Package: txdbmaker (via r-universe)

November 12, 2026

Title Tools for making TxDb objects from genomic annotations

Description A set of tools for making TxDb objects from genomic annotations from various sources (e.g. UCSC, Ensembl, and GFF files). These tools allow the user to download the genomic locations of transcripts, exons, and CDS, for a given assembly, and to import them in a TxDb object. TxDb objects are implemented in the GenomicFeatures package, together with flexible methods for extracting the desired features in convenient formats.

biocViews Infrastructure, DataImport, Annotation, GenomeAnnotation, GenomeAssembly, Genetics, Sequencing

URL https://bioconductor.org/packages/txdbmaker

BugReports https://github.com/Bioconductor/txdbmaker/issues

Version 1.8.0

License Artistic-2.0

Encoding UTF-8

Depends BiocGenerics, S4Vectors (>= 0.47.6), Seqinfo, GenomicRanges (>= 1.61.1), GenomicFeatures (>= 1.61.4)

Imports methods, utils, stats, tools, httr, rjson, DBI, RSQLite (>= 2.0), IRanges, UCSC.utils (>= 1.7.1), GenomeInfoDb, AnnotationDbi, Biobase, BiocIO, rtracklayer, biomaRt (>= 2.59.1)

Suggests RMariaDB, ensembldb, GenomeInfoDbData, RUnit, BiocStyle, knitr

VignetteBuilder knitr

Collate utils.R Ensembl-utils.R findCompatibleMarts.R TxDb-schema.R TxDb-CREATE-TABLE-helpers.R makeTxDb.R makeTxDbFromUCSC.R makeTxDbFromBiomart.R makeTxDbFromEnsembl.R makeTxDbFromGRanges.R makeTxDbFromGFF.R makeFeatureDbFromUCSC.R makeTxDbPackage.R zzz.R

Config/pak/sysreqs make libbz2-dev libicu-dev liblzma-dev libpng-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 13:02:30 UTC

RemoteUrl https://github.com/bioc/txdbmaker

RemoteRef RELEASE_3_23

RemoteSha b7845a9f37a158b242a60f1ce26c363bdac9c9b2

Contents

txdbmaker-package 2
makeFeatureDbFromUCSC 3
makeTxDb 5
makeTxDbFromBiomart 8
makeTxDbFromEnsembl 13
makeTxDbFromGFF 15
makeTxDbFromGRanges 17
makeTxDbFromUCSC 19
makeTxDbPackage 21

Index **26**

`txdbmaker-package` *Tools for making TxDb objects from genomic annotations*

Description

The **txdbmaker** package contains a set of tools for making TxDb objects from genomic annotations from various sources (e.g. UCSC, Ensembl, and GFF files). These tools allow the user to download the genomic locations of transcripts, exons, and CDS, for a given assembly, and to import them in a TxDb object.

Note that TxDb objects are implemented in the **GenomicFeatures** package, together with flexible methods for extracting the desired features in convenient formats.

Details

For a quick overview of the provided tools, please see the "Making TxDb Objects" vignette included in this package.

To access the vignette from your R session, run `browseVignettes(package="txdbmaker")`. This requires the **txdbmaker** package to be already installed.

Alternatively this vignette should also be available online here: https://bioconductor.org/packages/release/bioc/vignettes/txdbmaker/inst/doc/txdbmaker.html

makeFeatureDbFromUCSC *Making a FeatureDb object from annotations available at the UCSC Genome Browser*

Description

WARNING: The FeatureDb/makeFeatureDbFromUCSC/features code base is no longer actively maintained and FeatureDb-related functionalities might get deprecated in the near future.

The `makeFeatureDbFromUCSC` function allows the user to make a FeatureDb object from simple annotation tracks at UCSC. The tracks in question must (at a minimum) have a start, end and a chromosome affiliation in order to be made into a FeatureDb. This function requires a precise declaration of its first three arguments to indicate which genome, track and table wish to be imported. There are discovery functions provided to make this process go smoothly.

Usage

```
UCSCFeatureDbTableSchema(genome,
                         track,
                         tablename)

makeFeatureDbFromUCSC(
        genome,
        track,
        tablename,
        columns = UCSCFeatureDbTableSchema(genome,track,tablename),
        url="https://genome.ucsc.edu/cgi-bin/",
        goldenPath.url=getOption("UCSC.goldenPath.url"),
        chromCol,
        chromStartCol,
        chromEndCol,
        taxonomyId=NA)
```

Arguments

`genome`	genome abbreviation used by UCSC and listed in `list_UCSC_genomes()[ , "genome"]`. For example: `"hg18"`.
`track`	Deprecated.
`tablename`	name of the UCSC table containing the annotations to retrieve. Use `list_UCSC_tracks` from the **UCSC.utils** package to get the list of available tracks and associated primary tables for a particular genome.
`columns`	a named character vector to list out the names and types of the other columns that the downloaded track should have. Use `UCSCFeatureDbTableSchema` to retrieve this information for a particular table.
`url`, `goldenPath.url`	Deprecated.

chromCol	If the schema comes back and the 'chrom' column has been labeled something other than 'chrom', use this argument to indicate what that column has been labeled as so we can properly designate it. This could happen (for example) with the knownGene track tables, which has no 'chromStart' or 'chromEnd' columns, but which DOES have columns that could reasonably substitute for these columns under particular circumstances. Therefore we allow these three columns to have arguments so that their definition can be re-specified
chromStartCol	Same thing as chromCol, but for renames of 'chromStart'
chromEndCol	Same thing as chromCol, but for renames of 'chromEnd'
taxonomyId	By default this value is NA and the organism inferred will be used to look up the correct value for this. But you can use this argument to override that and supply your own valid taxId here.

Details

makeFeatureDbFromUCSC is a convenience function that builds a tiny database from one of the UCSC track tables.

UCSCFeatureDbTableSchema is a convenience function that creates a named vector of types for all the fields that can potentially be supported for a given track. By default, this will be called on your specified tablename to include all of the fields in a track.

Value

A FeatureDb object for makeFeatureDbFromUCSC.

Author(s)

M. Carlson

See Also

list_UCSC_genomes and list_UCSC_tracks in the **UCSC.utils** package.

