Package 'DASiR'

October 27, 2015

Type Package

Title Distributed Annotation System in R

Version 1.10.0

Date 2015-06-08

Author Oscar Flores, Anna Mantsoki

Maintainer Ricard Illa <ricard.illa@irbbarcelona.org>

Description R package for programmatic retrieval of information from DAS servers.

License LGPL (>= 3)

Depends Biostrings,IRanges,GenomicRanges

Imports XML

LazyLoad yes

biocViews Annotation

NeedsCompilation no

R topics documented:

`DASiR-package` *Distributed Annotation System in R*

Description

R package for programmatic retrieval of information from DAS servers

Details

Package:	DASiR
Type:	Package
License:	LGPL(>= 3)

Distributed Annotation System (DAS) is a protocol for information exchange between a server and a client. Is widely used in bioinformatics and the most important repositories include a DAS server parallel to their main front-end.

DAS uses XML and HTTP protocols, so the requirements are much less than using MySQL or BioMart based alternatives.

This package provides a convenient R-DAS interface to programmatically access DAS servers available from your network. It supports only the basic features of DAS 1.6 protocol (DAS/2 is not supported) but these should be enough for most of the users.

Despite a large number (>1200 according official numbers on February 2013) are available, this package is designed with range-features (neither coverages nor sequences). DAS also supports querying genomic sequences and protein structures, but there are already better ways to access such static data in R.

In order to start with this package, you can give a look to basic metadata functions in `getDas-functions` man page and to the main functions `getDasFeature`, `getDasSequence`, `getDasStructure`.

Basic support for graphical representation is also provided through `plotFeatures`.

As a final consideration, you should have in mind that public available DAS servers are a valuable service and you should give them a reasonable use. Please, don't overload the servers with many or large queries.

Author(s)

Oscar Flores <oflores@mmb.pcb.ub.es>

Anna Mantsoki <anna.mantsoki@bsc.es>

References

BioDas Wiki - http://www.biodas.org/

DAS 1.6 Specification - http://www.ebi.ac.uk/~aj/1.6_draft7/documents/spec.html

DAS servers catalog - http://www.dasregistry.org/

Examples

```
setDasServer(server="http://www.ensembl.org/das")
print(getDasServer())

# See the sources (supported organisms)
sources <- getDasSource()
head(sources)

dsn_sources <- getDasDsn()
head(dsn_sources)

# Notice that sources returns more information but we need the source ID
# given by getDasDsn, that corresponds to getDasSource$title in this
```

```
# server.

# The server we are quering is a genomic sequence server, so no gene
# information nor structure is available here

# Let's see what this sever supports for SCerevisiae
source <- "Saccharomyces_cerevisiae.EF4.reference"
# das1:entry_points das1:sequence das1:features
sources[grep(source, sources$title), ]

# Ask for the entries
entries <- getDasEntries(source)
head(entries)

# Retrive first 1000nts in the beginning of chromosomes I and II
range <- GRanges(c("I", "II"), IRanges(start=1, end=1000))
seq <- getDasSequence(source, range, class="DNAStringSet")

# Server supports features, but not types...
types <- getDasTypes(source)
# Server returns null... So we will perform a query without types
print(types)

# Let's see which features is returning the server
features <- getDasFeature(source, range, NULL)
print(features)
# Features here are not genomic features (like genes), this is only
# sequence info

# In the webpage http://www.dasregistry.org/listSources.jsp
# We can see all the registred DAS servers and the information they provide

# Remember that DASiR is only an interface to DAS servers! You should know
# what the server you are querying contains and if it supports your desired
# features!

# Now let's retrieve genes in that region from another source
# This is UCSC Genome browser, which has access to all the features
# displayed in the webpage
setDasServer(server="http://genome.ucsc.edu/cgi-bin/das")
# Notice that id now changes, we can retrieve it from getDasDsn()
source <- "sacCer3"

# Retrieve the types
types <- getDasTypes(source)

# We want the official genes from 'sdgGene' and the ENSEMBL predicted genes
# from 'ensGene'
features <- getDasFeature(source, range, c("sgdGene","ensGene"))
print(features)

# We obtain a total of 4 genes, 2 in chromosome I and 2 in chromosome II
# Notice each gene appears 2 times, one as a sgdGene and another as ensGene

# Now plot the features (but only from one range at a time!)
# Notice that with the default parameters each feature will appear with a
# different color
```

```
plotFeatures(features[features$segment.range == 1,])
```

`getDas-functions` *DAS metadata information handling*

Description

Functions to query metadata from the DAS server.

