exon"), filter)

## S4 method for signature 'EnsDb'
transcripts(x, columns=listColumns(x, "tx"),
                             filter, order.by, order.type="asc",
                             return.type="GRanges")

## S4 method for signature 'EnsDb'
transcriptsBy(x, by=c("gene", "exon"),
                               columns=listColumns(x, "tx"), filter)

## S4 method for signature 'EnsDb'
promoters(x, upstream=2000, downstream=200, ...)

## S4 method for signature 'EnsDb'
genes(x, columns=listColumns(x, "gene"), filter,
                        order.by, order.type="asc",
                        return.type="GRanges")

## S4 method for signature 'EnsDb'
```

```
disjointExons(x, aggregateGenes=FALSE,
                                includeTranscripts=TRUE, filter, ...)

## S4 method for signature 'GRangesList'
toSAF(x, ...)

## S4 method for signature 'EnsDb'
getGenomeFaFile(x, pattern="dna.toplevel.fa")
```

Arguments

	(In alphabetic order)
	For promoters: additional arguments to be passed to the transcripts method.
aggregateGenes	For disjointExons: When FALSE (default) exon fragments that overlap multiple genes are dropped. When TRUE, all fragments are kept and the gene_id metadata column includes all gene IDs that overlap the exon fragment.
by	For exonsBy: wheter exons sould be fetched by genes or by transcripts; as in the corresponding function of the GenomicFeatures package. For transcriptsBy: whether transcripts should be fetched by genes or by exons; fetching transcripts by cds as supported by the transcriptsBy method in the GenomicFeatures package is currently not implemented.
columns	Columns to be retrieved from the database tables. Default values for genes are all columns from the gene database table, for exons and exonsBy the column names of the exon database table table and for transcript and transcriptBy the columns of the tx data base table (see details below for more information). Note that any of the column names of the database tables can be submitted to any of the methods (use listTables or listColumns methods for a complete list of allowed column names).
downstream	For method promoters: the number of nucleotides downstream of the transcription start site that should be included in the promoter region.
filter	A filter object extending BasicFilter or a list of such object(s) to select specific entries from the database (see examples below).
includeTranscripts	For disjointExons: When TRUE (default) a tx_name metadata column is included that lists all transcript IDs that overlap the exon fragment. Note: this is different to the disjointExons function in the GenomicFeatures package, that lists the transcript names, not IDs.
order.by	Name of one of the columns above on which the results should be sorted.
order.type	If the results should be ordered ascending (asc, default) or descending (desc).
pattern	For method getGenomeFaFile: the pattern to be used to identify the fasta file representing genomic DNA sequence.
return.type	Type of the returned object. Can be either "data.frame", "DataFrame" or "GRanges". In the latter case the return object will be a GRanges object with the

GRanges specifying the chromosomal start and end coordinates of the feature (gene, transcript or exon, depending whether `genes`, `transcripts` or `exons` was called). All additional columns are added as metadata columns to the GRanges object.

`upstream` For method `promoters`: the number of nucleotides upstream of the transcription start site that should be included in the promoter region.

`x` For `toSAF` a `GRangesList` object. For all other methods an `EnsDb` instance.

Details

A detailed description of all database tables and the associated attributes/column names is also given in the vignette of this package. An overview of the columns is given below:

gene_id the Ensembl gene ID of the gene.

gene_name the name of the gene (in most cases its official symbol).

entrezid the NCBI Entrezgene ID of the gene; note that this can also be a ";" separated list of IDs for Ensembl genes mapped to more than one Entrezgene.

gene_biotype the biotype of the gene.

gene_seq_start the start coordinate of the gene on the sequence (usually a chromosome).

gene_seq_end the end coordinate of the gene.

seq_name the name of the sequence the gene is encoded (usually a chromosome).

seq_strand the strand on which the gene is encoded

seq_coord_system the coordinate system of the sequence.

tx_id the Ensembl transcript ID.

tx_biotype the biotype of the transcript.

tx_seq_start the chromosomal start coordinate of the transcript.

tx_seq_end the chromosomal end coordinate of the transcript.

tx_cds_seq_start the start coordinate of the coding region of the transcript (NULL for non-coding transcripts).

tx_cds_seq_end the end coordinate of the coding region.

exon_id the ID of the exon. In Ensembl, each exon specified by a unique chromosomal start and end position has its own ID. Thus, the same exon might be part of several transcripts.