Examples

```
## Display the list of genomes available at UCSC:
library(UCSC.utils)
list_UCSC_genomes()[ , "genome"]

## Display the list of tracks available for your genome:
list_UCSC_tracks("mm10")

## Lookup the primary table associated with your track of interest:
subset(list_UCSC_tracks("mm10"), track == "qPCR Primers")

## Display fields that could be passed in to colnames:
UCSCFeatureDbTableSchema(genome="mm10",
                         tablename="qPcrPrimers")

## Retrieving a full transcript dataset for Mouse from UCSC:
```

```
fdb <- makeFeatureDbFromUCSC(genome="mm10",
                             tablename="qPcrPrimers")
fdb
```

---

makeTxDb *Making a TxDb object from user supplied annotations*

Description

makeTxDb is a low-level constructor for making a TxDb object from user supplied transcript annotations.

Note that the end user will rarely need to use makeTxDb directly but will typically use one of the high-level constructors makeTxDbFromUCSC, makeTxDbFromEnsembl, or makeTxDbFromGFF.

Usage

```
makeTxDb(transcripts, splicings, genes=NULL,
         chrominfo=NULL, metadata=NULL,
         reassign.ids=FALSE, on.foreign.transcripts=c("error", "drop"))
```

Arguments

transcripts	Data frame containing the genomic locations of a set of transcripts.
splicings	Data frame containing the genomic locations of exons and CDS parts of the transcripts in transcripts.
genes	Data frame containing the genes associated to a set of transcripts.
chrominfo	Data frame containing information about the chromosomes hosting the set of transcripts.
metadata	2-column data frame containing meta information about this set of transcripts like organism, genome, UCSC table, etc... The names of the columns must be "name" and "value" and their type must be character.
reassign.ids	TRUE or FALSE. Controls how internal ids should be assigned for each type of feature i.e. for transcripts, exons, and CDS parts. For each type, if reassign.ids is FALSE (the default) and if the ids are supplied, then they are used as the internal ids, otherwise the internal ids are assigned in a way that is compatible with the order defined by ordering the features first by chromosome, then by strand, then by start, and finally by end.
on.foreign.transcripts	Controls what to do when the input contains *foreign transcripts* i.e. transcripts that are on sequences not in chrominfo. If set to "error" (the default)

Details

The `transcripts` (required), `splicings` (required) and `genes` (optional) arguments must be data frames that describe a set of transcripts and the genomic features related to them (exons, CDS parts, and genes at the moment). The `chrominfo` (optional) argument must be a data frame containing chromosome information like the length of each chromosome.

`transcripts` must have 1 row per transcript and the following columns:

- `tx_id`: Transcript ID. Integer vector. No NAs. No duplicates.
- `tx_chrom`: Transcript chromosome. Character vector (or factor) with no NAs.
- `tx_strand`: Transcript strand. Character vector (or factor) with no NAs where each element is either `"+"` or `"-"`.
- `tx_start`, `tx_end`: Transcript start and end. Integer vectors with no NAs.
- `tx_name`: [optional] Transcript name. Character vector (or factor). NAs and/or duplicates are ok.
- `tx_type`: [optional] Transcript type (e.g. mRNA, ncRNA, snoRNA, etc...). Character vector (or factor). NAs and/or duplicates are ok.
- `gene_id`: [optional] Associated gene. Character vector (or factor). NAs and/or duplicates are ok.

Other columns, if any, are ignored (with a warning).

`splicings` must have N rows per transcript, where N is the nb of exons in the transcript. Each row describes an exon plus, optionally, the CDS part associated with this exon. Its columns must be:

- `tx_id`: Foreign key that links each row in the `splicings` data frame to a unique row in the `transcripts` data frame. Note that more than 1 row in `splicings` can be linked to the same row in `transcripts` (many-to-one relationship). Same type as `transcripts$tx_id` (integer vector). No NAs. All the values in this column must be present in `transcripts$tx_id`.
- `exon_rank`: The rank of the exon in the transcript. Integer vector with no NAs. (`tx_id`, `exon_rank`) pairs must be unique.
- `exon_id`: [optional] Exon ID. Integer vector with no NAs.
- `exon_name`: [optional] Exon name. Character vector (or factor). NAs and/or duplicates are ok.
- `exon_chrom`: [optional] Exon chromosome. Character vector (or factor) with no NAs. If missing then `transcripts$tx_chrom` is used. If present then `exon_strand` must also be present.
- `exon_strand`: [optional] Exon strand. Character vector (or factor) with no NAs. If missing then `transcripts$tx_strand` is used and `exon_chrom` must also be missing.
- `exon_start`, `exon_end`: Exon start and end. Integer vectors with no NAs.
- `cds_id`: [optional] ID of the CDS part associated with the exon. Integer vector. If present then `cds_start` and `cds_end` must also be present. NAs are allowed and must match those in `cds_start` and `cds_end`.
- `cds_name`: [optional] Name of the CDS part. Character vector (or factor). If present then `cds_start` and `cds_end` must also be present. NAs and/or duplicates are ok. Must contain NAs at least where `cds_start` and `cds_end` contain them.

- cds_start, cds_end: [optional] Start/end of the CDS part. Integer vectors. If one of the 2 columns is missing then all cds_* columns must be missing. NAs are allowed and must occur at the same positions in cds_start and cds_end.
- cds_phase: [optional] Phase of the CDS part. Integer vector. If present then cds_start and cds_end must also be present. NAs are allowed and must match those in cds_start and cds_end.

Other columns, if any, are ignored (with a warning).

genes should not be supplied if transcripts has a gene_id column. If supplied, it must have N rows per transcript, where N is the nb of genes linked to the transcript (N will be 1 most of the time). Its columns must be:

- tx_id: [optional] genes must have either a tx_id or a tx_name column but not both. Like splicings$tx_id, this is a foreign key that links each row in the genes data frame to a unique row in the transcripts data frame.
- tx_name: [optional] Can be used as an alternative to the genes$tx_id foreign key.
- gene_id: Gene ID. Character vector (or factor). No NAs.

Other columns, if any, are ignored (with a warning).

chrominfo must have 1 row per chromosome and the following columns:

- chrom: Chromosome name. Character vector (or factor) with no NAs and no duplicates.
- length: Chromosome length. Integer vector with either all NAs or no NAs.
- is_circular: [optional] Chromosome circularity flag. Logical vector. NAs are ok.

