Package 'refGenome'

September 18, 2014

Type Package

Title Gene And Splice Site Annotation Using Annotation Data From Ensembl And UCSC Genome Browsers

Version 1.3.0

Date 2013-05-08

Author Wolfgang Kaisers

Maintainer Wolfgang Kaisers <kaisers@med.uni-duesseldorf.de>

Description The packge contains functionality for import and managing of downloaded genome annotation Data from Ensembl genome browser (European Bioinformatics Institute) and from UCSC genome browser (University of California, Santa Cruz) and annotation routines for genomic positions and splice site positions.

License GPL-2

Depends methods,doBy,RSQLite

Imports DBI

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-09-18 09:56:19

R topics documented:

`refGenome-package` *Managing annotation data for reference Genomes from UCSC and Ensembl.*

Description

The package contains classes for managing (GTF-) annotation data for UCSC and Ensembl genomes. Data can be imported, merged, viewed, searched and saved (as .RData and as SQLite database). There is also a C-routine for detection of overlapping (alignment) ranges with annotated regions.

Details

Package:	refGenome
Type:	Package
Version:	1.0
Date:	2012-10-06
License:	What license is it under?
Depends:	methods

Author(s)

Wolfgang Kaisers Maintainer: Wolfgang Kaisers <kaisers@med.uni-duesseldorf.de>

Examples

```
ens<-ensemblGenome()
basedir(ens)<-system.file("extdata",package="refGenome")
ens_gtf<-"hs.ensembl.62.small.gtf"
read.gtf(ens,ens_gtf)
tableAttributeTypes(ens)
moveAttributes(ens,c("gene_name","transcript_name","exon_number"))
ddx<-extractByGeneName(ens,"DDX11L1")
ddx
fam<-extractTranscript(ens,"ENST00000417324")
fam
```

```
enpa<-extractSeqids(ens,ensPrimAssembly())
enpa
tableTranscript.id(ens)
tableTranscript.name(ens)
```

---

addIsCoding *Add information on coding/non-coding status.*

Description

The addIsCoding method extracts positions from 'CDS' features in the ensemblGenome object. The positions of the 'CDS' features are compared with the splice-junction positions in the ensemblJunctions object. When a match is found, the splice junction is marked as coding, otherwise the junction is marked as non-coding. The coding information is calculated for each flanking exon of the splice-junction. The column name for the left side (lend) is licd, the name for the right side (rstart) is ricd.

Usage

```
addIsCoding(object,ens)
```

Arguments

object	ensemblJunctions. The object to which isCoding columns are added.
ens	ensemblGenome. Genome object which should contain 'CDS' features.

Author(s)

Wolfgang Kaisers

Examples

```
ef<-system.file("extdata", "hs.ensembl.62.small.RData", package="refGenome")
ens<-loadGenome(ef)
enj<-getSpliceTable(ens)
addIsCoding(enj,ens)
```

ensemblGenome-class *Class* "ensemblGenome"

Description

ensemblGenome represents ensembl genomic annotation data.

Objects from the Class

Objects can be created by calls of the form `ensemblGenome(dbfile)`. 'dbfile' represents SQLite database file.

Slots

`basedir`: Object of class "character" Directory where SQLite database is written.

`ev`: Object of class "environment" Environment that contains data structures. Optionally, there are gtf and attr data.frames.

Methods

show `signature(object = "refGenome")`: Creates a sensible printout.

getGtf `signature(object = "refGenome")`: Returns content of gtf table.

setGtf `signature(object = "refGenome")`: Writes content of gtf table.

getAttr `signature(object = "refGenome")`: Returns content of attribute table.

getGeneTable `signature(object = "refGenome")`: Returns content of genes table when table exists. Otherwise NULL is returned.

setAttr `signature(object = "refGenome")`: Writes content of attribute table.

read.gtf `signature(object = "refGenome")`: Imports content of gtf file. This is the basic mechanism for data import. It works the same way for ucscGenome and for ensemblGenome. The attribute items are parsed and written in parts to gtf table and attribute table.

extractPaGenes `signature(object="ensemblGenome")`: Extracts all annotations on primary assembly. The function returns a data.frame. Used as shortcut to directly extract a table from gtf files.