exon_seq_start the chromosomal start coordinate of the exon.

exon_seq_end the chromosomal end coordinate of the exon.

exon_idx the index of the exon in the transcript model. As noted above, an exon can be part of several transcripts and thus its position inside these transcript might differ.

Also, the vignette provides examples on how to retrieve sequences for genes/transcripts/exons.

Value

For exons, transcripts and genes, a data.frame, DataFrame or a GRanges, depending on the value of the return.type parameter. The result is ordered as specified by the parameter order.by, NOT by any ordering of values in eventually submitted filter objects.

For exonsBy, transcriptsBy: a GRangesList, depending on the value of the return.type parameter. The results are ordered by the value of the by parameter.

For toSAF: a data.frame with column names "GeneID" (the group name from the GRangesList, i.e. the ID by which the GRanges are split), "Chr" (the seqnames from the GRanges), "Start" (the start coordinate), "End" (the end coordinate) and "Strand" (the strand).

For disjointExons: a GRanges of non-overlapping exon parts.

For getGenomeFaFile: a FaFile-class object with the genomic DNA sequence.

Methods and Functions

exons Retrieve exon information from the database. Additional columns from transcripts or genes associated with the exons can be specified and are added to the respective exon annotation.

exonsBy Retrieve exons grouped by transcript or by gene. This function returns a GRangesList as does the analogous function in the GenomicFeatures package. Using the columns parameter it is possible to determine which additional values should be retrieved from the database. These will be included in the GRanges object for the exons as metadata columns. The exons in the inner GRanges are ordered by the exon index within the transcript (if by="tx"), or increasingly by the chromosomal start position of the exon or decreasingly by the chromosomal end position of the exon depending whether the gene is encoded on the + or - strand (for by="gene"). The GRanges in the GRangesList will be ordered by the name of the gene or transcript.

transcripts Retrieve transcript information from the database. Additional columns from genes or exons associated with the transcripts can be specified and are added to the respective transcript annotation.

transcriptsBy Retrieve transcripts grouped by gene or exon. This function returns a GRangesList as does the analogous function in the GenomicFeatures package. Using the columns parameter it is possible to determine which additional values should be retrieved from the database. These will be included in the GRanges object for the transcripts as metadata columns. The transcripts in the inner GRanges are ordered increasingly by the chromosomal start position of the transcript for genes encoded on the + strand and in a decreasing manner by the chromosomal end position of the transcript for genes encoded on the - strand. The GRanges in the GRangesList will be ordered by the name of the gene or exon.

promoters Retrieve promoter information from the database. Additional columns from genes or exons associated with the promoters can be specified and are added to the respective promoter annotation.

genes Retrieve gene information from the database. Additional columns from transcripts or exons associated with the genes can be specified and are added to the respective gene annotation.

disjointExons This method is identical to disjointExons defined in the GenomicFeatures package. It creates a GRanges of non-overlapping exon parts with metadata columns of gene_id and exonic_part. Exon parts that overlap more than one gene can be dropped with aggregateGenes=FALSE.

toSAF Reformats a GRangesList object into a data.frame corresponding to a standard SAF (Simplified Annotation Format) file (i.e. with column names "GeneID", "Chr", "Start", "End" and "Strand"). Note: this method makes only sense on a GRangesList that groups features (exons, transcripts) by gene.

getGenomeFaFile Returns a FaFile-class (defined in Rsamtools) with the genomic sequence of the genome build matching the Ensembl version of the EnsDb object. The file is retrieved using the AnnotationHub package. See the vignette for an example to work with such files.

Note

Ensembl defines genes not only on standard chromosomes, but also on patched chromosomes and chromosome variants. Thus it might be advisable to restrict the queries to just those chromosomes of interest (e.g. by specifying a SeqnameFilter(c(1:22, "X", "Y"))). In addition, also so called LRG genes (Locus Reference Genomic) are defined in Ensembl. Their gene id starts with LRG instead of ENS for Ensembl genes, thus, a filter can be applied to specifically select those genes or exclude those genes (see examples below).

Author(s)

Johannes Rainer, Tim Triche

See Also

makeEnsembldbPackage, BasicFilter, listColumns, lengthOf

Examples

```
library(EnsDb.Hsapiens.v75)

