GenomicFeatures

October 5, 2010

`TranscriptDb-class` *TranscriptDb objects*

Description

The TranscriptDb class is a container for storing transcript annotations.

See `?makeTranscriptDbFromUCSC` and `?makeTranscriptDbFromBiomart` for making a TranscriptDb object from the UCSC or BioMart sources.

See `?saveFeatures` and `?loadFeatures` for saving and loading the database contents of a TranscriptDb object.

Methods

In the code snippets below, `x` is a TranscriptDb object.

`seqnames(x)`: Returns the names of all chromosomes in a character vector. Note that "all" here means at least the chromosomes that have features. But some TranscriptDb objects (in particular those created with `makeTranscriptDbFromUCSC`) are storing the names of all the chromosomes forming the genome.

`seqlengths(x)`: Returns the lengths of the chromosomes *or NAs* in an integer vector which names are `seqnames(x)`. It's important to note that either *all* elements are NAs or *none* is, depending on the avaibality of this information at creation time.

`as.list(x)`: Dumps the entire db into a list of data frames `txdump` that can be used in `do.call(makeTranscriptDb, txdump)` to make the db again with no loss of information. Note that the transcripts are dumped in the same order in all the data frames.

See `?transcripts`, and `?transcriptsByOverlaps` for other useful operations on TranscriptDb objects.

Author(s)

H. Pages

See Also

`makeTranscriptDbFromUCSC`, `makeTranscriptDbFromBiomart`, `loadFeatures`, `transcripts`, `transcriptsByOverlaps`,

Examples

```
txdb_file <- system.file("extdata", "Biomart_Ensembl_sample.sqlite",
                         package="GenomicFeatures")
txdb <- loadFeatures(txdb_file)
txdb

seqnames(txdb)
seqlengths(txdb)

txdump <- as.list(txdb)
txdump
txdb1 <- do.call(makeTranscriptDb, txdump)
stopifnot(identical(as.list(txdb1), txdump))
```

`extractTranscriptsFromGenome`
Extract transcripts from a genome

Description

`extractTranscriptsFromGenome` extracts the transcript sequences from a BSgenome data package using transcript information (exon boundaries) stored in a "gene table".

Usage

```
extractTranscriptsFromGenome(genome, txdb, use.names=TRUE)
```

Arguments

`genome`	A `BSgenome` object. See the `available.genomes` function in the BSgenome package for how to install a genome.
`txdb`	A TranscriptDb object, a GRangesList object, or a data frame like that returned by `geneHuman`.
`use.names`	`TRUE` or `FALSE`. Ignored if `txdb` is not a TranscriptDb object. If `TRUE` (the default), the returned sequences are named with the transcript names. If `FALSE`, they are named with the transcript internal ids. Note that, unlike the transcript internal ids, the transcript names are not guaranteed to be unique or even defined (they could be all `NA`s). A warning is issued when this happens.

Value

A DNAStringSet object.

Note

`extractTranscriptsFromGenome` is based on the `extractTranscripts` function defined in the Biostrings package. See `¿extractTranscripts` for more information and related functions like `transcriptLocs2refLocs` for converting transcript-based locations into chromosome-based (aka reference-based) locations.

Author(s)