extractFeature `signature(object="ensemblGenome")`: Extracts annotated positions which are classified as given 'feature' argument. Returns an 'ensemblGenome' object.

extractByGeneName `signature(object="ensemblGenome", geneNames="character")`: Extracts ensemblGenome object which contains table subsets. When none of the `geneNames` matches, the function returns `NULL`.

extractTranscript `signature(object="ensemblGenome", transcripts="character")`: Extracts ensemblGenome object which contains table subsets

getGenePositions `signature(object="ucscGenome", force="logical")`: Extracts table with position data for whole genes (smallest exon start position and largest exon end position. A copy of the table will be placed inside the internal environment. Upon subsequent call only a copy of the contained table is returned unless `force=TRUE` is given. Upon `force=TRUE` new gene positions are calculated regardless of existing tables.)

tableTranscript.name `signature(object="ensemblGenome")`: Extracts table object which contains tabled 'transcript_name' column of gtf table

tableTranscript.id `signature(object="ensemblGenome")`: Extracts table object which contains tabled 'transcript_id' column of gtf table

writeDB `signature(object = "refGenome")`: Copies content of gtf, attr and xref table to database.

Author(s)

Wolfgang Kaisers

References

http://www.ensembl.org/info/data/ftp/index.html http://mblab.wustl.edu/GTF22.html#fields

Examples

```
# + + + + + + + + + + + + + + + + + + + + #
# Create an instance from scratch
# Real data:
# ftp://ftp.ensembl.org/pub/release-70/gtf/homo_sapiens/Homo_sapiens.GRCh37.70.gtf.gz
# + + + + + + + + + + + + + + + + + + + + #
ens<-ensemblGenome()
basedir(ens)<-system.file("extdata",package="refGenome")
ens_gtf<-"hs.ensembl.62.small.gtf"
read.gtf(ens,ens_gtf)
tableAttributeTypes(ens)
moveAttributes(ens,c("gene_name","transcript_name","exon_number"))
# Loading of a saved genome:
ensfile<-system.file("extdata", "hs.ensembl.62.small.RData", package="refGenome")
ens<-loadGenome(ensfile)

# + + + + + + + + + + + + + + + + + + + + #
# Saving and loading
# Save as R-image (fast loading)
# + + + + + + + + + + + + + + + + + + + + #
basedir(ens)<-getwd()
saveGenome(ens,"hs.ensembl.62.small.RData",useBasedir=FALSE)
enr<-loadGenome("hs.ensembl.62.small.RData")

# Save as SQLite database
# + + + + + + + + + + + + + + + + + + + + #
# Commented out because RSQlite
# seems to produce memory leaks
# + + + + + + + + + + + + + + + + + + + + #
#writeDB(ens,filename="ens62.db3",useBasedir=FALSE)
#edb<-loadGenomeDb(filename="ens62.db3")

# + + + + + + + + + + + + + + + + + + + + #
# Extract data for Primary Assembly seqids
# + + + + + + + + + + + + + + + + + + + + #
```

```
enpa<-extractSeqids(ens,ensPrimAssembly())
# Tables all features in 'gtf' table
tableFeatures(enpa)
# Extract Coding sequences for Primary Assemblys
enpafeat<-extractFeature(enpa,"exon")
# Shortcut. Returns data.frame
engen<-extractPaGenes(ens)

# + + + + + + + + + + + + + + + + + + + #
# Extract data for indival Genes
# + + + + + + + + + + + + + + + + + + + #
ddx<-extractByGeneName(ens,"DDX11L1")
ddx
tableTranscript.id(ddx)
tableTranscript.name(ddx)
fam<-extractTranscript(ens,"ENST00000417324")
fam
# Extract range limits of entire Genes
gp<-getGenePositions(ens)
gp
# + + + + + + + + + + + + + + + + + + + #
```

`extractByGeneName` *Extract subsets of refGenome by gene-name.*

Description

The function takes objects derived from `refGenome` or `refJunctions` and returns a subset in which `gene_name` matches the given values. The returned object is of the same class as the given object.