######   genes
##
## get all genes endcoded on chromosome Y
AllY <- genes(EnsDb.Hsapiens.v75, filter=SeqnameFilter("Y"))
AllY

## return result as DataFrame.
AllY.granges <- genes(EnsDb.Hsapiens.v75,
                      filter=SeqnameFilter("Y"),
                      return.type="DataFrame")
AllY.granges

## include all transcripts of the gene and their chromosomal
## coordinates, sort by chrom start of transcripts and return as
## GRanges.
AllY.granges.tx <- genes(EnsDb.Hsapiens.v75,
                         filter=SeqnameFilter("Y"),
                         columns=c("gene_id", "seq_name",
                             "seq_strand", "tx_id", "tx_biotype",
                             "tx_seq_start", "tx_seq_end"),
                         order.by="tx_seq_start")
```

```
AllY.granges.tx



######   transcripts
##
## get all transcripts of a gene
Tx <- transcripts(EnsDb.Hsapiens.v75,
                  filter=GeneidFilter("ENSG00000184895"),
                  order.by="tx_seq_start")
Tx

## get all transcripts of two genes along with some information on the
## gene and transcript
Tx <- transcripts(EnsDb.Hsapiens.v75,
                  filter=GeneidFilter(c("ENSG00000184895",
                      "ENSG00000092377")),
                      columns=c("gene_id", "gene_seq_start",
                          "gene_seq_end", "gene_biotype", "tx_biotype"))
Tx

######   promoters
##
## get the bona-fide promoters (2k up- to 200nt downstream of TSS)
promoters(EnsDb.Hsapiens.v75, filter=GeneidFilter(c("ENSG00000184895",
                                                    "ENSG00000092377")))

######   exons
##
## get all exons of the provided genes
Exon <- exons(EnsDb.Hsapiens.v75,
              filter=GeneidFilter(c("ENSG00000184895",
                  "ENSG00000092377")),
              order.by="exon_seq_start",
              columns=c( "gene_id", "gene_seq_start",
                  "gene_seq_end", "gene_biotype"))
Exon



#####    exonsBy
##
## get all exons for transcripts encoded on chromosomes X and Y.
ETx <- exonsBy(EnsDb.Hsapiens.v75, by="tx",
               filter=SeqnameFilter(c("X", "Y")))
ETx
## get all exons for genes encoded on chromosome 1 to 22, X and Y and
## include additional annotation columns in the result
EGenes <- exonsBy(EnsDb.Hsapiens.v75, by="gene",
                  filter=SeqnameFilter(c("X", "Y")),
                  columns=c("gene_biotype", "gene_name"))
EGenes
```

```
## Note that this might also contain "LRG" genes.
length(grep(names(EGenes), pattern="LRG"))

## to fetch just Ensemblgenes, use an GeneidFilter with value
## "ENS%" and condition "like"


#####    transcriptsBy
##
TGenes <- transcriptsBy(EnsDb.Hsapiens.v75, by="gene",
                        filter=SeqnameFilter(c("X", "Y")))
TGenes