Usage

```
setDasServer(server)
getDasServer()
getDasSource()
getDasDsn()
getDasEntries(source, as.GRanges=TRUE)
getDasTypes(source)
```

Arguments

`server`	URL of the DAS server.
`source`	The data source id to get the metadata from. Be sure this information is consistent with those returned by `getDasSource`. In somes cases, depending on the server, you might need to use the source title, instead of the source id.
`as.GRanges`	For `getDasEntries`, return a `GRanges` object instead of a `data.frame` (if possible).

Details

The functions documented in this man page are used to set the session's DAS server and retrieve metadata from it.

- setDasServer sets the URL of the DAS server to be used.
- getDasServer returns the URL of the DAS server used in this session
- getDasSource returns the id's, the titles and the capabilities of the data sources available in the server
- getDasDsn returns the id's of the dsn data sources available in the server
- getDasEntries returns the entry points of the given data source
- getDasTypes returns the types (a.k.a. tables) of the given data source

Please, be sure to pass the identifiers EXACTLY as the values returned by the functions available here (DAS is case-sensitive). It is also recommended that you check the capabilities for each of the data sources.

For performance reasons, the values in the main calls `getDasFeature`, `getDasSequence`, `getDasStructure` are not validated against the server. If you pass wrong values for `source, ranges or types` it will result in `NULL`, but no error will rise.

Value

For `getDasSever, getDasDsn and getDasTypes` a `character` vector

For `getDasSource` a `data.frame` with the information of the DBs

For `getDasEntries` a GRanges (`as.GRanges=TRUE`) or `data.frame` otherwise

Author(s)

Oscar Flores <oflores@mmb.pcb.ub.es>

Anna Mantsoki <anna.mantsoki@bsc.es>

See Also

`getDasFeature getDasSequence getDasStructure`

Examples

```
# Set session server to EMBL-EBI Genomic DAS Server

setDasServer(server="http://www.ebi.ac.uk/das-srv/genomicdas/das")

print(getDasServer())

sources <- getDasSource()
print(sources)

dsn_sources <- getDasDsn()
print(dsn_sources)

entries <- getDasEntries(sources$id[[1]])
print(entries)

types <- getDasTypes(sources$id[[1]])
print(types)
```

`getDasFeature` *Query for features to session's DAS server*

Description

This function retrieves information for the features for the given ranges and types using the corresponding data source id (or title) in the session's DAS server.

Usage

```
getDasFeature(source, ranges, types)
```

Arguments

`source`	Data source id (or title). See `getDasSource`.
`ranges`	GRanges object containing the seqname(s) and range(s) to query. Be sure that this information is coherent with those returned by `getDasEntries`.
`types`	Names of the features to retrieve. They must match one or more of the values returned by `getDasTypes`.

Details

This function queries the DAS server and returns the available information for the type(s) at the given range(s).

Please, be sure to pass the identifiers EXACTLY as the values returned by the functions available here (DAS is case-sensitive). It is also recommended that you check the capabilities for each of the data sources.

Please, notice that this function codes the parameters in a URL to query the server. This function could fail if you ask for many ranges or types. Also, notice that the speed of the query depends on the number of elements and the load in the server.

Please, use this function smartly and do not overload servers.

Value

`data.frame` with the following columns: `id, start, stop, version, label, type, method, score, orientat` which match the contents of the specific annotation for the given ranges and types in the server (some of the fields are optional, so not all the servers provide information for all the above columns).

If there are not matching results, output is `NULL`.

Author(s)

Oscar Flores <oflores@mmb.pcb.ub.es>

Anna Mantsoki <anna.mantsoki@bsc.es>

See Also

getDas-functions

Examples

```
# Set session server to UniProt DAS Reference Server

setDasServer(server="http://www.ebi.ac.uk/das-srv/uniprot/das")
source <- "uniprot-summary"
ranges <- GRanges(seqnames=c("A0A000","A0A001"), IRanges(start=c(1,1), end=c(394,591)))
types <- c("description", "sequence-summary")

features <- getDasFeature(source, ranges, types)
print(features)
```

`getDasSequence` *Query nucleotide or amino acid sequence to session's DAS server*

Description

This function retrieves the nucleotide or amino acid sequence for the given ranges using the corresponding data source id (or title) in the session's DAS server

Usage

```
getDasSequence(source,ranges, class=c("character", "DNAStringSet", "RNAStringSet", "AAStringSe
```

Arguments

source	Data source id (or title). See getDasSource.
ranges	GRanges object containing the seqname(s) and range(s) to query. Make sure this information is consistent with those returned by getDasEntries.
class	Instance of the class to be used in the output. By default it returns a regular character vector, but we recomend using Biostring classes for retrieving large sequences. Most usual Biosting classes are DNAStringSet for genomic sequences, RNAStringSet for transcripts and AAStringSet for aminoacids.

