Package 'sitadela'

November 12, 2026

Type Package

Title An R package for the easy provision of simple but complete tab-delimited genomic annotation from a variety of sources and organisms

Depends R (>= 4.1.0)

Imports Biobase, BiocGenerics, biomaRt, Biostrings, Seqinfo, GenomicFeatures, GenomicRanges, IRanges, methods, parallel, Rsamtools, RSQLite, rtracklayer, S4Vectors, tools, txdbmaker, utils

Suggests GenomeInfoDb, BiocStyle, BSgenome, knitr, rmarkdown, RMySQL, RUnit

Description Provides an interface to build a unified database of genomic annotations and their coordinates (gene, transcript and exon levels). It is aimed to be used when simple tab-delimited annotations (or simple GRanges objects) are required instead of the more complex annotation Bioconductor packages. Also useful when combinatorial annotation elements are reuired, such as RefSeq coordinates with Ensembl biotypes. Finally, it can download, construct and handle annotations with versioned genes and transcripts (where available, e.g. RefSeq and latest Ensembl). This is particularly useful in precision medicine applications where the latter must be reported.

License Artistic-2.0

Encoding UTF-8

LazyData false

URL https://github.com/pmoulos/sitadela

biocViews Software, WorkflowStep, RNASeq, Transcription, Sequencing, Transcriptomics, BiomedicalInformatics, FunctionalGenomics, SystemsBiology, AlternativeSplicing, DataImport, ChIPSeq

VignetteBuilder knitr

BugReports https://github.com/pmoulos/sitadela/issues

Version 1.20.1

Date 2026-03-26

Collate 'argcheck.R' 'fromtxdb.R' 'query-ensembl.R' 'query-ncbi.R' 'query-refseq.R' 'query-ucsc.R' 'query-utils.R' 'reduceops.R' 'sitadela.R' 'testfuns.R' 'zzz.R'

git_url https://git.bioconductor.org/packages/sitadela

git_branch RELEASE_3_23

git_last_commit edf9027

git_last_commit_date 2026-10-26

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Panagiotis Moulos [aut, cre]

Maintainer Panagiotis Moulos <moulos@fleming.gr>

Contents

addAnnotation 2
addCustomAnnotation 5
getAnnotation 7
getInstalledAnnotations 9
getSeqInfo 9
getsetDbPath 10
getUserAnnotations 11
importCustomAnnotation 12
loadAnnotation 13
removeAnnotation 15
testFuns 16

Index **19**

`addAnnotation` *Build a local genomic regions annotation database*

Description

This function is the main annotation database creator of sitadela. It creates a local SQLite database for various organisms and categories of genomic regions. Annotations are retrieved in simple, tab-delimited or GRanges formats.

Usage

```
addAnnotation(organisms, sources, db = getDbPath(),
    versioned = FALSE, forceDownload = TRUE, retries = 5,
    rc = NULL, stopIfNotBS = FALSE)
```

Arguments

organisms	a list of organisms and versions for which to download and build annotations. See also Details.
sources	a character vector of public sources from which to download and build annotations. It can be one or more of "ensembl", "ucsc", "refseq" or "ncbi". See also Details.
db	a valid path (accessible at least by the current user) where the annotation database will be set up. It defaults to system.file(package = "sitadela"), "annotation.sqlite") that is, the installation path of sitadela package.
versioned	create an annotation database with versioned genes and transcripts, when possible.
forceDownload	by default, addAnnotation will not download an existing annotation again (FALSE). Set to TRUE if you wish to update the annotation database for a particular version.
retries	how many times should the annotation worker try to re-connect to internet resources in case of a connection problem or failure.
rc	fraction (0-1) of cores to use in a multicore system. It defaults to NULL (no parallelization). Sometimes used for building certain annotation types.
stopIfNotBS	stop or warn (default) if certain BSgenome packages are not present. See also Details.