## convert this to a SAF formatted data.frame that can be used by the
## featureCounts function from the Rsubreader package.
head(toSAF(TGenes))
```

GeneidFilter-class *Filter results fetched from the Ensembl database*

Description

These classes allow to specify which entries (i.e. genes, transcripts or exons) should be retrieved from the database.

Details

`ExonidFilter` Allows to filter based on the (Ensembl) exon identifier.

`EntrezidFilter` Filter results based on the NCBI Entrezgene identifierts of the genes. Use the `listGenebiotypes` method to get a complete list of all available gene biotypes.

`GenebiotypeFilter` Filter results based on the gene biotype as defined in the Ensembl database.

`GeneidFilter` Filter results based on the Ensembl gene identifiers.

`GenenameFilter` Allows to filter on the gene names (symbols) of the genes.

`SeqendFilter` Filter based on the chromosomal end coordinate of the exons, transcripts or genes.

`SeqnameFilter` Filter on the sequence name on which the features are encoded (mostly the chromosome names).

`SeqstartFilter` Filter based on the chromosomal start coordinates of the exons, transcripts or genes.

`SeqstrandFilter` Filter based on the strand on which the features are encoded.

`TxbiotypeFilter` Filter on the transcript biotype defined in Ensembl. Use the `listTxbiotypes` method to get a complete list of all available transcript biotypes.

`TxidFilter` Filter on the Ensembl transcript identifiers.

Objects from the Class

While objects can be created by calls e.g. of the form `new("GeneidFilter", ...)` users are strongly encouraged to use the specific functions: `GeneidFilter`, `EntrezidFilter`, `GenenameFilter`, `GenebiotypeFilter`, `TxidFilter`, `TxbiotypeFilter`, `ExonidFilter`, `SeqnameFilter`, `SeqstrandFilter`, `SeqstartFilter` and `SeqendFilter`. See examples below for usage.

Slots

`condition`**:** Object of class `"character"`: can be either `"="`, `"in"` or `"like"` to filter on character values (e.g. gene id, gene biotype, seqname etc), or `"="`, `">"` or `"<"` for numerical values (chromosome/seq coordinates). Note that for `"like"` value should be a SQL pattern (e.g. `"ENS%"`).

`value`**:** Object of class `"character"`: the value to be used for filtering.

Extends

Class `BasicFilter`, directly.

Methods

Note: these methods are applicable to all classes extending the `BasicFilter` class.

`signature(object = "GeneidFilter", db="EnsDb",with.tables="character")`: returns the column (attribute name) to be used for the filtering. Submitting the `db` parameter ensures that returned column is valid in the corresponding database schema. The optional argument `with.tables` allows to specify which in which database table the function should look for the attribute/column name. By default the method will check all database tables.

colu~~colu~~mn `signature(object = "GeneidFilter", db="EnsDb",with.tables="missing")`: returns the column (attribute name) to be used for the filtering. Submitting the `db` parameter ensures that returned column is valid in the corresponding database schema.

column `signature(object = "GeneidFilter", db="missing",with.tables="missing")`: returns the column (table column name) to be used for the filtering.

condition `signature(x="BasicFilter")`: returns the value for the `condition` slot.

where `signature(object = "GeneidFilter", db="EnsDb",with.tables="character")`: returns the where clause for the SQL call. Submitting also the `db` parameter ensures that the columns are valid in the corresponding database schema. The optional argument `with.tables` allows to specify which in which database table the function should look for the attribute/column name. By default the method will check all database tables.

where `signature(object = "GeneidFilter", db="EnsDb",with.tables="missing")`: returns the where clause for the SQL call. Submitting also the `db` parameter ensures that the columns are valid in the corresponding database schema.

where `signature(object = "GeneidFilter", db="missing",with.tables="missing")`: returns the where clause for the SQL call.

Note

The column and where methods should be always called along with the EnsDb object, as this ensures that the returned column names are valid for the database schema. The optional argument with.tables should on the other hand only be used rarely as it is more intended for internal use.

Note that the database column "entrezid" queried for EntrezidFilter classes can contain multiple, ";" separated, Entrezgene IDs, thus, using this filter at present might not return all entries from the database.

Author(s)

Johannes Rainer

See Also

genes, transcripts, exons, listGenebiotypes, listTxbiotypes

Examples

```
## create a filter that could be used to retrieve all informations for
## the respective gene.
Gif <- GeneidFilter("ENSG00000012817")
Gif
## returns the where clause of the SQL querys
where(Gif)

## create a filter for a chromosomal end position of a gene
Sef <- SeqendFilter(10000, condition=">", "gene")
Sef

## for additional examples see the help page of "genes"
```

---

makeEnsembldbPackage *Generating a Ensembl annotation package from Ensembl*

Description

These functions allow to retrieve annotations from the Ensembl database (fetchTablesFromEnsembl) create an SQLite database from these (makeEnsemblSQLiteFromTables) and to generate an annotation package providing access to this resource (makeEnsembldbPackage).

Usage

```
ensDbFromGRanges(x, outfile, path, organism, genomeVersion,
                 version, verbose=FALSE)

ensDbFromGtf(gtf, outfile, path, organism, genomeVersion,
             version, verbose=FALSE)

fetchTablesFromEnsembl(version, ensemblapi, user="anonymous",
                       host="ensembldb.ensembl.org", pass="",
                       port=5306, species="human")

makeEnsemblSQLiteFromTables(path=".", dbname)