Usage

```
extractByGeneName(object,geneNames,src,...)
```

Arguments

`object`	refGenome (or derived). Object from which subset is extracted.
`geneNames`	Character. Vector with gene names.
`src`	Unused within this package. Needed for compatibility reasons.
`...`	(unused)

Value

Same class as `object`

Author(s)

Wolfgang Kaisers

Examples

```
# + + + + + + + + + + + + + + + + + + + #
# A) Extract from Genome
# + + + + + + + + + + + + + + + + + + + #
ensfile<-system.file("extdata",
                     "hs.ensembl.62.small.RData", package="refGenome")
ens<-loadGenome(ensfile)
ws7<-extractByGeneName(ens,"WASH7P")
ws7
# + + + + + + + + + + + + + + + + + + + #
# B) Extract from splice junctions
# + + + + + + + + + + + + + + + + + + + #
junc<-getSpliceTable(ens)
ddx<-extractByGeneName(junc,"DDX11L1")
ddx
```

`getGenePositions` *Extract subsets of refGenome by gene-name.*

Description

The function searches in the given data for unique `gene_id` values. For each item, values like gene_id, seqid and strand are extracted. Optionally (when present) also gene_name and gene_biotype are extracted. The function assignes unique id values which are ordered by gene_id values (not genetic positions). Owing to this, id values equal `as.numeric(gene_id)`.

Usage

```
getGenePositions(object,by,force=FALSE,...)
```

Arguments

`object`	ensemblGenome, ucscGenome or ensemblJunctions. Object from which gene positions are extracted.
`by`	Character. Determines criterion by which genes are discerned. Accepted values: "gene_id" and "gene_name". For Ensembl genomes the default is 'gene_id' and for USCC genomes the default is 'gene_name'.
`force`	Logical. When FALSE, gene positions will only be calculated when a position table is not present in local environment. The function then returns a copy.
`...`	Unused.

Details

The function stores a copy of the result in the internal environment (`genes` table). Upon subsequent calls the values only are re-calculated when force=TRUE is given. Otherwise the function returns a copy of the contained table. Present `genes` tables will be automatically saved and restored by `saveGenome` and `load.X` functions.

Value

data.frame

Author(s)

Wolfgang Kaisers

Examples

```
# + + + + + + + + + + + + + + + + + + + #
# A) Ensembl genome:
# + + + + + + + + + + + + + + + + + + + #
ensfile<-system.file("extdata",
                        "hs.ensembl.62.small.RData", package="refGenome")
ens<-loadGenome(ensfile)
gp<-getGenePositions(ens)

# + + + + + + + + + + + + + + + + + + + #
# B) Ensembl junctions:
# + + + + + + + + + + + + + + + + + + + #
junc<-getSpliceTable(ens)
genes<-getGenePositions(junc)

# + + + + + + + + + + + + + + + + + + + #
# C) Ucsc genome:
# + + + + + + + + + + + + + + + + + + + #
ucfile<-system.file("extdata", "hs.ucsc.small.RData", package="refGenome")
uc<-loadGenome(ucfile)
junc<-getSpliceTable(uc)
gp<-getGenePositions(junc)
```

---

overlap *overlap function*

Description

Overlaps query ranges with reference ranges. The function assumes that there is no overlap between reference ranges.

Usage

```
overlap(qry, ref)
```

Arguments

qry	data.frame with query ranges. qry should have columns 'id','start' and 'end'. The routine assumes that the table is ascending sorted by column 'start'.
ref	data.frame with reference ranges. ref should have columns 'id','start' and 'end'. The routine assumes that the table is ascending sorted by column 'start'.

Details

The routine assumes that qry and ref tables are ascending sorted by column 'start'. Otherwise the result will be incorrect (i.e. missing hits). The function assumes that there is no overlap between reference ranges. It will otherwise return only one, possibly arbitrary, hit per query range.

Value

The function returns a data.frame

`overlap`	Factor which encodes type of overlap between query and reference range. Levels no (no overlap; refid is set to 0), l (qry left overhangs ref), n (qry is contained in ref), r (qry right overhangs ref)
`leftDiff`	Distance on left side between margins of query and reference.
`rightDiff`	Distance on right side between margins of query and reference.
`queryid`	id from qry table.
`refid`	id from ref table.