H. Pages

See Also

available.genomes, geneHuman, transcriptLocs2refLocs

Examples

```
library(BSgenome.Hsapiens.UCSC.hg18)  # load the genome

## ---------------------------------------------------------------------
## A. USING A TranscriptDb OBJECT
## ---------------------------------------------------------------------
txdb_file <- system.file("extdata", "UCSC_knownGene_sample.sqlite",
                         package="GenomicFeatures")
txdb <- loadFeatures(txdb_file)
transcripts <- extractTranscriptsFromGenome(Hsapiens, txdb)
transcripts

## ---------------------------------------------------------------------
## B. USING A GRangesList OBJECT
## ---------------------------------------------------------------------
## Exons grouped by transcripts (gives the same result as above except
## that now transcripts are named by their internal id i.e. by tx_id
## instead of tx_name):
extractTranscriptsFromGenome(Hsapiens, exonsBy(txdb))
## CDSs grouped by transcripts (this extracts only the translated parts
## of the transcripts):
cds <- extractTranscriptsFromGenome(Hsapiens, cdsBy(txdb))

## ---------------------------------------------------------------------
## C. USING A UCSC-LIKE DATA FRAME
## ---------------------------------------------------------------------
## IMPORTANT NOTE: This is provided for compatibility with the old
## GenomicFeatures.*.UCSC.* packages and might be removed at any time.
library(GenomicFeatures.Hsapiens.UCSC.hg18)  # load the gene table
genes <- geneHuman()
library(Biostrings)  # for transcriptWidths()
tw <- transcriptWidths(genes$exonStarts, genes$exonEnds)

if (interactive()) {
  ## Takes about 30 sec.:
  transcripts <- extractTranscriptsFromGenome(Hsapiens, genes)
  ## Sanity check:
  stopifnot(identical(width(transcripts), tw))
}

## Get the reference-based locations of the first 4 (5' end)
## and last 4 (3' end) nucleotides in each transcript:
tlocs <- lapply(tw, function(w) c(1:4, (w-3):w))
rlocs <- transcriptLocs2refLocs(tlocs, genes$exonStarts, genes$exonEnds,
             genes$strand, reorder.exons.on.minus.strand=TRUE)
```

makeTranscriptDb *Making a TranscriptDb object from user supplied annotations*

Description

makeTranscriptDb is a low-level constructor for making a TranscriptDb object from user supplied transcript annotations. See ?makeTranscriptDbFromUCSC and ?makeTranscriptDbFromBiomart for higher-level functions that feed data from the UCSC or BioMart sources to makeTranscriptDb.

Usage

```
makeTranscriptDb(transcripts, splicings,
                 genes=NULL, chrominfo=NULL, metadata=NULL, ...)
```

Arguments

transcripts	data frame containing the genomic locations of a set of transcripts
splicings	data frame containing the exon and cds locations of a set of transcripts
genes	data frame containing the genes associated to a set of transcripts
chrominfo	data frame containing the chromosome information of a set of transcripts
metadata	2-column data frame containing meta information about this set of transcripts like species, organism, genome, UCSC table, etc... The names of the columns must be "name" and "value" and their type must be character.
...	ignored for now

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 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_name: [optional] Transcript name. Character vector (or factor).
- tx_chrom: Transcript chromosome. Character vector (or factor) with no NAs.
- tx_strand: Transcript strand. Character vector (or factor) where each element is either "+" or "-".
- tx_start, tx_end: Transcript start and end. Integer vectors with no NAs.

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 eventually the cds contained in 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).
- exon_chrom: [optional] Exon chromosome. Character vector (or factor) with no NAs. If missing then transcripts$tx_chrom is used. If present then exon_strand must be present too.
- exon_strand: [optional] Exon strand. Character vector (or factor) with no NAs. If missing then transcripts$tx_strand is used and exon_chrom must be missing too.
- exon_start, exon_end: Exon start and end. Integer vectors with no NAs.
- cds_id: [optional] cds ID. Integer vector. If present then cds_start and cds_end must be too. NAs are allowed and must match NAs in cds_start and cds_end.
- cds_name: [optional] cds name. Character vector (or factor). If present then cds_start and cds_end must be too. NAs are allowed and must match NAs in cds_start and cds_end.
- cds_start, cds_end: [optional] cds start and end. 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.

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

genes 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.
- length: Chromosome length. Either all NAs or an integer vector with no NAs.

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

Value

A TranscriptDb object.

Author(s)

H. Pages

See Also

TranscriptDb, makeTranscriptDbFromUCSC, makeTranscriptDbFromBiomart

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))

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

`makeTranscriptDbFromBiomart` *Making a TranscriptDb object from annotations available on a BioMart database*

Description

The `makeTranscriptDbFromBiomart` function allows the user to make a TranscriptDb object from transcript annotations available on a BioMart database.

Usage

```
makeTranscriptDbFromBiomart(biomart="ensembl",
                            dataset="hsapiens_gene_ensembl",
                            transcript_ids=NULL)
```

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"`, `"scerevisiae_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 TranscriptDb object will say 'Full dataset: no'. Otherwise it will say 'Full dataset: yes'.