makeEnsembldbPackage(ensdb, version, maintainer, author,
                     destDir=".", license="Artistic-2.0")
```

Arguments

	(in alphabetical order)
	The author of the package.
dbname	The name for the database (optional). By default a name based on the species and Ensembl version will be automatically generated (and returned by the function).
destDir	Where the package should be saved to.
ensdb	The file name of the SQLite database generated by `makeEnsemblSQLiteFromTables`.
ensemblapi	The path to the Ensembl perl API installed locally on the system. The Ensembl perl API version has to fit the version.
genomeVersion	For `ensDbFromGtf`: the version of the genome (e.g. `"GRCh37"`). If not provided the function will try to guess it from the file name (assuming file name convention of Ensembl GTF files).
gtf	The GTF file name.
host	The hostname to access the Ensembl database.
license	The license of the package.
maintainer	The maintainer of the package.
organism	For `ensDbFromGtf`: the organism name (e.g. `"Homo_sapiens"`). If not provided the function will try to guess it from the file name (assuming file name convention of Ensembl GTF files).
outfile	The desired file name of the SQLite file. If not provided the name of the GTF file will be used.
pass	The password for the Ensembl database.
path	The directory in which the tables retrieved by `fetchTablesFromEnsembl` or the SQLite database file generated by `ensDbFromGtf` are stored.

port	The port to be used to connect to the Ensembl database.
species	The species for which the annotations should be retrieved.
user	The username for the Ensembl database.
verbose	If progress messages should be shown.
version	For fetchTablesFromEnsembl, ensDbFromGRanges and ensDbFromGtf: the Ensembl version for which the annotation should be retrieved (e.g. 75). The ensDbFromGtf function will try to guess the Ensembl version from the GTF file name if not provided. For makeEnsemblDbPackage: the version for the package.
x	For ensDbFromGRanges: the GRanges object.

Details

The fetchTablesFromEnsembl function internally calls the perl script get_gene_transcript_exon_tables.pl to retrieve all required information from the Ensembl database using the Ensembl perl API.

As an alternative way, a EnsDb database file can be generated by the ensDbFromGtf from a GTF file from Ensembl or with the ensDbFromGRanges from a GRanges object e.g. retrieved from the AnnotationHub package. The returned database file name can then be used as an input to the makeEnsembldbPackage.

Value

makeEnsemblSQLiteFromTables, ensDbFromGRanges and ensDbFromGtf: the name of the SQLite file.

Note

A local installation of the Ensembl perl API is required for the fetchTablesFromEnsembl. See http://www.ensembl.org/info/docs/api/api_installation.html for installation inscructions.

A database generated from a GTF file lacks some features as they are not available in the GTF files from Ensembl. These are: chromosome lengths, NCBI Entrezgene IDs.

GRanges objects provided by the AnnotationHub package on the other hand contain already the required seqinfo information (chromosome length etc) to build an EnsDb database using the ensDbFromGRanges.

Author(s)

Johannes Rainer

See Also

EnsDb, genes

Examples

```
## Not run:

## get all human gene/transcript/exon annotations from Ensembl (75)
## the resulting tables will be stored by default to the current working
## directory; if the correct Ensembl api (version 75) is defined in the
## PERL5LIB environment variable, the ensemblapi parameter can also be omitted.
fetchTablesFromEnsembl(75,
                       ensemblapi="/home/bioinfo/ensembl/75/API/ensembl/modules",
                       species="human")

## These tables can then be processed to generate a SQLite database
## containing the annotations
DBFile <- makeEnsemblSQLiteFromTables()

## and finally we can generate the package
makeEnsembldbPackage(ensdb=DBFile, version="0.0.1",
                     maintainer="Johannes Rainer <johannes.rainer@eurac.edu>",
                     author="J Rainer")

## Build an annotation file from a GTF file.
## the GTF file can be downloaded from
## ftp://ftp.ensembl.org/pub/release-75/gtf/homo_sapiens/
gtffile <- "Homo_sapiens.GRCh37.75.gtf.gz"
## generate the SQLite database file
DB <- ensDbFromGtf(gtf=paste0(ensemblhost, gtffile), verbose=TRUE)

## load the DB file directly
EDB <- EnsDb(DB)


## End(Not run)

## Generate a sqlite database for genes encoded on chromosome Y
chrY <- system.file("chrY", package="ensembldb")
DBFile <- makeEnsemblSQLiteFromTables(path=chrY ,dbname=tempfile())
## load this database:
edb <- EnsDb(DBFile)

edb

## Generate a sqlite database from a GRanges object specifying
## genes encoded on chromosome Y
load(system.file("YGRanges.RData", package="ensembldb"))