Other columns, if any, are ignored (with a warning).

Value

A TxDb object.

Author(s)

Hervé Pagès

See Also

- makeTxDbFromUCSC, makeTxDbFromBiomart, and makeTxDbFromEnsembl, for making a TxDb object from online resources.
- makeTxDbFromGRanges and makeTxDbFromGFF for making a TxDb object from a GRanges object, or from a GFF or GTF file.
- TxDb objects implemented in the **GenomicFeatures** package.
- saveDb and loadDb in the **AnnotationDbi** package for saving and loading a TxDb object as an SQLite file.

Examples

```
transcripts <- data.frame(
                  tx_id=1:3,
                  tx_chrom="chr1",
                  tx_strand=c("-", "+", "+"),
                  tx_start=c(1, 2001, 2001),
                  tx_end=c(999, 2199, 2199))
splicings <-  data.frame(
                  tx_id=c(1L, 2L, 2L, 2L, 3L, 3L),
                  exon_rank=c(1, 1, 2, 3, 1, 2),
                  exon_start=c(1, 2001, 2101, 2131, 2001, 2131),
                  exon_end=c(999, 2085, 2144, 2199, 2085, 2199),
                  cds_start=c(1, 2022, 2101, 2131, NA, NA),
                  cds_end=c(999, 2085, 2144, 2193, NA, NA),
                  cds_phase=c(0, 0, 2, 0, NA, NA))

txdb <- makeTxDb(transcripts, splicings)
```

---

makeTxDbFromBiomart *Make a TxDb object from annotations available on a BioMart database*

Description

The makeTxDbFromBiomart function allows the user to make a TxDb object from transcript annotations available on a BioMart database.

Note that makeTxDbFromBiomart is being phased out in favor of makeTxDbFromEnsembl.

Usage

```
makeTxDbFromBiomart(biomart="ENSEMBL_MART_ENSEMBL",
                    dataset="hsapiens_gene_ensembl",
                    transcript_ids=NULL,
                    circ_seqs=NULL,
                    filter=NULL,
                    id_prefix="ensembl_",
                    host="https://www.ensembl.org",
                    taxonomyId=NA)

getChromInfoFromBiomart(biomart="ENSEMBL_MART_ENSEMBL",
                        dataset="hsapiens_gene_ensembl",
                        id_prefix="ensembl_",
                        host="https://www.ensembl.org")
```

Arguments

biomart	which BioMart database to use. Get the list of all available BioMart databases with the listMarts function from the biomaRt package. See the details section below for a list of BioMart databases with compatible transcript annotations.

dataset	which dataset from BioMart. For example: "hsapiens_gene_ensembl", "mmusculus_gene_ensembl", "dmelanogaster_gene_ensembl", "celegans_gene_ensembl", etc in the ensembl database. See the examples section below for how to discover which datasets are available in a given BioMart database.
transcript_ids	optionally, only retrieve transcript annotation data for the specified set of transcript ids. If this is used, then the meta information displayed for the resulting TxDb object will say 'Full dataset: no'. Otherwise it will say 'Full dataset: yes'.
circ_seqs	a character vector to list out which chromosomes should be marked as circular.
filter	Additional filters to use in the BioMart query. Must be a named list. An example is filter=list(source="entrez")
id_prefix	Specifies the prefix used in BioMart attributes. For example, some BioMarts may have an attribute specified as "ensembl_transcript_id" whereas others have the same attribute specified as "transcript_id". Defaults to "ensembl_".
host	The host URL of the BioMart. Defaults to www.ensembl.org.
taxonomyId	By default this value is NA and the dataset selected will be used to look up the correct value for this. But you can use this argument to override that and supply your own taxId here (which will be independently checked to make sure its a real taxonomy id). Normally you should never need to use this.

Details

makeTxDbFromBiomart is a convenience function that feeds data from a BioMart database to the lower level makeTxDb function. See ?makeTxDbFromUCSC for a similar function that feeds data from the UCSC source.

Here is a list of datasets known to be compatible with makeTxDbFromBiomart (list updated on September 18, 2017):

1. All the datasets in the main Ensembl database. Get the list with:

```
mart <- biomaRt::useEnsembl(biomart="ENSEMBL_MART_ENSEMBL",
                            host="https://www.ensembl.org")
biomaRt::listDatasets(mart)
```

2. All the datasets in the Ensembl Fungi database. Get the list with:

```
mart <- biomaRt::useEnsemblGenomes(biomart="fungi_mart")
biomaRt::listDatasets(mart)
```

3. All the datasets in the Ensembl Metazoa database. Get the list with:

```
mart <- biomaRt::useEnsemblGenomes(biomart="metazoa_mart")
biomaRt::listDatasets(mart)
```

4. All the datasets in the Ensembl Plants database. Get the list with:

```
mart <- biomaRt::useEnsemblGenomes(biomart="plants_mart")
biomaRt::listDatasets(mart)
```

5. All the datasets in the Ensembl Protists database. Get the list with:

```
mart <- biomaRt::useEnsemblGenomes(biomart="protists_mart")
biomaRt::listDatasets(mart)
```

6. All the datasets in the Gramene Mart. Get the list with:

```
mart <- biomaRt::useEnsembl(biomart="ENSEMBL_MART_PLANT",
                            host="https://ensembl.gramene.org")
biomaRt::listDatasets(mart)
```

Note that BioMart is not currently available for Ensembl Bacteria.

Also please note that not all these datasets have CDS information.

Value

A TxDb object for `makeTxDbFromBiomart`.

A data frame with 1 row per chromosome (or scaffold) and with columns `chrom` and `length` for `getChromInfoFromBiomart`.

Author(s)

M. Carlson and H. Pagès

See Also

- `makeTxDbFromUCSC` and `makeTxDbFromEnsembl` for making a TxDb object from other online resources.
- `makeTxDbFromGRanges` and `makeTxDbFromGFF` for making a TxDb object from a GRanges object, or from a GFF or GTF file.
- The `listMarts`, `useEnsembl`, `listDatasets`, and `listFilters` functions in the **biomaRt** package.
- TxDb objects implemented in the **GenomicFeatures** package.
- `makeTxDb` for the low-level function used by the `makeTxDbFrom*` functions to make the TxDb object returned to the user.