Author(s)

Wolfgang Kaisers

Examples

```
qry <- data.frame(id=1:6, start=c(10,18,61,78,82,110), end=c(15,22,63,87,90,120))
ref <- data.frame(id=1:5, start=c(20,40,60,80,100), end=c(25,45,65,85,105))
overlap(qry, ref)
```

`overlapJuncs` *overlapJuncs function*

Description

Overlaps query gap-sites with annotated splice junctions.

Usage

```
overlapJuncs(qry, junc)
```

Arguments

`qry`	`data.frame`. Table with query ranges. qry should have columns 'id', 'seqid', 'lstart', 'lend', 'rstart', 'rend'.
`junc`	`refJunctions`. Object which contains table of splice junctions in reference genome.

Details

The function finds for each given gap-site. A gap-site is the combination of two genomic regions (= exons) which enclose an intermediary (= intron). The function identifies junction records which overlap with the given gap-site (=hits) and select a junction with an optimal fit. The goodness of fit is measured by the distance of the inner gap boundaries (= the splice sites) between query and junction record. A junction with minimal sum of upstream and downstrem distances is selected. The selection of the best hit depends on the order a version of the junction table wich is sorted by `lstart` and `rend`.)

Value

The function returns a data.frame

`qid`	`Integer`. Query id value from qry table
`refid`	`Integer`. Reference id from junctions object for best hit.
`ldiff`	`Integer`. Difference between lend values in qry and `junc` table for best hit (refid) record.
`rdiff`	`Integer`. Difference between rstart values in qry and `junc` table for best hit (refid) record.
`nref`	`Integer`. Number of `junc` records which possibly overlap with query item. `nref=0` when no overlap has been found for query.
`sod`	`Integer`. Sum of distances (=abs(ldiff) + abs(rdiff)). `sod=0` when qry exactly hits an annotated site. `sod=NA` when no overlap has been found for query.
`first_refid`	`Integer`. `id` for first overlapping record in `junc` table.
`last_refid`	`Integer`. `id` for last overlapping record in `junc` table.
`nadv`	`Integer`. Number of advancing iterations during search for
`strand`	Strand value derived from annotation.
`gene_id`	Gene id from `refJunctions` or `genpos` table.
`transcript_id`	Transcript id from `refJunctions` table.
`gene_name`	Gene name from `refJunctions` or `genpos` table.

Author(s)

Wolfgang Kaisers

Examples

```
# + + + + + + + + + + + + + + + + + + + + #
# A) Example query  data
# + + + + + + + + + + + + + + + + + + + + #
##                         1       2       3       4       5       6       7 ##
qry<-data.frame(id = 1:7, seqid = "1",
            lstart = c(10100L, 11800L, 12220L, 12220L, 12220L, 32000L, 40000L),
            lend =   c(10100L, 12000L, 12225L, 12227L, 12227L, 32100L, 40100L),
            rstart = c(10200L, 12200L, 12057L, 12613L, 12650L, 32200L, 40200L),
            rend =   c(10300L, 12250L, 12179L, 12620L, 12700L, 32300L, 40300L))
##                         1       2       3       4       5       6       7 ##

# + + + + + + + + + + + + + + + + + + + + #
# B) Example reference data
# + + + + + + + + + + + + + + + + + + + + #
# B.1) Ensembl genome:
ensfile<-system.file("extdata", "hs.ensembl.62.small.RData", package="refGenome")
ens<-loadGenome(ensfile)
gp<-getGenePositions(ens)
# B.2) Ensembl junctions:
junc<-getSpliceTable(ens)
# + + + + + + + + + + + + + + + + + + + + #
# C) Do overlap
# + + + + + + + + + + + + + + + + + + + + #
res<-overlapJuncs(qry,junc)
```

---

refExons-class *Class* "refExons"

Description

refExons represents genomic annotation data on exon-features for Ensembl and UCSC genomes. Relative locations of CDS, start_codon and stop_codon features are added.

Objects from the Class

Objects can be created by calls of the form `refExons(rg)`. 'rg' represents an object of class `ensemblGenome` or `ucscGenome`.

Slots

`basedir`: Object of class `"character"` Directory where SQLite database is written.