Y

DB <- ensDbFromGRanges(Y, path=tempdir(), version=75,
                       organism="Homo_sapiens")
edb <- EnsDb(DB)
```

runEnsDbApp *Search annotations interactively*

Description

This function starts the interactive EnsDb shiny web application that allows to look up gene/transcript/exon annotations from an EnsDb annotation package installed locally.

Usage

```
runEnsDbApp(...)
```

Arguments

...	Additional arguments passed to the runApp function from the shiny package.

Details

The shiny based web application allows to look up any annotation available in any of the locally installed EnsDb annotation packages.

Value

Nothing is returned by the function.

Author(s)

Johannes Rainer

See Also

EnsDb, genes

SeqendFilter *Constructor functions for filter objects*

Description

These functions allow to create filter objects that can be used to retrieve specific elements from the annotation database.

Usage

```
EntrezidFilter(value, condition = "=")
GeneidFilter(value, condition = "=")
GenenameFilter(value, condition = "=")
GenebiotypeFilter(value, condition = "=")
TxidFilter(value, condition = "=")
TxbiotypeFilter(value, condition = "=")
ExonidFilter(value, condition = "=")
SeqnameFilter(value, condition = "=")
SeqstrandFilter(value, condition = "=")
SeqstartFilter(value, condition = "=", feature = "gene")
SeqendFilter(value, condition = "=", feature = "gene")
```

Arguments

value	The filter value, e.g., for GeneidFilter the id of the gene for which the data should be retrieved. For character values (all filters except SeqstartFilter and SeqendFilter) also a character vector of values is allowed. Allowed values for SeqstrandFilter are: "+", "-", "1" or "-1".
condition	The condition to be used in the comparison. For character values "=", "in" and "like" are allowed, for numeric values (SeqstartFilter and SeqendFilter) "=", ">", ">=", "<" and "<=". Note that for "like" value should be a SQL pattern (e.g. "ENS%").
feature	For SeqstartFilter and SeqendFilter: the chromosomal position of which features should be used in the filter (either "gene", "transcript" or "exon")?

Details

EntrezidFilter Filter results based on the NCBI Entrezgene ID of the genes.

GeneidFilter Filter results based on Ensembl gene IDs.

GenenameFilter Filter results based on gene names (gene symbols).

GenebiotypeFilter Filter results based on the biotype of the genes. For a complete list of available gene biotypes use the listGenebiotypes method.

TxidFilter Filter results based on the Ensembl transcript IDs.

TxbiotypeFilter Filter results based on the biotype of the transcripts. For a complete list of available transcript biotypes use the listTxbiotypes method.

`ExonidFilter` Filter based on the Ensembl exon ID.

`SeqnameFilter` Filter results based on the name of the sequence the features are encoded.

`SeqstrandFilter` Filter results based on the strand on which the features are encoded.

`SeqstartFilter` Filter results based on the (chromosomal) start coordinate of the features (exons, genes or transcripts).

`SeqendFilter` Filter results based on the (chromosomal) end coordinates.

Value

Depending on the function called an instance of: `EntrezidFilter`, `GeneidFilter`, `GenenameFilter`, `GenebiotypeFilter`, `TxidFilter`, `TxbiotypeFilter`, `ExonidFilter`, `SeqnameFilter`, `SeqstrandFilter`, `SeqstartFilter`, `SeqendFilter`

Author(s)

Johannes Rainer

See Also

`EntrezidFilter`, `GeneidFilter`, `GenenameFilter`, `GenebiotypeFilter`, `TxidFilter`, `TxbiotypeFilter`, `ExonidFilter`, `SeqnameFilter`, `SeqstrandFilter`, `SeqstartFilter`, `SeqendFilter`

Examples

```
## create a filter that could be used to retrieve all informations for
## the respective gene.
Gif <- GeneidFilter("ENSG00000012817")
Gif
## returns the where clause of the SQL querys
where(Gif)

## create a filter for a chromosomal end position of a gene
Sef <- SeqendFilter(100000, condition="<", "gene")
Sef

## To find genes within a certain chromosomal position filters should be
## combined:
Ssf <- SeqstartFilter(10000, condition=">", "gene")
Snf <- SeqnameFilter("2")
## combine the filters
Filter <- list(Ssf, Sef, Snf)

Filter

## generate the where SQL call for these filters:
where(Filter)
```

Index