Examples

```
## ---------------------------------------------------------------------
## A. BASIC USAGE
## ---------------------------------------------------------------------

## We can use listDatasets() from the biomaRt package to list the
## datasets available in the "ENSEMBL_MART_ENSEMBL" BioMart database:
library(biomaRt)
listMarts(host="https://www.ensembl.org")
mart <- useEnsembl(biomart="ENSEMBL_MART_ENSEMBL", host="https://www.ensembl.org")
datasets <- listDatasets(mart)
head(datasets)
subset(datasets, grepl("elegans", dataset, ignore.case=TRUE))

## Retrieve the full transcript dataset for Worm:
txdb1 <- makeTxDbFromBiomart(dataset="celegans_gene_ensembl")
txdb1
```

```
## Retrieve an incomplete transcript dataset for Human:
transcript_ids <- c(
    "ENST00000013894",
    "ENST00000268655",
    "ENST00000313243",
    "ENST00000435657",
    "ENST00000384428",
    "ENST00000478783"
)

if (interactive()) {
  txdb2 <- makeTxDbFromBiomart(dataset="hsapiens_gene_ensembl",
                               transcript_ids=transcript_ids)
  txdb2  # note that these annotations match the GRCh38 genome assembly
}

## ---------------------------------------------------------------------
## B. ACCESSING THE EnsemblGenomes MARTS
## ---------------------------------------------------------------------

library(biomaRt)

## Note that BioMart is not currently available for Ensembl Bacteria.

## ---------------------
## --- Ensembl Fungi ---

mart <- useEnsemblGenomes(biomart="fungi_mart")
datasets <- listDatasets(mart)
datasets$dataset
yeast_txdb <- makeTxDbFromBiomart(biomart="fungi_mart",
                                  dataset="scerevisiae_eg_gene",
                                  host="https://fungi.ensembl.org")
yeast_txdb

## ----------------------
## --- Ensembl Metazoa ---

## The metazoa mart is slow and at the same time it doesn't seem to
## support requests that take more than 1 min at the moment. So a call to
## biomaRt::getBM() will fail with a "Timeout was reached" error if the
## requested data takes more than 1 min to download. This unfortunately
## happens with the example below so we don't try to run it for now.


  mart <- useEnsemblGenomes(biomart="metazoa_mart")
  datasets <- listDatasets(mart)
  datasets$dataset
  worm_txdb <- makeTxDbFromBiomart(biomart="metazoa_mart",
                                   dataset="celegans_eg_gene",
                                   host="https://metazoa.ensembl.org")
  worm_txdb
```

```
  ## Note that even if the dataset for Worm on Ensembl Metazoa contains
  ## the same transcript as on the main Ensembl database, the transcript
  ## type might be annotated with slightly different terms (e.g. antisense
  ## vs antisense_RNA):
  filter <- list(tx_name="Y71G12B.44")
  transcripts(worm_txdb, filter=filter, columns=c("tx_name", "tx_type"))
  transcripts(txdb1, filter=filter, columns=c("tx_name", "tx_type"))

## ----------------------
## --- Ensembl Plants ---

## Like the metazoa mart (see above), the plants mart is also slow and
## doesn't seem to support requests that take more than 1 min either.
## So we don't try to run the example below for now.


mart <- useEnsemblGenomes(biomart="plants_mart")
datasets <- listDatasets(mart)
datasets[ , 1:2]
athaliana_txdb <- makeTxDbFromBiomart(biomart="plants_mart",
                                      dataset="athaliana_eg_gene",
                                      host="https://plants.ensembl.org")
athaliana_txdb

## ------------------------
## --- Ensembl Protists ---

mart <- useEnsemblGenomes(biomart="protists_mart")
datasets <- listDatasets(mart)
datasets$dataset
tgondii_txdb <- makeTxDbFromBiomart(biomart="protists_mart",
                                    dataset="tgondii_eg_gene",
                                    host="https://protists.ensembl.org")
tgondii_txdb

## ---------------------------------------------------------------------
## C. USING AN Ensembl MIRROR
## ---------------------------------------------------------------------

## You can use the 'host' argument to access the "ENSEMBL_MART_ENSEMBL"
## BioMart database at a mirror (e.g. at uswest.ensembl.org). A gotcha
## when doing this is that the name of the database on the mirror might
## be different! We can check this with listMarts() from the biomaRt
## package:
if (interactive()) {

  listMarts(host="https://useast.ensembl.org")

  txdb3 <- makeTxDbFromBiomart(biomart="ENSEMBL_MART_ENSEMBL",
                               dataset="hsapiens_gene_ensembl",
                               transcript_ids=transcript_ids,
                               host="https://useast.ensembl.org")
  txdb3
```

```
}
## Therefore in addition to setting 'host' to "uswest.ensembl.org", we
## might also need to specify the 'biomart' argument.


## ---------------------------------------------------------------------
## D. USING FILTERS
## ---------------------------------------------------------------------

## We can use listFilters() from the biomaRt package to get valid filter
## names:
mart <- useEnsembl(biomart="ENSEMBL_MART_ENSEMBL",
                dataset="hsapiens_gene_ensembl",
                host="https://www.ensembl.org")
head(listFilters(mart))

## Retrieve transcript dataset for Ensembl gene ENSG00000011198:
my_filter <- list(ensembl_gene_id="ENSG00000011198")

if (interactive()) {
  txdb4 <- makeTxDbFromBiomart(dataset="hsapiens_gene_ensembl",
                               filter=my_filter)
  txdb4
  transcripts(txdb4, columns=c("tx_id", "tx_name", "gene_id"))
  transcriptLengths(txdb4)
}

## ---------------------------------------------------------------------
## E. RETRIEVING CHROMOSOME INFORMATION ONLY
## ---------------------------------------------------------------------

chrominfo <- getChromInfoFromBiomart(dataset="celegans_gene_ensembl")
chrominfo
```

`makeTxDbFromEnsembl` *Make a TxDb object from an Ensembl database*

Description

The `makeTxDbFromEnsembl` function creates a TxDb object for a given organism by importing the genomic locations of its transcripts, exons, CDS, and genes from an Ensembl database.

Note that it uses the **RMariaDB** package internally so make sure that this package is installed.