Details

Regarding the organisms argument, it is a list with specific format which instructs addAnnotation on which organisms and versions to download from the respective sources. Such a list may have the format: organisms=list(hg19=75, mm9=67, mm10=96:97) This is explained as follows:

- A database comprising the human genome versions hg19 and the mouse genome versions mm9, mm10 will be constructed.
- If "ensembl" is in sources, version 75 is downloaded for hg19 and versions 67, 96, 97 for mm9, mm10.
- If "ucsc" or "refseq" are in sources, the latest versions are downloaded and marked by the download date. As UCSC and RefSeq versions are not accessible in the same way as Ensembl, this procedure cannot always be replicated.

organisms can also be a character vector with organism names/versions (e.g. organisms = c("mm10","hg19")), then the latest versions are downloaded in the case of Ensembl.

The supported supported organsisms are, for human genomes "hg18", "hg19" or "hg38", for mouse genomes "mm9", "mm10", for rat genomes "rn5" or "rn6", for drosophila genome "dm3" or "dm6", for zebrafish genome "danrer7", "danrer10" or "danrer11", for chimpanzee genome "pantro4", "pantro5", for pig genome "susscr3", "susscr11", for Arabidopsis thaliana genome "tair10" and for Equus caballus genome "equcab2" and "equcab3". Finally, it can be "USER_NAMED_ORG" with a custom organism which has been imported to the annotation database by the user using a GTF/GFF file. For example org="mm10_p1".

Regarding sources, "ucsc" corresponds to UCSC Genome Browser annotated transcripts, "refseq" corresponds to UCSC RefSeq maintained transcripts while "ncbi" corresponds to NCBI RefSeq

annotated and maintained transcripts. UCSC, RefSeq and NCBI annotations are constructed by querying the UCSC Genome Browser database.

Regarding `stopIfNotBS`, when `sources` includes `"ucsc"`, `"refseq"` or `"ncbi"`, the GC content of a gene is not available as a database attribute as with Ensembl and has to be calculated if to be included in the respective annotation. For this reason, sitadela uses 'BSgenome' packages. If `stopIfNotBS=FALSE` (default), then the annotation building continues and GC content is `NA` for the missing 'BSgenome' packages.If `stopIfNotBS=FALSE`, then building stops until all the required packages for the selected organisms become available (installed by the user).

Value

The function does not return anything. Only the SQLite database is created or updated.

Author(s)

Panagiotis Moulos

Examples

```
# Build a test database with one genome
myDb <- file.path(tempdir(),"testann.sqlite")

organisms <- list(mm39=115)
sources <- "ensembl"

# If the example is not running in a multicore system, rc is ignored
#addAnnotation(organisms,sources,db=myDb,rc=0.5)

# A more complete case, don't run as example
# Since we are using Ensembl, we can also ask for a version
#organisms <- list(
#    mm9=67,
#    mm10=96:97,
#    hg19=75,
#    hg38=96:97
#)
#sources <- c("ensembl", "refseq")

## Build on the default location (depending on package location, it may
## require root/sudo)
#addAnnotation(organisms,sources)

## Build on an alternative location
#myDb <- file.path(path.expand("~"),"my_ann.sqlite")
#addAnnotation(organisms,sources,db=myDb)
```

addCustomAnnotation *Import custom annotation to existing or new sitadela annotation database from GTF file*

Description

This function imports a GTF file with some custom annotation to a sitadela annotation database.

Usage

```
addCustomAnnotation(gtfFile, metadata, db = getDbPath(),
    rewrite=TRUE)
```

Arguments

gtfFile	a GTF file containing the gene structure of the organism to be imported.
metadata	a list with additional information about the annotation to be imported. See Details.
db	a valid path (accessible at least by the current user) where the annotation database will be set up. It defaults to `system.file(package = "sitadela"), "annotation.sqlite")` that is, the installation path of sitadela package. See also Details.
rewrite	if custom annotation found, rewrite? (default FALSE). Set to TRUE if you wish to update the annotation database for a particular custom annotation.

Details

Regarding the metadata argument, it is a list with specific format which instructs addCustomAnnotation on importing the custom annotation. Such a list may has the following members:

- organism a name of the organism which is imported (e.g. "my_mm9"). This is the only mandatory member.
- source a name of the source for this custom annotation (e.g. "my_mouse_db"). If not given or NULL, the word "inhouse" is used.
- version a string denoting the version. If not given or NULL, current date is used.
- chromInfo it can be one of the following:
  - a tab-delimited file with two columns, the first being the chromosome/sequence names and the second being the chromosome/sequence lengths.
  - a BAM file to read the header from and obtain the required information
  - a data.frame with one column with chromosome lengths and chromosome names as rownames.