Details

makeTranscriptDbFromBiomart is a convenience function that feeds data from a BioMart database to the lower level makeTranscriptDb function. See ?makeTranscriptDbFromUCSC for a similar function that feeds data from the UCSC source.

As of June 28, 2010, the BioMart databases with compatible transcript annotations are:

- ensembl: ENSEMBL GENES 58 (SANGER UK)
- bacterial_mart_5: ENSEMBL BACTERIA 5 (EBI UK)
- fungal_mart_5: ENSEMBL FUNGAL 5 (EBI UK)
- metazoa_mart_5: ENSEMBL METAZOA 5 (EBI UK)
- plant_mart_5: ENSEMBL PLANT 5 (EBI UK)
- protist_mart_5: ENSEMBL PROTISTS 5 (EBI UK)
- ensembl_expressionmart_48: EURATMART (EBI UK)
- Ensembl56: PANCREATIC EXPRESSION DATABASE (INSTITUTE OF CANCER UK)
- ENSEMBL_MART_ENSEMBL: GRAMENE 30 ENSEMBL GENES (CSHL/CORNELL US)

Only ensembl and Ensembl56 have CDS information.

Value

A TranscriptDb object.

Author(s)

M. Carlson and H. Pages

See Also

listMarts, useMart, listDatasets, makeTranscriptDbFromUCSC, makeTranscriptDb

Examples

```
## Discover which datasets are available in the "ensembl" and
## "plant_mart_5" BioMart databases:
library(biomaRt)
listDatasets(useMart("ensembl"))
listDatasets(useMart("plant_mart_5"))

## Retrieving an incomplete transcript dataset for Human from the
## "ensembl" BioMart database:
transcript_ids <- c(
    "ENST00000400839",
    "ENST00000400840",
    "ENST00000478783",
    "ENST00000435657",
    "ENST00000268655",
    "ENST00000313243",
    "ENST00000341724"
)
txdb <- makeTranscriptDbFromBiomart(transcript_ids=transcript_ids)
txdb  # note that these annotations match the GRCh37 genome assembly
```

makeTranscriptDbFromUCSC *Making a TranscriptDb object from annotations available at the UCSC Genome Browser*

Description

The makeTranscriptDbFromUCSC function allows the user to make a TranscriptDb object from transcript annotations available at the UCSC Genome Browser.

Usage

```
supportedUCSCtables()

makeTranscriptDbFromUCSC(
        genome="hg18",
        tablename="knownGene",
        transcript_ids=NULL,
        url="http://genome.ucsc.edu/cgi-bin/",
        goldenPath_url="http://hgdownload.cse.ucsc.edu/goldenPath")
```

Arguments

genome	genome abbreviation used by UCSC and obtained by ucscGenomes()[, "db"]. For example: "hg18".
tablename	name of the UCSC table containing the transcript annotations to retrieve. Use the supportedUCSCtables utility function to get the list of supported tables. Note that not all tables are available for all genomes.
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 TranscriptDb object will say 'Full dataset: no'. Otherwise it will say 'Full dataset: yes'.
url,goldenPath_url	use to specify the location of an alternate UCSC Genome Browser.

Details

makeTranscriptDbFromUCSC is a convenience function that feeds data from the UCSC source to the lower level makeTranscriptDb function. See ?makeTranscriptDbFromBiomart for a similar function that feeds data from a BioMart database.

Value

A TranscriptDb object.

Author(s)

M. Carlson and H. Pages

See Also

ucscGenomes, makeTranscriptDbFromBiomart, makeTranscriptDb

Examples

```
## Display the list of genomes available at UCSC:
library(rtracklayer)
ucscGenomes()[ , "db"]

## Display the list of tables supported by makeTranscriptDbFromUCSC():
supportedUCSCtables()

## Retrieving a full transcript dataset for Yeast from UCSC:
txdb1 <- makeTranscriptDbFromUCSC(genome="sacCer2", tablename="ensGene")
txdb1

## Retrieving 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 <- makeTranscriptDbFromUCSC(genome="mm9", tablename="knownGene",
                                  transcript_ids=transcript_ids)
txdb2
```

---

regions *Functions that compute genomic regions of interest.*

Description

Functions that compute genomic regions of interest such as promotor, upstream regions etc, from the genomic locations provided in data like the data.frame returned by geneHuman.