Usage

```
makeTxDbFromEnsembl(organism="Homo sapiens",
                    release=NA,
                    circ_seqs=NULL,
                    server="ensembldb.ensembl.org",
```

```
                    username="anonymous", password=NULL, port=0L,
                    tx_attrib=NULL,
                    taxonomyId=NA)
```

Arguments

organism	The *scientific name* (i.e. genus and species, or genus and species and subspecies) of the organism for which to import the data. Case is not sensitive. Underscores can be used instead of white spaces e.g. "homo_sapiens" is accepted.
release	The Ensembl release to query e.g. 89. If set to NA (the default), the current release is used.
circ_seqs	A character vector to list out which chromosomes should be marked as circular.
server	The name of the MySQL server to query. See https://www.ensembl.org/info/data/mysql.html for the list of Ensembl public MySQL servers. Make sure to use the server nearest to you. It can make a big difference!
username	Login username for the MySQL server.
password	Login password for the MySQL server.
port	Port of the MySQL server.
tx_attrib	If not NULL, only select transcripts with an attribute of the given code, a string, like "gencode_basic".
taxonomyId	By default this argument is set to NA and the taxonomy ID for the supplied organism will be automatically obtained and stored in the returned TxDb object. However, you can override this by supplying your own (valid) taxonomy ID, even though you should never need to do this.

Value

A TxDb object.

Note

makeTxDbFromEnsembl tends to be faster and more reliable than makeTxDbFromBiomart.

Author(s)

H. Pagès

See Also

- makeTxDbFromUCSC and makeTxDbFromBiomart for making a TxDb object from other online resources.
- makeTxDbFromGRanges and makeTxDbFromGFF for making a TxDb object from a GRanges object, or from a GFF or GTF file.
- TxDb objects implemented in the **GenomicFeatures** package.
- makeTxDb for the low-level function used by the makeTxDbFrom* functions to make the TxDb object returned to the user.

Examples

```
## Note that, right after a new Ensembl release, it can take up to 24 or
## 48 hours for the MySQL server at useastdb.ensembl.org to get updated
## with the new release. During that period, the server will be out-of-sync
## with the content at https://ftp.ensembl.org/pub/current_mysql/, which
## can cause makeTxDbFromEnsembl() to fail with an error like:
##   Error: Failed to connect: Unknown database 'some_organism_core_114_4'
## To avoid running into this issue in the context of this example, we
## first try useastdb.ensembl.org, and, if it fails, we fall back to
## ensembldb.ensembl.org.
txdb <- try(makeTxDbFromEnsembl("Saccharomyces cerevisiae",
                                server="useastdb.ensembl.org"))
if (inherits(txdb, "try-error")) {
    txdb <- try(makeTxDbFromEnsembl("Saccharomyces cerevisiae",
                                    server="ensembldb.ensembl.org"))
}
txdb
```

`makeTxDbFromGFF` *Make a TxDb object from annotations available as a GFF3 or GTF file*

Description

The `makeTxDbFromGFF` function allows the user to make a TxDb object from transcript annotations available as a GFF3 or GTF file.

Usage

```
makeTxDbFromGFF(file,
                format=c("auto", "gff3", "gtf"),
                dataSource=NA,
                organism=NA,
                taxonomyId=NA,
                circ_seqs=NULL,
                chrominfo=NULL,
                metadata=NULL,
                dbxrefTag)
```

Arguments

`file`	Input GFF3 or GTF file. Can be a path to a file, or an URL, or a connection object, or a GFF3File or GTFFile object.
`format`	Format of the input file. Accepted values are: `"auto"` (the default) for auto-detection of the format, `"gff3"`, or `"gtf"`. Use `"gff3"` or `"gtf"` only if auto-detection failed.
`dataSource`	A single string describing the origin of the data file. Please be as specific as possible.

`organism` What is the Genus and species of this organism. Please use proper scientific nomenclature for example: "Homo sapiens" or "Canis familiaris" and not "human" or "my fuzzy buddy". If properly written, this information may be used by the software to help you out later.

`taxonomyId` By default this value is NA and the organism provided will be used to look up the correct value for this. But you can use this argument to override that and supply your own taxonomy id here (which will be separately validated). Since providing a valid taxonomy id will not require us to look up one based on your organism: this is one way that you can loosen the restrictions about what is and isn't a valid value for the organism.

`circ_seqs` A character vector to list out which chromosomes should be marked as circular.

`chrominfo` Data frame containing information about the chromosomes. Will be passed to the internal call to `makeTxDb`. See ?`makeTxDb` for more information. Alternatively, can be a Seqinfo object.

`metadata` A 2-column data frame containing meta information to be included in the TxDb object. See ?`makeTxDb` for more information about the format of `metadata`.

`dbxrefTag` If not missing, the values in the `Dbxref` attribute with the specified tag (like "GeneID") are used for the feature names.

Details

`makeTxDbFromGFF` is a convenience function that feeds data from the parsed file to the `makeTxDbFromGRanges` function.

Value

A TxDb object.

Author(s)

H. Pagès and M. Carlson

See Also

- `makeTxDbFromGRanges`, which `makeTxDbFromGFF` is based on, for making a TxDb object from a GRanges object.
- The `import` function in the **rtracklayer** package (also used by `makeTxDbFromGFF` internally).
- `makeTxDbFromUCSC`, `makeTxDbFromBiomart`, and `makeTxDbFromEnsembl`, for making a TxDb object from online resources.
- TxDb objects implemented in the **GenomicFeatures** package.
- `makeTxDb` for the low-level function used by the `makeTxDbFrom*` functions to make the TxDb object returned to the user.

Examples

```
## TESTING GFF3
gffFile <- system.file("extdata", "GFF3_files", "a.gff3", package="txdbmaker")
txdb <- makeTxDbFromGFF(gffFile,
                        dataSource="partial gtf file for Tomatoes for testing",
                        organism="Solanum lycopersicum")

## TESTING GTF, this time specifying some metadata and the chrominfo
gtfFile <- system.file("extdata", "GTF_files",
                       "GCA_002204515.1_AaegL5.0_genomic.gtf.gz",
                       package="txdbmaker")
resource_url <- paste0("ftp.ncbi.nlm.nih.gov/genomes/all/GCA/002/204/515/",
                       "GCA_002204515.1_AaegL5.0/")
metadata <- data.frame(name="Resource URL", value=resource_url)
chrominfo <- data.frame(chrom="MF194022.1",
                        length=16790,
                        is_circular=TRUE,
                        genome="AaegL5.0")
txdb2 <- makeTxDbFromGFF(gtfFile,
                         dataSource="NCBI",
                         organism="Aedes aegypti",
                         chrominfo=chrominfo,
                         metadata=metadata)
```

`makeTxDbFromGRanges` *Make a TxDb object from a GRanges object*

Description

The `makeTxDbFromGRanges` function allows the user to extract gene, transcript, exon, and CDS information from a GRanges object structured as GFF3 or GTF, and to return that information in a TxDb object.