See the examples below for a metadata example.

Regarding db, this controls the location of the installation database. If the default is used, then there is no need to provide the local database path to any function that uses the database. Otherwise, the user will either have to provide this each time, or the annotation will have to be downloaded and used on-the-fly.

Value

The function does not return anything. Only the SQLite database is created or updated.

Author(s)

Panagiotis Moulos

Examples

```
# Dummy database as example
customDir <- file.path(tempdir(),"test_custom")
dir.create(customDir)

myDb <- file.path(customDir,"testann.sqlite")
chromInfo <- data.frame(length=c(1000L,2000L,1500L),
    row.names=c("A","B","C"))

# Build with the metadata list filled (you can also provide a version)
if (.Platform$OS.type == "unix") {
    addCustomAnnotation(
        gtfFile=file.path(system.file(package="sitadela"),
            "dummy.gtf.gz"),
        metadata=list(
            organism="dummy",
            source="dummy_db",
            version=1,
            chromInfo=chromInfo
        ),
        db=myDb
    )

    # Try to retrieve some data
    myGenes <- loadAnnotation(genome="dummy",refdb="dummy_db",
        type="gene",db=myDb)
    myGenes
}

## Real data!
## Setup a temporary directory to download files etc.
#customDir <- file.path(tempdir(),"test_custom")
#dir.create(customDir)

#myDb <- file.path(customDir,"testann.sqlite")

## Gene annotation dump from Ensembl
#download.file(paste0("ftp://ftp.ensembl.org/pub/release-98/gtf/",
#  "dasypus_novemcinctus/Dasypus_novemcinctus.Dasnov3.0.98.gtf.gz"),
#  file.path(customDir,"Dasypus_novemcinctus.Dasnov3.0.98.gtf.gz"))

## Chromosome information will be provided from the following BAM file
## available from Ensembl
#bamForInfo <- paste0("ftp://ftp.ensembl.org/pub/release-98/bamcov/",
```

```
#  "dasypus_novemcinctus/genebuild/Dasnov3.broad.Ascending_Colon_5.1.bam")

## Build with the metadata list filled (you can also provide a version)
#addCustomAnnotation(
#  gtfFile=file.path(customDir,"Dasypus_novemcinctus.Dasnov3.0.98.gtf.gz"),
#  metadata=list(
#    organism="dasNov3_test",
#    source="ensembl_test",
#    chromInfo=bamForInfo
#  ),
#  db=myDb
#)

## Try to retrieve some data
#dasGenes <- loadAnnotation(genome="dasNov3_test",refdb="ensembl_test",
#  type="gene",db=myDb)
#dasGenes
```

`getAnnotation` *Annotation downloader*

Description

For Ensembl based annotations, this function connects to the EBI's Biomart service using the package biomaRt and downloads annotation elements (gene co-ordinates, exon co-ordinates, gene identifications, biotypes etc.) for each of the supported organisms. For UCSC/RefSeq annotations, it connects to the respective UCSC SQL databases if the package `RMySQL` is present, otherwise it downloads flat files and build a temporary SQLite database to make the necessary build queries. Gene and transcript versions can be attached (when available) using the `tv` argument. This is very useful when transcript versioning is required, such as several precision medicine applications.

Usage

```
getAnnotation(org, type, refdb = "ensembl", ver = NULL,
    tv = FALSE, rc = NULL)
```

Arguments

`org`	the organism for which to download annotation (one of the supported ones, see Details).
`type`	the transcriptional unit annotation level to load. It can be one of `"gene"` (default), `"transcript"`, `"utr"`, `"transexon"`, `"transutr"`, `"exon"`. See Details for further explanation of each option.
`refdb`	the online source to use to fetch annotation. It can be `"ensembl"` (default), `"ucsc"`, `"refseq"` or `"ncbi"`. In the later three cases, an SQL connection is opened with the UCSC public databases.
`ver`	the version of the annotation to use.
`tv`	attach or not gene/transcript version to gene/transcript name. Defaults to `FALSE`.
`rc`	Fraction of cores to use. Same as the `rc` in `addAnnotation`.