`ev`: Object of class `"environment"` Environment that contains data structures. Optionally, there are gtf and attr data.frames.

Methods

show `signature(object = "ensemblExons")`: Creates a sensible printout.

getSpliceTable `signature(object = "ensemblExons", coding="logical")`: Returns tabled splice sites. When `coding=TRUE` only entries with `gene_biotype=="protein_coding"` are included.

Author(s)

Wolfgang Kaisers

References

Ensembl File index http://www.ensembl.org/info/data/ftp/index.html

GTF Field definitions http://mblab.wustl.edu/GTF22.html#fields

UCSC home page http://genome.ucsc.edu/

Examples

```
# + + + + + + + + + + + + + + + + + + + #
# A) Ensembl
# + + + + + + + + + + + + + + + + + + + #
ensfile<-system.file("extdata",
                        "hs.ensembl.62.small.RData", package="refGenome")
ens<-loadGenome(ensfile)
enex<-refExons(ens)
saveGenome(enex,"enex.RData", useBasedir=FALSE)
er<-loadGenome("enex.RData")

# + + + + + + + + + + + + + + + + + + + #
# B) UCSC
# + + + + + + + + + + + + + + + + + + + #
ucfile<-system.file("extdata", "hs.ucsc.small.RData", package="refGenome")
uc<-loadGenome(ucfile)
ucex<-refExons(uc)
saveGenome(ucex, "ucex.RData", useBasedir=FALSE)
ur<-loadGenome("ucex.RData")
```

---

refGenome-class *Class* "refGenome"

Description

refGenome class: Virtual base class for ucscGenome and ensemblGenome.

Objects from the Class

Objects can be created by calls of the form `refGenome(dbfile)`. dbfile represents SQLite database file.

Slots

`basedir`: Object of class `"character"` Directory where SQLite database is written.

`ev`: Object of class `"environment"` Environment that contains data structures. Optionally, there are gtf and attr data.frames.

Methods

show `signature(object = "refGenome")`: Creates a sensible printout.

tableSeqids `signature(object = "refGenome")`: Returns tabled seqids which counts all annotations for each seqid. The regex argument will display, which seqid is covered by regex. Intended as preparation for extractSeqids.

extractSeqids `signature(object = "refGenome")`: Returns a filtered version of object. Only data for the 'regex' specified seqids is contained. The functions 'ucPrimAssembly' and 'ensPrimAssembly' return regular expressions that allow extraction of primary assemblies for UCSC and Ensembl respectively.

getGtf `signature(object = "refGenome")`: Returns content of gtf table.

setGtf `signature(object = "refGenome")`: Writes content of gtf table.

getAttr `signature(object = "refGenome")`: Returns content of attribute table.

setAttr `signature(object = "refGenome")`: Writes content of attribute table.

read.gtf `signature(object = "refGenome")`: Imports content of gtf file. This is the basic mechanism for data import. It works the same way for ucscGenome and for ensemblGenome. The attribute items are parsed and written in parts to gtf table and attribute table.

tableAttributeTypes `signature(object= "refGenome")`: Prints out number of rows for each type in attribute table.

saveGenome `signature(object = "refGenome", filename="character", useBasedir="logical")`: Saves content of all tables to RData file. When useBasedir is set (default), basedir (from basedir-slot) is prefixed.

writeDB `signature(object = "refGenome")`: Copies content of gtf, attr and xref table to database.

Author(s)

Wolfgang Kaisers

Examples

```
# + + + + + + + + + + + + + + + + + + + + #
# Loading sample data:
# + + + + + + + + + + + + + + + + + + + + #
ensfile<-system.file("extdata", "hs.ensembl.62.small.RData", package="refGenome")
ens<-loadGenome(ensfile)
ens
ddx<-extractByGeneName(ens,"DDX11L1")
ddx
ucfile<-system.file("extdata", "hs.ucsc.small.RData", package="refGenome")
uc<-loadGenome(ucfile)
```

```
uc
ddx<-extractByGeneName(uc,"DDX11L1")
ddx
```

`refJunctions-class` *Class* `"refJunctions"`

Description

refJunctions represents ensembl genomic annotation data for splice-junctions.