Usage

```
makeTxDbFromGRanges(gr, drop.stop.codons=FALSE, metadata=NULL, taxonomyId=NA)
```

Arguments

`gr`	A GRanges object structured as GFF3 or GTF, typically obtained with `BiocIO::import()`.
`drop.stop.codons`	`TRUE` or `FALSE`. If `TRUE`, then features of type `stop_codon` are ignored. Otherwise (the default) the stop codons are considered to be part of the CDS and merged to them.
`metadata`	A 2-column data frame containing meta information to be included in the TxDb object. This data frame is just passed to `makeTxDb`, which `makeTxDbFromGRanges` calls at the end to make the TxDb object from the information extracted from `gr`. See `?makeTxDb` for more information about the format of `metadata`.

`taxonomyId` By default this value is NA which will result in an NA field since there is no reliable way to infer this from a GRanges object. But you can use this argument to supply your own valid taxId here and if you do, then the Organism can be filled in as well

Value

A TxDb object.

Author(s)

Hervé Pagès

See Also

- `makeTxDbFromUCSC`, `makeTxDbFromBiomart`, and `makeTxDbFromEnsembl`, for making a TxDb object from online resources.
- `makeTxDbFromGFF` for making a TxDb object from a GFF or GTF file.
- The `import` generic function in the **BiocIO** package.
- The `asGFF` method for TxDb objects (asGFF,TxDb-method) for the reverse of `makeTxDbFromGRanges`, that is, for turning a TxDb object into a GRanges object structured as GFF.
- TxDb objects implemented in the **GenomicFeatures** package.
- `makeTxDb` for the low-level function used by the `makeTxDbFrom*` functions to make the TxDb object returned to the user.

Examples

```
library(BiocIO)  # for the import() function

## ---------------------------------------------------------------------
## WITH A GRanges OBJECT STRUCTURED AS GFF3
## ---------------------------------------------------------------------
GFF3_files <- system.file("extdata", "GFF3_files", package="txdbmaker")

path <- file.path(GFF3_files, "a.gff3")
gr <- import(path)
txdb <- makeTxDbFromGRanges(gr)
txdb

## Reverse operation:
gr2 <- asGFF(txdb)

## Sanity check (asGFF() does not propagate the CDS phase at the moment):
target <- as.list(txdb)
target$splicings$cds_phase <- NULL
stopifnot(identical(target, as.list(makeTxDbFromGRanges(gr2))))

## ---------------------------------------------------------------------
## WITH A GRanges OBJECT STRUCTURED AS GTF
## ---------------------------------------------------------------------
```

```
GTF_files <- system.file("extdata", "GTF_files", package="txdbmaker")

## test1.gtf was grabbed from http://mblab.wustl.edu/GTF22.html (5 exon
## gene with 3 translated exons):
path <- file.path(GTF_files, "test1.gtf")
gr <- import(path)
txdb <- makeTxDbFromGRanges(gr)
txdb

path <- file.path(GTF_files, "GCA_002204515.1_AaegL5.0_genomic.gtf.gz")
gr <- import(path)
txdb <- makeTxDbFromGRanges(gr)
txdb
```

`makeTxDbFromUCSC` *Make a TxDb object from annotations available at the UCSC Genome Browser*

Description

The `makeTxDbFromUCSC` function allows the user to make a TxDb object from transcript annotations available at the UCSC Genome Browser.

Note that it uses the **RMariaDB** package internally so make sure that this package is installed.

Usage

```
makeTxDbFromUCSC(genome="hg19", tablename="knownGene",
        transcript_ids=NULL,
        circ_seqs=NULL,
        goldenPath.url=getOption("UCSC.goldenPath.url"),
        taxonomyId=NA)

supportedUCSCtables(genome="hg19")

browseUCSCtrack(genome="hg19", tablename="knownGene",
                url="https://genome.ucsc.edu/cgi-bin/")
```

Arguments

`genome`	The name of a UCSC genome assembly e.g. `"hg19"` or `"panTro6"`. Get the list of UCSC genomes currently available with `list_UCSC_genomes()[ , "genome"]`.
`tablename`	The name of the UCSC table containing the transcript genomic locations to retrieve. Use the `supportedUCSCtables` utility function to get the list of tables known to work with `makeTxDbFromUCSC`.
`transcript_ids`	Optionally, only retrieve transcript locations for the specified set of transcript ids. If this is used, then the meta information displayed for the resulting TxDb object will say 'Full dataset: no'. Otherwise it will say 'Full dataset: yes'.

`circ_seqs` Like GRanges objects, SummarizedExperiment objects, and many other objects in Bioconductor, the TxDb object returned by `makeTxDbFromUCSC` contains a seqinfo component that can be accessed with `seqinfo()`. This component contains various sequence-level information like the sequence names, lengths, and circularity flag for the genome assembly of the TxDb object.

As far as we know the information of which sequences are circular is not available in the UCSC Genome Browser. However, for the most commonly used UCSC genome assemblies `makeTxDbFromUCSC` will get this information from a knowledge database stored in the **GenomeInfoDb** package (see `?registered_UCSC_genomes`).

For less commonly used UCSC genome assemblies, `makeTxDbFromUCSC` will make a guess based on the chromosome names (e.g. chrM or 2micron will be assumed to be circular). Even though this works most of the time, it is not guaranteed to work *all the time*. So in this case a warning is issued. If you think the guess is incorrect then you can supply your own list of circular sequences (as a character vector) via the `circ_seqs` argument.

`goldenPath.url` A single string specifying the URL to the UCSC goldenPath location where the chromosome sizes are expected to be found.

`url` Use to specify the location of an alternate UCSC Genome Browser.