Details

Regarding org, it can be, for human genomes "hg18", "hg19" or "hg38", for mouse genomes "mm9", "mm10", for rat genomes "rn5" or "rn6", for drosophila genome "dm3" or "dm6", for zebrafish genome "danrer7", "danrer10" or "danrer11", for chimpanzee genome "pantro4", "pantro5", for pig genome "susscr3", "susscr11", for Arabidopsis thaliana genome "tair10" and for Equus caballus genome "equcab2" and "equcab3". Finally, it can be "USER_NAMED_ORG" with a custom organism which has been imported to the annotation database by the user using a GTF/GFF file. For example org="mm10_p1".

Regarding type, it defines the level of transcriptional unit (gene, transcript, 3' UTR, exon) coordinates to be loaded or fetched if not present. The following types are supported:

- "gene": canonical gene coordinates are retrieved from the chosen database.
- "transcript": all transcript coordinates are retrieved from the chosen database.
- "utr": all 3' UTR coordinates are retrieved from the chosen database, grouped per gene.
- "transutr": all 3' UTR coordinates are retrieved from the chosen database, grouped per \ transcript.
- "transexon": all exon coordinates are retrieved from the chosen database, grouped per transcript.
- "exon": all exon coordinates are retrieved from the chosen database.

Value

A data frame with the canonical genes, transcripts, exons or 3' UTRs of the requested organism. When type="genes", the data frame has the following columns: chromosome, start, end, gene_id, gc_content, strand, gene_name, biotype. When type="exon" and type="transexon" the data frame has the following columns: chromosome, start, end, exon_id, gene_id, strand, gene_name, biotype. When type="utr" or type="transutr", the data frame has the following columns: chromosome, start, end, transcript_id, gene_id, strand, gene_name, biotype. The latter applies to when type="transcript". The gene_id and exon_id correspond to type="transcript" Ensembl, UCSC or RefSeq gene, transcript and exon accessions respectively. The gene_name corresponds to HUGO nomenclature gene names.

Note

The data frame that is returned contains only "canonical" chromosomes for each organism. It does not contain haplotypes or non-anchored sequences and does not contain mitochondrial chromosomes.

Author(s)

Panagiotis Moulos

Examples

```
mm39Genes <- getAnnotation("mm39","gene")
```

`getInstalledAnnotations` *List installed sitadela annotations*

Description

This function returns a data frame with information on locally installed, supported or custom, annotations.

Usage

```
getInstalledAnnotations(obj = NULL)
```

Arguments

`obj`	`NULL` or the path to a sitadela SQLite annotation database. If `NULL`, the function will try to guess the location of the SQLite database.

Value

The function returns a `data.frame` object with the installed local annotations.

Author(s)

Panagiotis Moulos

Examples

```
db <- file.path(system.file(package="sitadela"),
    "annotation.sqlite")
if (file.exists(db))
    ig <- getInstalledAnnotations(obj=db)
```

`getSeqInfo` *Retrieve sequence length and other information*

Description

This function retrieves sequence (chromosome) length and other information for a set of reference sequences for a sitadela supported organism. If the organism is supported by the `getChromInfoFromUCSC` function of the **GenomeInfoDb** package, then this function is used, otherwise, a directo download from the UCSC golden path takes place to retrieve the required data.

Usage

```
getSeqInfo(org, asSeqinfo = FALSE)
```

Arguments

org	a supported organism to retrieve sequence (aka chromosome) information for. See also addAnnotation about supported organisms.
asSeqinfo	return a Seqinfo object or a data.frame.

Value

The function returns a Seqinfo or a data.frame with the a subset of a Seqinfo information. See also Seqinfo.

Author(s)

Panagiotis Moulos

Examples

```
require(GenomeInfoDb)
s <- getSeqInfo("mm10")
```

---

getsetDbPath *Get and set sitadela default database path*

Description

The setDbPath and getDbPath functions are used to set and get the path to a sitadela annotation database. If not explicitly provided, it defaults to file.path(system.file(package="sitadela"),"annotation.sqlite' Essentially, the setter function adds an option to the R environment pointing to the desired path.

Usage

```
setDbPath(db = NULL)
getDbPath()
```

Arguments

db	path to a valid SQLite database file.

