Package 'RefNet'

April 23, 2016

Type Package

Title A queryable collection of molecular interactions, from many sources

Version 1.6.1

Date 2016-02-18

Author Paul Shannon

Maintainer Paul Shannon <paul.thurmond.shannon@gmail.com>

Depends R (>= 2.15.0), methods, IRanges, PSICQUIC, AnnotationHub, RCurl, shiny

Suggests RUnit, BiocStyle, org.Hs.eg.db

Imports BiocGenerics

Description Molecular interactions with metadata, some archived, some dynamically obtained

License Artistic-2.0

LazyLoad yes

biocViews GraphAndNetwork

NeedsCompilation no

R topics documented:

detectDuplicateInteractions
detectDuplicateInteractions

Description

Assign a shared "dupGroup" number to each duplicated interaction, here defined as A.canonical, type, B.canonical. All other information about the interaction is ignored. An A-B interaction is treated the same as a B-A interaction. This assessment prepares a possibly large interaction set for scrutiny by eye, or programmatically, for filtering, in which a single preferred interaction can be selected, out of each dupGroup, for further use.

Usage

```
detectDuplicateInteractions(tbl)
```

Arguments

tbl	a data.frame, returned by interactions method

Value

A data.frame identical to the one passed as an argument, but with one additional column "dupGroup". dupGroup 0 (zero) contains all of the interactions which have NO duplicates.

Author(s)

Paul Shannon

See Also

RefNet, providerClasses, interactions, addStandardNames, pickBestFromDupGroup

Examples

```
filename <- system.file(package="RefNet", "extdata", "tbl.28g2.RData")
load(filename)
tbl.withDupsColumn <- detectDuplicateInteractions(tbl.28g2)
table(tbl.withDupsColumn$dupGroup)
```

interactions *interactions*

Description

This is the primary interface to the RefNet services, in which approximately thirty interaction providers return annotated molecular (often protein-protein) interactions in response to a standard query language ("MIQL" – the molecular interaction query language). This method is thus an R interface to MIQL, with several conveniences added.

In the most typical and basic use, one specifies a species and a gene symbol, and RefNet returns a data.frame listing all of the interactions for that gene (or the protein it encodes) from all RefNet providers, of all interaction types and all detection methods.

More focused queries are easily accomplished: all arguments (execpt for the RefNet object the method dispatches on) are optional, and each specified argument limits the search space: to one or more genes, a publication identifier (typically a pubmed id), interaction types, a detection methods, and/or one or more species.

In principle you could obtain all interactions, of all types, from all providers, by using default values for all of the search-constraining arugments. But this would be very unwise! Highly specific or exploratory queries of modest scope are encouraged, as are bulk downloads using *ftp* directed at the providers' web sites for acquiring large interaction datasets.

Usage

```
## S4 method for signature 'RefNet'
interactions(object,
                                    id=NA,
                                    species=NA,
                                    speciesExclusive=TRUE,
                                    type=NA,
                                    provider=NA,
                                    detectionMethod=NA,
                                    publicationID=NA,
                                    quiet=TRUE)
```

Arguments

object	a RefNet object.
id	one or more character strings. These are most often gene symbols (e.g., "RAD17"), to which most of the RefNet providers respond, translating these into the native identifiers of their data store, which is typically a protein identifier of one sort or another. You may also provide such a protein identifier directly (e.g., uniprot "O75943", or refseq "NP_002684"). Protein identifiers typically return fewer interactions, presumably because providers do not translate them into their native identifiers, and only a subset of the providers find matches. If multiple ids are supplied, then only interactions which occur between any two of the ids is returned. If only one id is supplied, all interactions are reported in which that id participates.

species	One or more character strings. Use an NCBI taxon code, for which the speciesIds method is useful. Note the next argument.
speciesExclusive	a logical variable, default TRUE. If set to FALSE than interactions will be included with proteins from species other than those explicitly named. Infection and transgenic experiments are examples of cross-species protein-protein interactions. We anticipate that the most common query seeks interactions among proteins in the same cell of the same organism. For this reason, this argument defaults to TRUE.
type	a list of character strings. See the method interactionTypes.
provider	a list of character strings. See the method providers for the currently available set. This argument defaults to NA, and all currently live providers are queried.
detectionMethod	a list of character strings. See the method detectionMethods
publicationID	a list of character strings. Usually a pubmedID, but sometimes an OMIM reference number.
quiet	a logical of character strings, default TRUE. Set to FALSE if you wish to trace the exection of your query against all providers, and explore the details of the REST API.

Value

A data.frame with 16 columns, and one row for every interaction, described as an annotated A/B relationship, with self-describing column names.