`taxonomyId` By default this value is NA and the organism inferred will be used to look up the correct value for this. But you can use this argument to supply your own valid taxId here.

Details

`makeTxDbFromUCSC` is a convenience function that feeds data from the UCSC source to the lower level `makeTxDb` function. See `?makeTxDbFromEnsembl` for a similar function that feeds data from an Ensembl database.

Value

For `makeTxDbFromUCSC`: A TxDb object.

For `supportedUCSCtables`: A data frame with 3 columns (`tablename`, `track`, and `subtrack`) and 1 row per table known to work with `makeTxDbFromUCSC`. IMPORTANT NOTE: In the returned data frame, the set of tables associated with a track with subtracks might contain tables that don't exist for the specified genome.

Author(s)

M. Carlson and H. Pagès

See Also

- `makeTxDbFromEnsembl` and `makeTxDbFromBiomart` for making a TxDb object from other online resources.
- `makeTxDbFromGRanges` and `makeTxDbFromGFF` for making a TxDb object from a GRanges object, or from a GFF or GTF file.
- `list_UCSC_genomes` in the **UCSC.utils** package to get the list of UCSC genomes.

- TxDb objects implemented in the **GenomicFeatures** package.
- makeTxDb for the low-level function used by the makeTxDbFrom* functions to make the TxDb object returned to the user.

Examples

```
## Use list_UCSC_genomes() to obtain the list of all UCSC genomes:
library(UCSC.utils)
list_UCSC_genomes()[ , "genome"]

## To search genomes for a particular organism:
list_UCSC_genomes("human")

## Display the list of tables known to work with makeTxDbFromUCSC():
supportedUCSCtables("hg38")
supportedUCSCtables("hg19")

## Open the UCSC track page for a given organism/table:
browseUCSCtrack("hg38", tablename="knownGene")
browseUCSCtrack("hg19", tablename="knownGene")

browseUCSCtrack("hg38", tablename="ncbiRefSeqSelect")
browseUCSCtrack("hg19", tablename="ncbiRefSeqSelect")

browseUCSCtrack("hg19", tablename="pseudoYale60")

browseUCSCtrack("sacCer3", tablename="ensGene")

## Retrieve a full transcript dataset for Yeast from UCSC:
txdb1 <- makeTxDbFromUCSC("sacCer3", tablename="ensGene")
txdb1

## Retrieve an incomplete transcript dataset for Mouse from UCSC (only
## transcripts linked to Entrez Gene ID 22290):
transcript_ids <- c(
    "uc009uzf.1",
    "uc009uzg.1",
    "uc009uzh.1",
    "uc009uzi.1",
    "uc009uzj.1"
)

txdb2 <- makeTxDbFromUCSC("mm10", tablename="knownGene",
                          transcript_ids=transcript_ids)
txdb2
```

---

makeTxDbPackage *Making a TxDb package from annotations available at the UCSC Genome Browser, biomaRt or from another source.*

Description

A TxDb package is an annotation package containing a TxDb object.

The `makeTxDbPackageFromUCSC` function allows the user to make a TxDb package from transcript annotations available at the UCSC Genome Browser.

The `makeTxDbPackageFromBiomart` function allows the user to do the same thing as `makeTxDbPackageFromUCSC` except that the annotations originate from biomaRt.

Finally, the `makeTxDbPackage` function allows the user to make a TxDb package directly from a TxDb object.

Usage

```
makeTxDbPackageFromUCSC(
    version=,
    maintainer,
    author,
    destDir=".",
    license="Artistic-2.0",
    genome="hg19",
    tablename="knownGene",
    transcript_ids=NULL,
    circ_seqs=NULL,
    goldenPath.url=getOption("UCSC.goldenPath.url"),
    taxonomyId=NA)

makeFDbPackageFromUCSC(
    version,
    maintainer,
    author,
    destDir=".",
    license="Artistic-2.0",
    genome="hg19",
    track="tRNAs",
    tablename="tRNAs",
    columns = UCSCFeatureDbTableSchema(genome, track, tablename),
    url="https://genome.ucsc.edu/cgi-bin/",
    goldenPath.url=getOption("UCSC.goldenPath.url"),
    chromCol=NULL,
    chromStartCol=NULL,
    chromEndCol=NULL,
    taxonomyId=NA)

makeTxDbPackageFromBiomart(
    version,
    maintainer,
    author,
    destDir=".",
    license="Artistic-2.0",
```

```
    biomart="ENSEMBL_MART_ENSEMBL",
    dataset="hsapiens_gene_ensembl",
    transcript_ids=NULL,
    circ_seqs=NULL,
    filter=NULL,
    id_prefix="ensembl_",
    host="https://www.ensembl.org",
    taxonomyId=NA)

makeTxDbPackage(txdb,
                version,
                maintainer,
                author,
                destDir=".",
                license="Artistic-2.0",
                pkgname=NULL,
                provider=NULL,
                providerVersion=NULL)

makePackageName(txdb)
```

Arguments

version	What is the version number for this package?
maintainer	Who is the package maintainer? (must include email to be valid). Should be a person object, or something coercible to one, like a string. May be omitted if the author argument is a person containing someone with the maintainer role.
author	Who is the creator of this package? Should be a person object, or something coercible to one, like a character vector of names. The maintainer argument will be merged into this list.
destDir	A path where the package source should be assembled.
license	What is the license (and it's version)
biomart	which BioMart database to use. Get the list of all available BioMart databases with the listMarts function from the biomaRt package. See the details section below for a list of BioMart databases with compatible transcript annotations.
dataset	which dataset from BioMart. For example: "hsapiens_gene_ensembl", "mmusculus_gene_ensembl", "dmelanogaster_gene_ensembl", "celegans_gene_ensembl", etc in the ensembl database. See the examples section below for how to discover which datasets are available in a given BioMart database.
genome	name of a UCSC genome assembly e.g. "hg19" or "panTro6". Get the list of UCSC genomes currently available with list_UCSC_genomes()[, "genome"].
track	name of the UCSC track. Use list_UCSC_tracks from the **UCSC.utils** package to get the list of available tracks and associated primary tables for a particular genome.
tablename	name of the UCSC table containing the transcript annotations to retrieve. Use the supportedUCSCtables utility function to get the list of tables known to work with makeTxDbFromUCSC.