Value

This function does not have a return value.

Author(s)

Panagiotis Moulos

Examples

```
myPath <- "/home/me/test.sqlite"
setDbPath(myPath)
getDbPath()
```

`getUserAnnotations` *List installed custom user-defined sitadela annotations*

Description

This function returns a data frame with information on locally installed, custom user-defined annotations only. For a list of all annotations, see `getInstalledAnnotations`.

Usage

```
getUserAnnotations(obj = NULL)
```

Arguments

`obj`	`NULL` or the path to a sitadela SQLite annotation database. If `NULL`, the function will try to guess the location of the SQLite database.

Value

The function returns a `data.frame` object with the installed, custom, user-defined local annotations only.

Author(s)

Panagiotis Moulos

Examples

```
db <- file.path(system.file(package="sitadela"),
    "annotation.sqlite")
if (file.exists(db))
    u <- getUserAnnotations(obj=db)
```

importCustomAnnotation

Import a metaseqR2 custom annotation element

Description

This function imports a custom GTF/GFF file in a manner helpful for the addition of custom annotations to sitadela.

Usage

```
importCustomAnnotation(gtfFile, metadata,
    type = c("gene", "transcript", "utr",
        "transexon", "transutr", "exon"))
```

Arguments

gtfFile	a GTF file containing the gene structure of the organism to be imported.
metadata	a list with additional information about the annotation to be imported. The same as in the addCustomAnnotation man page.
type	same as the type in loadAnnotation.

Value

The function returns a GenomicRanges object with the requested annotation.

Author(s)

Panagiotis Moulos

Examples

```
# Dummy GTF as example
chromInfo <- data.frame(length=c(1000L,2000L,1500L),
    row.names=c("A","B","C"))

# Build with the metadata list filled (you can also provide a version)
if (.Platform$OS.type == "unix" && !grepl("^darwin",R.version$os)) {
    myGenes <- importCustomAnnotation(
        gtfFile=file.path(system.file(package="sitadela"),
            "dummy.gtf.gz"),
        metadata=list(
            organism="dummy",
            source="dummy_db",
            version=1,
            chromInfo=chromInfo
        ),
        type="gene"
```

```
    )
}

## Real data!
## Gene annotation dump from Ensembl
#download.file(paste0("ftp://ftp.ensembl.org/pub/release-98/gtf/",
#  "dasypus_novemcinctus/Dasypus_novemcinctus.Dasnov3.0.98.gtf.gz"),
#  file.path(tempdir(),"Dasypus_novemcinctus.Dasnov3.0.98.gtf.gz"))

## Build with the metadata list filled (you can also provide a version)
#dasGenes <- importCustomAnnotation(
#  gtfFile=file.path(tempdir(),"Dasypus_novemcinctus.Dasnov3.0.98.gtf.gz"),
#  metadata=list(
#    organism="dasNov3_test",
#    source="ensembl_test"
#  ),
#  type="gene"
#)
```

`loadAnnotation` *Load a sitadela simple annotation element*

Description

This function loads an annotation element from a local sitadela annotation database. If the annotation is not found and the organism is supported, the annotation is fetched and created on the fly but not imported in the local database. Use `addAnnotation` for this purpose (build/update/add annotations).

Usage

```
loadAnnotation(genome, refdb,
    type = c("gene", "transcript", "utr",
        "transutr", "transexon", "exon"),
    version="auto", wtv = FALSE,
    db = getDbPath(), summarized = FALSE,
    asdf = FALSE, rc = NULL)
```

Arguments

`genome`	a sitadela supported organism or a custom organism name imported by the user.
`refdb`	a sitadela supported annotation source or a custom name imported by the user.
`type`	the transcriptional unit annotation level to load. It can be one of `"gene"` (default), `"transcript"`, `"utr"`, `"transexon"`, `"transutr"`, `"exon"`. See Details for further explanation of each option.
`version`	the version of the annotation to use. See Details.
`wtv`	load annotations with versioned genes and transcripts when/where available.

db	same as the db in addAnnotation.
summarized	if TRUE, retrieve summarized, non-overlaping elements where appropriate (e.g. exons).
asdf	return the result as a data.frame (default FALSE).
rc	same as the rc in addAnnotation.