Usage

```
transcripts_deprecated(genes, proximal = 500, distal = 10000)
exons_deprecated(genes)
introns_deprecated(genes)
```

Arguments

genes	A data.frame like that returned by geneHuman.
proximal	The number of bases on either side of TSS and 3'-end for the promoter and end region, respectively.
distal	The number of bases on either side for upstream/downstream, i.e. enhancer/silencer regions.

Details

The assumption made for introns is that there must be more than one exon, and that the introns are between the end of one exon and before the start of the next exon.

Value

All of these functions return a RangedData object with a gene column with the UCSC ID of the gene. For transcripts_deprecated, each element corresponds to a transcript, and there are columns for each type of region (promoter, threeprime, upstream, and downstream). For exons_deprecated, each element corresponds to an exon. For introns_deprecated, each element corresponds to an intron.

Author(s)

M. Lawrence.

Examples

```
library(GenomicFeatures.Hsapiens.UCSC.hg18)
## promoter 300bp up and down from TSS (threeprime from TES)
transcripts_deprecated(geneHuman(), proximal = 300)
```

---

saveFeatures *Methods to save and load the database contents for a Transcript Object.*

Description

These methods provide a way to dump a TranscriptDb object to an SQLite file, and to recreate that object the saved file.

Usage

```
saveFeatures(x, file)
loadFeatures(file)
```

Arguments

x	a transcripts object, which contains a connection to a DB.
file	A SQLite Database filename.

Value

For loadFeatures only, a TranscriptDb object is returned.

Author(s)

M. Carlson

See Also

TranscriptDb

Examples

```
## Not run:
  ##Have a TranscriptDb object called txdb?
  ##You can save it:
  saveFeatures(txdb, file="HG18.sqlite")
  ##Then later you can quickly re-constitute it like this:
  txdb <- loadFeatures("HG18.sqlite")

## End(Not run)
```

`transcripts` *Retrieving genomic features from a TranscriptDb object.*

Description

Functions to retrieve genomic features from a TranscriptDb object.

Usage

```
transcripts(txdb, vals=NULL, columns=c("tx_id", "tx_name"))
exons(txdb, vals=NULL)
cds(txdb, vals=NULL)
```

Arguments

`txdb`	A TranscriptDb object.
`vals`	Either `NULL` or a named list of vectors to be used to restrict the output. For `transcripts` the element names can be `"gene_id"`, `"tx_id"`, `"tx_name"`, `"tx_chrom"`, and `"tx_strand"`. For `exons` the element names can be `"exon_id"`, `"exon_chrom"`, and `"exon_strand"`. For `cds` the element names can be `"cds_id"`, `"cds_chrom"`, and `"cds_strand"`.
`columns`	columns to include in the output. Can be `NULL` or a combination of `"tx_id"`, `"tx_name"`, `"gene_id"`, `"exon_id"`, and `"cds_id"`.

Details

These are the main functions for retrieving transcript information from a TranscriptDb object. They can restrict the output based on categorical information. To restrict the output based on interval information, use the `transcriptsByOverlaps`, `exonsByOverlaps`, and `cdsByOverlaps` functions.

Value

a GRanges object

Author(s)

M. Carlson and P. Aboyoun

See Also

TranscriptDb, `transcriptsByOverlaps`

Examples

```
txdb <- loadFeatures(system.file("extdata", "UCSC_knownGene_sample.sqlite",
                                 package="GenomicFeatures"))
vals <- list(tx_chrom = c("chr3", "chr5"), tx_strand = "+")
transcripts(txdb, vals)
```

---

transcriptsBy *Extract and group genomic features of a given type*

Description

These functions extract genomic features of a given type from a TranscriptDb object and group them according to a user-specified criteria.