Objects from the Class

Objects can be created by calls of the form getSpliceTable(rg) where 'rg' is an object of class `refGenome` (ensemblGenome or ucscGenome).

Slots

`basedir`**:** Object of class `"character"` Directory where SQLite database is written.

`ev`**:** Object of class `"environment"` Environment that contains data structures. The splice-junction data is stored in 'gtf' named data.frame inside ev. Optionally the environment also contains the result tables from `unifyJuncs` or `getGenePositions` functions. They are automatically included in save and load procedures.

Methods

show Creates a sensible printout.

unifyJuncs `signature(object = "refJunctions")`: Calculates unique splice-sites and associates each site with gene-id. Adds uid to 'gtf' table and creates a new 'unique junction site' (ujs) table.

getGenePositions `signature(object="refJunctions", force="logical")`: Extracts table with position data for whole genes (smallest exon start position and largest exon end position. A copy of the table will be placed inside the internal environment. Upon subsequent call only a copy of the contained table is returned unless `force=TRUE` is given. Upon `force=TRUE` new gene positions are calculated regardless of existing tables.)

Author(s)

Wolfgang Kaisers

References

http://www.ensembl.org/info/data/ftp/index.html http://mblab.wustl.edu/GTF22.html#fields

Examples

```
# + + + + + + + + + + + + + + + + + + + + #
# A) Ensembl
# + + + + + + + + + + + + + + + + + + + + #
ef<-system.file("extdata", "hs.ensembl.62.small.RData", package="refGenome")
ens<-loadGenome(ef)
enj<-getSpliceTable(ens)
ufe<-unifyJuncs(enj)
saveGenome(enj,"enj.RData",useBasedir=FALSE)
enjr<-loadGenome("enj.RData")

# + + + + + + + + + + + + + + + + + + + + #
# B) UCSC
# + + + + + + + + + + + + + + + + + + + + #
uf<-system.file("extdata", "hs.ucsc.small.RData", package="refGenome")
uc<-loadGenome(uf)
ucj<-getSpliceTable(uc)
ufu<-unifyJuncs(ucj)
saveGenome(ucj,"ucj.RData",useBasedir=FALSE)
ucjr<-loadGenome("ucj.RData")
```

---

saveGenome *Saving and loading* refGenome *objects*

Description

refGenome objects contain all annotation data inside an environment. During saving and loading, the whole content of the environment is loaded and saved. The correct object type is also determined in this way.

Usage

```
saveGenome(object,filename,useBasedir=TRUE,...)
```

Arguments

object	refGenome (or derived)
filename	Character. Filename of the R-data-file wich is written.
useBasedir	Logical. When TRUE the output file is written into basedir. Otherwise the directory depicted by dirname(filename) is used.
...	(unused)

Author(s)

Wolfgang Kaisers

Examples

```
# + + + + + + + + + + + + + + + + + + + + #
# A) Ensembl genome
# + + + + + + + + + + + + + + + + + + + + #
ensfile<-system.file("extdata", "hs.ensembl.62.small.RData", package="refGenome")
ens<-loadGenome(ensfile)
saveGenome(ens,"ens.RData",useBasedir=FALSE)

# + + + + + + + + + + + + + + + + + + + + #
# B) Ensembl junctions
# + + + + + + + + + + + + + + + + + + + + #
junc<-getSpliceTable(ens)
saveGenome(junc,"junc.RData",useBasedir=FALSE)
loadGenome("junc.RData")

# + + + + + + + + + + + + + + + + + + + + #
# C) Ensembl exons
# + + + + + + + + + + + + + + + + + + + + #
enex<-refExons(ens)
saveGenome(enex,"enex.RData",useBasedir=FALSE)
er<-loadGenome("enex.RData")

# + + + + + + + + + + + + + + + + + + + + #
# D) UCSC genome:
# + + + + + + + + + + + + + + + + + + + + #
ucfile<-system.file("extdata", "hs.ucsc.small.RData", package="refGenome")
uc<-loadGenome(ucfile)
saveGenome(uc,"uc.RData",useBasedir=FALSE)

# + + + + + + + + + + + + + + + + + + + + #
# E) UCSC junctions
# + + + + + + + + + + + + + + + + + + + + #
junc<-getSpliceTable(uc)
saveGenome(junc,"junc.RData",useBasedir=FALSE)
jr<-loadGenome("junc.RData")

# + + + + + + + + + + + + + + + + + + + + #
# F) UCSC exons
# + + + + + + + + + + + + + + + + + + + + #
ucex<-refExons(uc)
saveGenome(ucex,"ucex.RData",useBasedir=FALSE)
ur<-loadGenome("ucex.RData")
```