`transcript_ids` optionally, only retrieve transcript annotation data for the specified set of transcript ids. If this is used, then the meta information displayed for the resulting TxDb object will say 'Full dataset: no'. Otherwise it will say 'Full dataset: yes'.

`circ_seqs` a character vector to list out which chromosomes should be marked as circular.

`filter` Additional filters to use in the BioMart query. Must be a named list. An example is `filter=as.list(c(source="entrez"))`

`host` The host URL of the BioMart. Defaults to https://www.ensembl.org.

`id_prefix` Specifies the prefix used in BioMart attributes. For example, some BioMarts may have an attribute specified as `"ensembl_transcript_id"` whereas others have the same attribute specified as `"transcript_id"`. Defaults to `"ensembl_"`.

`columns` a named character vector to list out the names and types of the other columns that the downloaded track should have. Use `UCSCFeatureDbTableSchema` to retrieve this information for a particular table.

`url`, `goldenPath.url`
use to specify the location of an alternate UCSC Genome Browser.

`chromCol` If the schema comes back and the 'chrom' column has been labeled something other than 'chrom', use this argument to indicate what that column has been labeled as so we can properly designate it. This could happen (for example) with the knownGene track tables, which has no 'chromStart' or 'chromEnd' columns, but which DOES have columns that could reasonably substitute for these columns under particular circumstances. Therefore we allow these three columns to have arguments so that their definition can be re-specified

`chromStartCol` Same thing as chromCol, but for renames of 'chromStart'

`chromEndCol` Same thing as chromCol, but for renames of 'chromEnd'

`txdb` A TxDb object that represents a handle to a transcript database. This object type is what is returned by `makeTxDbFromUCSC`, `makeTxDbFromUCSC` or `makeTxDb`

`taxonomyId` By default this value is NA and the organism provided (or inferred) will be used to look up the correct value for this. But you can use this argument to override that and supply your own valid taxId here

`pkgname` By default this value is NULL and does not need to be filled in (a package name will be generated for you). But if you override this value, then the package and it's object will be instead named after this value. Be aware that the standard rules for package names will apply, (so don't include spaces, underscores or dashes)

`provider` If not given, a default is taken from the 'Data source' field of the metadata table.

`providerVersion`
If not given, a default is taken from one of 'UCSC table', 'BioMart version' or 'Data source' fields of the metadata table.

Details

`makeTxDbPackageFromUCSC` is a convenience function that calls both the `makeTxDbFromUCSC` and the `makeTxDbPackage` functions. The `makeTxDbPackageFromBiomart` follows a similar pattern and calls the `makeTxDbFromBiomart` and `makeTxDbPackage` functions.

Value

A TxDb object.

Author(s)

M. Carlson

See Also

makeTxDbFromUCSC, makeTxDbFromBiomart, makeTxDb, list_UCSC_genomes

Examples

```
## First consider relevant helper/discovery functions:
## Get the list of tables known to work with makeTxDbPackageFromUCSC():
supportedUCSCtables(genome="hg19")

## Next are examples of actually building a package:

## Makes a transcript package for Yeast from the ensGene table at UCSC:
makeTxDbPackageFromUCSC(version="0.01",
                        maintainer="Some One <so@someplace.org>",
                        author="Some One <so@someplace.com>",
                        genome="sacCer2",
                        tablename="ensGene")

## Makes a transcript package from Human by using biomaRt and limited to a
## small subset of the transcripts.
transcript_ids <- c(
    "ENST00000400839",
    "ENST00000400840",
    "ENST00000478783",
    "ENST00000435657",
    "ENST00000268655",
    "ENST00000313243",
    "ENST00000341724")

makeTxDbPackageFromBiomart(version="0.01",
                           maintainer="Some One <so@someplace.org>",
                           author="Some One <so@someplace.com>",
                           transcript_ids=transcript_ids)
```

Index

∗ **package**
txdbmaker-package, 2

asGFF,TxDb-method, *18*

browseUCSCtrack *(*makeTxDbFromUCSC*)*, 19

FeatureDb, *3, 4*

getChromInfoFromBiomart
*(*makeTxDbFromBiomart*)*, 8
GFF3File, *15*
GRanges, *7*, *10*, *14*, *16–18*, *20*
GTFFile, *15*

import, *16–18*

list_UCSC_genomes, *3, 4*, *19, 20*, *23*, *25*
list_UCSC_tracks, *3, 4*, *23*
listDatasets, *10*
listFilters, *10*
listMarts, *8*, *10*, *23*
loadDb, *7*

makeFDbPackageFromUCSC
*(*makeTxDbPackage*)*, 21
makeFeatureDbFromUCSC, 3
makePackageName *(*makeTxDbPackage*)*, 21
makeTxDb, 5, *9, 10*, *14*, *16–18*, *20, 21*, *25*
makeTxDbFromBiomart, *7*, 8, *14*, *16*, *18*, *20*, *24, 25*
makeTxDbFromEnsembl, *5*, *7, 8*, *10*, 13, *16*, *18*, *20*
makeTxDbFromGFF, *5*, *7*, *10*, *14*, 15, *18*, *20*
makeTxDbFromGRanges, *7*, *10*, *14*, *16*, 17, *20*
makeTxDbFromUCSC, *5*, *7*, *9, 10*, *14*, *16*, *18*, 19, *24, 25*
makeTxDbPackage, 21, *24*
makeTxDbPackageFromBiomart
*(*makeTxDbPackage*)*, 21
makeTxDbPackageFromUCSC
*(*makeTxDbPackage*)*, 21

person, *23*

registered_UCSC_genomes, *20*

saveDb, *7*
Seqinfo, *16*
seqinfo, *20*
SummarizedExperiment, *20*
supportedUCSCFeatureDbTables
*(*makeFeatureDbFromUCSC*)*, 3
supportedUCSCFeatureDbTracks
*(*makeFeatureDbFromUCSC*)*, 3
supportedUCSCtables, *23*
supportedUCSCtables *(*makeTxDbFromUCSC*)*, 19

TxDb, *2*, *5*, *7–10*, *13–22*, *24, 25*
txdbmaker *(*txdbmaker-package*)*, 2
txdbmaker-package, 2

UCSCFeatureDbTableSchema
*(*makeFeatureDbFromUCSC*)*, 3
useEnsembl, *10*