Details

Regarding org, it can be, for human genomes "hg18", "hg19" or "hg38", for mouse genomes "mm9", "mm10", for rat genomes "rn5" or "rn6", for drosophila genome "dm3" or "dm6", for zebrafish genome "danrer7", "danrer10" or "danrer11", for chimpanzee genome "pantro4", "pantro5", for pig genome "susscr3", "susscr11", for Arabidopsis thaliana genome "tair10" and for Equus caballus genome "equcab2" and "equcab3". Finally, it can be "USER_NAMED_ORG" with a custom organism which has been imported to the annotation database by the user using a GTF/GFF file. For example org="mm10_p1".

Regarding type, it defines the level of transcriptional unit (gene, transcript, 3' UTR, exon) coordinates to be loaded or fetched if not present. The following types are supported:

- "gene": canonical gene coordinates are retrieved from the chosen database.
- "transcript": all transcript coordinates are retrieved from the chosen database.
- "utr": all 3' UTR coordinates are retrieved from the chosen database, grouped per gene.
- "transutr": all 3' UTR coordinates are retrieved from the chosen database, grouped per \ transcript.
- "transexon": all exon coordinates are retrieved from the chosen database, grouped per transcript.
- "exon": all exon coordinates are retrieved from the chosen database.

Regarding version, this is an integer denoting the version of the annotation to use from the local annotation database or fetch on the fly. For Ensembl, it corresponds to Ensembl releases, while for UCSC/RefSeq, it is the date of creation (locally).

Value

The function returns a GenomicRanges object or a data.frame with the requested annotation.

Author(s)

Panagiotis Moulos

Examples

```
db <- file.path(system.file(package="sitadela"),
    "annotation.sqlite")
if (file.exists(db))
    gr <- loadAnnotation(genome="hg19",refdb="ensembl",
        type="gene",db=db)
```

removeAnnotation *Remove an annotation from a sitadela database*

Description

This function removes a specific annotation from a sitadela database. It does not support multiple organism, resource and version removal for now.

Usage

```
removeAnnotation(org, refdb, ver = NULL, db = NULL)
```

Arguments

org	an existing organism to remove from the database. See also addAnnotation and addCustomAnnotation for details.
refdb	an existing annotation source to remove from the database. See also addAnnotation and addCustomAnnotation for details.
ver	an existing annotation version to remove from the database. See also addAnnotation and addCustomAnnotation for details. If NULL (default), all versions corresponding to org and refdb will be removed.
db	the database to remove from, defaults to getDbPath().

Value

The function return the number of rows removed from the database contents table.

Author(s)

Panagiotis Moulos

Examples

```
# Dummy database as example
customDir <- file.path(tempdir(),"test_remove")
dir.create(customDir)

myDb <- file.path(customDir,"testann.sqlite")
chromInfo <- data.frame(length=c(1000L,2000L,1500L),
    row.names=c("A","B","C"))

# Build with the metadata list filled (you can also provide a version)
if (.Platform$OS.type == "unix") {
    addCustomAnnotation(
        gtfFile=file.path(system.file(package="sitadela"),
            "dummy.gtf.gz"),
        metadata=list(
            organism="dummy",
```

```
            source="dummy_db",
            version=1,
            chromInfo=chromInfo
        ),
        db=myDb
    )

    # Try to retrieve some data
    myGenes <- loadAnnotation(genome="dummy",refdb="dummy_db",
        type="gene",db=myDb)
    myGenes

    # Now remove
    n <- removeAnnotation("dummy","dummy_db",1,myDb)
}
```

---

testFuns *Query and database build testing functions*

Description

This group of testing fuctions can be used to test the entirety of sitadela annotation building capabilities from known resources or custom GTF/GFF files. They are useful for testing the particular annotation the user wishes to build prior to building the final database, in order to avoid failures during the longer build. In all cases, useful messages are also displayed.