---

ucscGenome-class *Class* "ucscGenome"

Description

ucscGenome class: Represents data stored for UCSC genome. The standard way to import data is to download a "gtf" file from the UCSC Genome Browser (-> Table Browser). Download the "knownGene" Table in output format "GTF". Then import the data via the `read.gtf` function.

Objects from the Class

Objects can be created by calls of the form `ucscGenome()`.

Slots

`basedir`**:** Object of class `"character"` Directory where SQLite database is written.

`ev`**:** Object of class `"environment"` Environment that contains data structures. Optionally, there are gtf, attr and additionally xref data.frames.

Methods

show `signature(object = "refGenome")`: Creates a sensible printout.

getGtf `signature(object = "refGenome")`: Returns content of gtf table.

setGtf `signature(object = "refGenome")`: Writes content of gtf table.

getAttr `signature(object = "refGenome")`: Returns content of attribute table.

setAttr `signature(object = "refGenome")`: Writes content of attribute table.

read.gtf `signature(object = "refGenome")`: Imports content of gtf file. This is the basic mechanism for data import. It works the same way for ucscGenome and for ensemblGenome. The attribute items are parsed and written in parts to gtf table and attribute table.

writeDB `signature(object = "refGenome")`: Copies content of gtf, attr and xref table to database.

addEnsembl `signature(object = "ucscGenome")`: Imports UCSC 'knownToEnsembl' table. It's appended to the gtf table.

addIsoforms `signature(object = "ucscGenome")`: Imports UCSC ' knownIsoforms' table. It's appendet to the gtf table.

addXref `signature(object = "ucscGenome")`: Imports UCSC 'kgXref' table. A 'geneSymbol' column is added to gtf table. The rest is written into xref table.

extractByGeneName `signature(object="ucscGenome", geneNames="character")`: Extracts ucscGenome object which contains table subsets. When none of the `geneNames` matches, the function returns `NULL`.

getXref `signature(object = "ucscGenome")`: Returns content of xref table.

getGenePositions `signature(object="ucscGenome", force="logical")`: Extracts table with position data for whole genes (smallest exon start position and largest exon end position. A copy of the table will be placed inside the internal environment. Upon subsequent call only a copy of the contained table is returned unless `force=TRUE` is given. Upon `force=TRUE` new gene positions are calculated regardless of existing tables.)

loadGenome `signature(filename = "character")`: Imports data from stored R-Environment Image.

loadGenomeDb `signature(filename = "character")`: Imports content of object from sqlite3 database.

tableFeatures `signature(object="ucscGenome")`: Tables content of "feature" column.

tableTranscript.id `signature(object="ucscGenome")`: Tables values in transcript_id column.

extractTranscript `signature(object="ucscGenome", transcripts="character")`: Extracts an object which contains data for subset defined by transcript names.

Author(s)