Author(s)

Paul Shannon

See Also

providers, addStandardNames, IDMapper

Examples

```
refnet <- RefNet()
providers(refnet)
    # query all providers for all known Myc interactions
tbl.1 <- interactions(refnet, id="Myc", species="9606",
                      provider="gerstein-2012")

    # all Myc/EP400 interactions known to BioGrid
    # make sure that BioGrid is currently available

if("BioGrid" %in% providers(refnet))
   tbl.2 <- interactions(refnet, c("Myc", "EP400"), species="9606",
                         provider="BioGrid")
```

```
    # or those between Myc and any other molecule, detected by
    # the "pull down" proteomics technique and judged to be
    # a "direct interaction"

# recon2 temporarily unavailable.  will be added to AnnotationHub soon
# tbl.3 <- interactions(refnet, "L-alanine transaminase",  provider="recon2")
# tbl.3[, c("A.common", "type", "B.common", "A.canonical", "B.canonical")]
```

---

pickBestFromDupGroup *pickBestFromDupGroup*

Description

Assign a shared "dupGroup" number to each duplicated interaction, here defined as A.canonical, type, B.canonical. All other information about the interaction is ignored. An A-B interaction is treated the same as a B-A interaction. This assessment prepares a possibly large interaction set for scrutiny by eye, or programmatically, for filtering, in which a single preferred interaction can be selected, out of each dupGroup, for further use.

Note that preferred.interaction.types elements may be substrings of the full (and often unwieldy) interaction type names used in the underlying data.

Usage

```
pickBestFromDupGroup(dupGrp, tbl.dups,
                     preferred.interaction.types)
```

Arguments

dupGrp	an integer
tbl.dups	data.frame, returned by detectDuplicateInteractions.
preferred.interaction.types	list of character strings

Value

The row name (or names, for dupGroup 0) of the tbl.dups row which is the best match to the ordered list of preferred.interaction.types.

Author(s)

Paul Shannon

See Also

RefNet, providerClasses, interactions, addStandardNames, detectDuplicateInteractions

Examples

```
filename <- system.file(package="RefNet", "extdata", "tbl.dups.RData")
load(filename)
preferred.types <- c("direct", "physical", "aggregation")

  # get the best from dupGroup 1
best.1 <- pickBestFromDupGroup(1, tbl.dups, preferred.types)
tbl.dups[best.1, c("A.common", "B.common", "type", "provider", "publicationID")]

  # get all of the best.  not every dupGroup will pass muster
dupGroups <- sort(unique(tbl.dups$dupGroup))
bestOfDups <- unlist(lapply(dupGroups, function(dupGroup)
                  pickBestFromDupGroup(dupGroup, tbl.dups, preferred.types)))
deleters <- which(is.na(bestOfDups))
if(length(deleters) > 0)
    bestOfDups <- bestOfDups[-deleters]
length(bestOfDups)
tbl.dups[bestOfDups, c("A.common", "B.common", "type", "provider", "publicationID")]
```

---

providerClasses *providerClasses*

Description

RefNet providerClasses are the names of the groups of available data sources, currently "PSIC-QUIC" and "native"

Usage

```
## S4 method for signature 'RefNet'
providerClasses(object)
```

Arguments

object	a RefNet object.

Value

This method returns a list of the current groups.

Author(s)

Paul Shannon

See Also

RefNet, interactions, addStandardNames, IDMapper

Examples

```
refnet <- RefNet()
providerClasses(refnet)
```

---

providers *providers*

Description

RefNet providers are the names of data sources, many from PSICQUIC, some from RefNet.db.

Usage

```
## S4 method for signature 'RefNet'
providers(object)
```

Arguments

object	a RefNet object.

Value

This method returns a named list of these data sources, arrayed in two named groups, "PSICQUIC" and "native".

Author(s)

Paul Shannon

See Also

RefNet, providerClasses, interactions, addStandardNames, IDMapper

Examples

```
refnet <- RefNet()
providers(refnet)
```

pubmedAbstract *pubmedAbstract*

Description

Get the text of a journal article's abstract from PubMed.

Usage

```
pubmedAbstract(pmid, split=TRUE)
```

Arguments

pmid	A character string, the pubmedID.
split	Whether or not to split the text on embedded newlines.

Value

A single character string, or a vector of strings split on the embedded newline characters.

Author(s)

Paul Shannon

See Also

RefNet, providerClasses, interactions, addStandardNames

Examples

```
pubmedAbstract("22959076")
```

RefNet-class *RefNet*

Description

A query interface to a large collecion of molecular interactions.

Constructor

RefNet: establishes connectin to the central PSICQUIC web server, and loads all "native" data sources.

Methods

`providers(x)`: lists short names of the data providers

`providerClasses(x)`: lists the names of logical groupings of the providers.

`interactions(x,id, species, speciesExclusive, type, provider,detectionMethod, p` retrieves all interactions matching the specified pattern.

`providerClasses(x)`: lists the names of logical groupings of the providers.

`show(x)`: displays current providers and related data

Functions

`detectDuplicateInteractions(tbl)`: adds a column grouping all interactions by their two participants. The 0th dupGroup is all singletons.

`pickBestFromDupGroup(dupGrr, tbl.dups, preferred.interaction.types)`: matches, in order, the preferred types against the interactions types, returning tbl.dups rowname for best match.

`pubmedAbstract(pmid, split=FALSE)`: returns the text of the pubmed abstract

Author(s)

Paul Shannon

See Also

PSICQUIC, interactions, providerClasses, providers, pubmedAbstract, detectDuplicateInteractions, pickBestFromDupGroup

Examples

```
  # List the sources
refNet <- RefNet()
show(refNet)
```

Index