Details

This function allows to query the DAS server and returns the nucleotide or amino acid sequence available at the given ranges.

Please, be sure to pass the identifiers EXACTLY as the values returned by the functions available here (DAS is case-sensitive). It is also recommended that you check the capabilities for each of the data sources.

Please, notice that this function codes the parameters in a URL to query the server. This function could fail if you ask for many ranges. Also, notice that the speed of the query depends on the number of elements and the load in the server.

Please, use this function smartly and do not overload servers.

Value

Nucleotide or amino acid sequences that match the content of the annotation for the given ranges in the server.

If there are not matching results, output is NULL.

Author(s)

Oscar Flores <oflores@mmb.pcb.ub.es>

Anna Mantsoki <anna.mantsoki@bsc.es>

See Also

getDas-functions

Examples

```
# Set session server to the UniProt DAS Reference Server

setDasServer(server="http://www.ebi.ac.uk/das-srv/uniprot/das")

print(getDasServer())

source <- "uniprot" #Uniprot

ranges <- GRanges(c("A0A000","A0A001"), IRanges(start=c(1,1), end=c(394,591)))

sequences <- getDasSequence(source, ranges)

print(sequences)
```

`getDasStructure` *Query a protein 3D structure to session's DAS server*

Description

This function retrieves the 3D structure,including metadata and coordinates for the given query(ID of the reference structure) using the data source id (or title) in the session's DAS server.

Usage

```
getDasStructure(source, query)
```

Arguments

`source`	Data source id (or title). See `getDasSource`
`query`	Query id

Details

This function allows to query the DAS server and returns the the 3D structure,including metadata and coordinates for the given query (id of the reference structure).

Please, be sure to pass the identifiers EXACTLY as the values returned by the functions available here (DAS is case-sensitive). It is also recommended that you check the capabilities for each of the data sources.

Please, notice that this function codes the parameters in a URL to query the server. This function could fail if you ask for a reference structure that does not exist. Also, notice that the speed of the query depends on the number of elements and the load in the server.

Please, use this function smartly and do not overload servers.

Value

`data.frame` with the following columns: `x,y,z,atomName,atomID, occupancy,tempFactor,type,groupID,name` which match with the contents of the reference structure of the given query.

If there are not matching results, output is `NULL`.

Author(s)

Oscar Flores <oflores@mmb.pcb.ub.es>

Anna Mantsoki <anna.mantsoki@bsc.es>

See Also

`getDasFeature` `getDas-functions` `getDasSequence`

Examples

```
# Set DAS server
setDasServer(server="http://das.cathdb.info/das")

source <- "cath_pdb"
query <- "1hck"  # PDB code

structure <- getDasStructure(source, query)
head(structure)
```

`plotFeatures` *Basic plotting function for obtained features*

Description

This functions creates a basic plot with the features retrieved by `getDasFeature` or adds them to an existing plot.

Usage

```
plotFeatures(df, col=NULL, col.start="start", col.end="end", col.id ="label", col.type="type",
```

Arguments

`df`	`data.frame` with the features. Columns `col.start`, `col.end`, `col.id` and `col.type` must be present.
`col`	Vector of colors for each box in `df`. If `NULL`, automatic rainbow coloring for different types will be used.
`col.start, col.end, col.id, col.type`	Names of the columns of the data.frame from which retrieve the values
`box.height, box.sep`	Height and separation between boxes (in percentual points respect y-axis)
`pos.label`	Position of the label respect the box. If `pos.label=="no"`, labels are disabled.
`new`	Open a new graphical device. If `new=FALSE`, boxes will be added to an existing plot (so, coordinate system must be consistent).
`legend`	Add legend to the plot
`...`	Other graphical parameters passed to `plot.default`

Details

This function provides a quick way to view (non-overlapped) boxes in a new or an existing plot with the features retrieved.

Notice that boxes will can be plotted overlaping the current open graphical device, so the x-coordinates must match.

Value

(none)

Author(s)

Oscar Flores <oflores@mmb.pcb.ub.es>

See Also

getDasFeature

Examples

```
# This is UCSC Genome browser
setDasServer(server="http://genome.ucsc.edu/cgi-bin/das")

# Note that id now changes, we can retrieve it from getDasDsn()
source <- "sacCer3"
range <- GRanges(c("I"), IRanges(start=1, end=2500))
type <- c("sgdGene")

# We want the official genes from 'sdgGene' in the range I:1-2500
features <- getDasFeature(source, range, type)
print(features)

plotFeatures(features)
```

Index