Wolfgang Kaisers

References

http://genome.ucsc.edu/

Examples

```
# + + + + + + + + + + + + + + + + + + + + #
# Loading and saving
# From and to R-image (fast loading)
# + + + + + + + + + + + + + + + + + + + + #
ucfile<-system.file("extdata", "hs.ucsc.small.RData", package="refGenome")
uc<-loadGenome(ucfile)
uc
saveGenome(uc,"uc.RData",useBasedir=FALSE)
ucr<-loadGenome("uc.RData")

# + + + + + + + + + + + + + + + + + + + + #
# Extract data for Primary Assembly seqids
# + + + + + + + + + + + + + + + + + + + + #
ucpa<-extractSeqids(uc,ucPrimAssembly())
# Extract data for indival Genes
ddx<-extractByGeneName(uc,"DDX11L1")
ddx
# Extract range limits of entire Genes
gp<-getGenePositions(uc)
gp
tableFeatures(uc)
extractByGeneName(ucpa,"DDX11L1")
tableTranscript.id(ucpa)

# + + + + + + + + + + + + + + + + + + + + #
# Create object from scratch
# + + + + + + + + + + + + + + + + + + + + #
# uc<-ucscGenome()
# basedir(uc)<-"/my/genome/basedir"
# Place all UCSC-files in folder
# read.gtf(uc,"knownGene.gtf")
# addXref(uc,"kgXref.csv")
# addEnsembl(uc,"knownToEnsembl.csv")
# addIsoforms(uc,"knownisoforms.csv")
# + + + + + + + + + + + + + + + + + + + + #
```

`unifyJuncs` *Unification of splice-junctions.*

Description

Overlaps query ranges with reference ranges. The function assumes that there is no overlap between reference ranges.

Usage

```
unifyJuncs(object)
```

Arguments

object	refJunctions object. Contains splice-junction data.

Details

Many splices sites are multiple times contained when calculated from transcripts. In order to obtain unique splice positions the function extracts one data set per (seqid, lend, rstart) combination. For each site the annotation information (gene_id, strand, fexid) is extracted from the most abundand gene name.

Value

The function returns a data.frame with the following columns:

id	Numeric index for unique site.
seqid	Package
lstart	1.0
lend	2012-10-06
rstart	What license is it under?
rend	methods
nSites	Number of refJunctions (transcripts) that contain this site.
gene_id	Gene identifier.
strand	Strand on which gene resides.
fexid	Id value of first refJunction with coordinates.

Author(s)

Wolfgang Kaisers

Examples

```
# + + + + + + + + + + + + + + + + + + + + #
# A) Ensembl
# + + + + + + + + + + + + + + + + + + + + #
ef<-system.file("extdata", "hs.ensembl.62.small.RData", package="refGenome")
ens<-loadGenome(ef)
enj<-getSpliceTable(ens)
ufe<-unifyJuncs(enj)
saveGenome(enj,"enj.RData",useBasedir=FALSE)
enjr<-loadGenome("enj.RData")
```

```
# + + + + + + + + + + + + + + + + + + + + #
# B) UCSC
# + + + + + + + + + + + + + + + + + + + + #
uf<-system.file("extdata", "hs.ucsc.small.RData", package="refGenome")
uc<-loadGenome(uf)
ucj<-getSpliceTable(uc)
ufu<-unifyJuncs(ucj)
saveGenome(ucj,"ucj.RData",useBasedir=FALSE)
ucjr<-loadGenome("ucj.RData")
```

`writeDB` *Saving and loading* `refGenome` *objects to and from SQLite databases.*

Description

`refGenome` objects keep annotation data in data.frames. The content of the data.frames is written to or extracted from a SQLite database.

Usage

```
writeDB(object,filename,useBasedir=TRUE,...)
```

Arguments

`object`	refGenome (or derived)
`filename`	Character. Filename of the R-data-file wich is written.
`useBasedir`	Logical. When TRUE the database will be created in basedir. Otherwise the database will be created in the current working directory or elsewhere (when path is included in filename).
`...`	(unused)

Author(s)

Wolfgang Kaisers

Examples

```
# + + + + + + + + + + + + + + + + + + + + #
# Commented out because RSQlite
# seems to produce memory leaks
# + + + + + + + + + + + + + + + + + + + + #
# A) Ensembl genome:
#ensfile<-system.file("extdata",
#                       "hs.ensembl.62.small.RData", package="refGenome")
#ens<-loadGenome(ensfile)
#writeDB(ens,"ens.db3",useBasedir=FALSE)
#ens<-loadGenomeDb("ens.db3")
# + + + + + + + + + + + + + + + + + + + + #
```

```
#ucfile<-system.file("extdata", "hs.ucsc.small.RData", package="refGenome")
#uc<-loadGenome(ucfile)
#writeDB(uc,"uc.db3",useBasedir=FALSE)
#uc<-loadGenomeDb("uc.db3")
# + + + + + + + + + + + + + + + + + + + + #
```

Index