Usage

```
testEnsembl(level = c("normal", "long", "short"),
    versioned = FALSE)
testEnsemblSimple(orgs, types, versioned = FALSE)

testUcsc(orgs, refdbs, types, versioned = FALSE)
testUcscAll()

testUcscUtr(orgs, refdbs, versioned = FALSE)
testUcscUtrAll()

testCustomGtf(gtf)

testKnownBuild(org, refdb, ver = NULL, tv = FALSE)
testCustomBuild(gtf, metadata)
```

Arguments

level	how many Ensembl versions from the supported organisms should be checked. It can be "normal" (default), "long" or "short". See also Details.

orgs	a vector of sitadela supported organisms. See also addAnnotation.
refdbs	a vector of sitadela supported annotation. sources. See also addAnnotation.
versioned	use versioned genes/transcripts where available.
types	a vector of sitadela annotation types. See also loadAnnotation.
org	as orgs above but only one organism.
refdb	as refdbs above but only one source.
ver	specific annotation version, see also addAnnotation.
tv	retrieve versioned genes and transcripts when possible, see also addAnnotation.
gtf	a valid GTF or GFF file.
metadata	additional information on the contents of GTF/GFF file. See also addCustomAnnotation.

Details

Regarding testEnsembl and its arguments, when level="normal", only the last one or two (depending on availability with Biomart) supported Ensembl versions are checked for fetching availability. If level="long", all available versions are checked for fetching availability (use with care, it can run for some time!). If level="short", only the last version of each supported organism is checked. Simpler tests with Ensembl (single organisms, types) can be performed with testEnsemblSimple. It will use only the latest version for the asked organism(s).

Regarding testUcsc, it can be used to test the queries used with the UCSC databases for a given organism and database. testUcscAll will test queries for all supported organisms and databases and may take a while to finish.

Similarly, testUcscUtr and testUcscUtrAll will test the queries and building of 3' UTR regions form UCSC databases. 3' UTR constructing is not part of the other UCSC testing functions as the process is different and may be tested only in Unix/Linux machines.

The function testCustomGtf will simply test whether the provided GTF/GFF file can be parsed and used to extract the sitadela annotation types. If this is not possible (rarely), this test will fail. If you wish to test complete database building with a custom GTF/GFF file, use testCustomBuild.

Finally, testKnownBuild will test database building and querying (add/remove annotation) for a single organism.

Value

This group of functions return either a vector of logical values showing success or failure of conducted tests, or a list of test failure reasons or NULL if all tests are successful. Specifically, testKnownBuild and testCustomBuild) return logicals while all the rest return NULL if tests are successful or a list of failure reasons (and the respective test) otherwise.

Author(s)

Panagiotis Moulos

Examples

```
# Test a dummy GTF file
gtf <- file.path(system.file(package="sitadela"),
    "dummy.gtf.gz")
chromInfo <- data.frame(length=c(1000L,2000L,1500L),
    row.names=c("A","B","C"))
metadata=list(
    organism="dummy",
    source="dummy_db",
    version=1,
    chromInfo=chromInfo
)

testResult <- testCustomBuild(gtf,metadata)
# For this case, just testResult <- testCustomBuild()
# would also work

# More real tests
if (require(RMySQL))
    f <- testUcsc("hg19","refseq","gene",TRUE)

# Test a complete build for Ensembl mm9
# testResult <- testKnownBuild()

# Test a complete build for UCSC dm6
# testResult <- testKnownBuild("dm6","ucsc")
```

Index

addAnnotation, 2, *7*, *10*, *14*, *15*, *17*
addCustomAnnotation, 5, *12*, *15*, *17*

data.frame, *5*, *14*

getAnnotation, 7
getChromInfoFromUCSC, *9*
getDbPath *(*getsetDbPath*)*, 10
getInstalledAnnotations, 9, *11*
getSeqInfo, 9
getsetDbPath, 10
getUserAnnotations, 11

importCustomAnnotation, 12

loadAnnotation, *12*, 13, *17*

removeAnnotation, 15

setDbPath *(*getsetDbPath*)*, 10

testCustomBuild *(*testFuns*)*, 16
testCustomGtf *(*testFuns*)*, 16
testEnsembl *(*testFuns*)*, 16
testEnsemblSimple *(*testFuns*)*, 16
testFuns, 16
testKnownBuild *(*testFuns*)*, 16
testUcsc *(*testFuns*)*, 16
testUcscAll *(*testFuns*)*, 16
testUcscUtr *(*testFuns*)*, 16
testUcscUtrAll *(*testFuns*)*, 16