Usage

```
transcriptsBy(txdb, by=c("gene", "exon", "cds"), use.names=FALSE)
exonsBy(txdb, by=c("tx", "gene"), use.names=FALSE)
cdsBy(txdb, by=c("tx", "gene"), use.names=FALSE)
intronsByTranscript(txdb, use.names=FALSE)
fiveUTRsByTranscript(txdb, use.names=FALSE)
threeUTRsByTranscript(txdb, use.names=FALSE)
```

Arguments

txdb	A TranscriptDb object.
by	One of "gene", "exon", "cds" or "tx". Determines the grouping.
use.names	Controls how to set the names of the returned GRangesList object. These functions return all the features of a given type (e.g. all the exons) grouped by another feature type (e.g. grouped by transcript) in a GRangesList object. By default (i.e. if use.names is FALSE), the names of this GRangesList object (aka the group names) are the internal ids of the features used for grouping (aka the grouping features), which are guaranteed to be unique. If use.names is TRUE, then the names of the grouping features are used instead of their internal ids. For example, when grouping by transcript (by="tx"), the default group names are the transcript internal ids ("tx_id"). But, if use.names=TRUE, the group names are the transcript names ("tx_name"). Note that, unlike the feature ids, the feature names are not guaranteed to be unique or even defined (they could be all NAs). A warning is issued when this happens. Finally, use.names=TRUE cannot be used when grouping by gene by="gene". This is because, unlike for the other features, the gene ids are external ids (e.g. Entrez Gene or Ensembl ids) so the db doesn't have a "gene_name" column for storing alternate gene names.

Details

These functions return a GRangesList object where the ranges within each of the elements are ordered according to the following rule:

When using exonsBy and cdsBy with by = "tx", the ranges are returned in the order they appear in the transcript, i.e. order by the splicing.exon_rank field in txdb's internal database. In all other cases, the ranges will be ordered by chromosome, strand, start, and end values.

Value

A GRangesList object.

Author(s)

M. Carlson, P. Aboyoun and H. Pages

See Also

TranscriptDb, transcripts, transcriptsByOverlaps

Examples

```
txdb_file <- system.file("extdata", "UCSC_knownGene_sample.sqlite",
                         package="GenomicFeatures")
txdb <- loadFeatures(txdb_file)

## Get the transcripts grouped by gene
transcriptsBy(txdb, "gene")

## Get the exons grouped by gene
exonsBy(txdb, "gene")

## Get the cds grouped by transcript
cdsBy(txdb, "tx")
cdsBy(txdb, "tx", use.names=TRUE)  # more informative group names

## Get the introns grouped by transcript
intronsByTranscript(txdb)

## Get the 5' UTRs grouped by transcript
fiveUTRsByTranscript(txdb)
fiveUTRsByTranscript(txdb, use.names=TRUE)  # more informative group names
```

---

transcriptsByOverlaps

Retrieving genomic features from a TranscriptDb object using a restriction by genomic location.

Description

Functions to retrieve genomic features for specified genomic locations.

Usage

```
transcriptsByOverlaps(txdb, ranges, maxgap = 0L, minoverlap = 1L,
                      type = c("any", "start", "end"),
                      columns = c("tx_id", "tx_name"))
exonsByOverlaps(txdb, ranges, maxgap = 0L, minoverlap = 1L,
                type = c("any", "start", "end"))
cdsByOverlaps(txdb, ranges, maxgap = 0L, minoverlap = 1L,
              type = c("any", "start", "end"))
```

Arguments

txdb	A TranscriptDb object.
ranges	A GRanges object to restrict the output.
type	How to perform the interval overlap operations of the ranges. See the findOverlaps manual page in the GRanges package for more information.
maxgap	A non-negative integer representing the maximum distance between a query interval and a subject interval.
minoverlap	Ignored.
columns	columns to include in the output. Can be NULL or a combination of "tx_id", "tx_name", "gene_id", "exon_id", and "cds_id".

Details

These functions subset the results of transcripts, exons, and cds function calls with using the results of findOverlaps calls based on the specified ranges.

Value

a GRanges object

Author(s)

P. Aboyoun

See Also

TranscriptDb, transcripts

Examples

```
txdb <- loadFeatures(system.file("extdata", "UCSC_knownGene_sample.sqlite",
                                 package="GenomicFeatures"))
gr <- GRanges(seqnames = rep("chr1",2),
              ranges = IRanges(start=c(500,10500), end=c(10000,30000)),
              strand = strand(rep("-",2)))
transcriptsByOverlaps(txdb, gr)
```

Index