Package 'EuPathDB'

November 11, 2020

Title Provides access to pathogen annotation resources available on EuPathDB databases

Version 1.6.0

Author Keith Hughitt, Ashton Trey Belew

Maintainer Keith Hughitt <khughitt@umd.edu>

Description Brings together annotation resources from the various EuPathDB databases (PlasmoDB, ToxoDB, TriTrypDB, etc.) and makes them available in R using the AnnotationHub framework.

Depends R (>= 3.5),
GenomeInfoDbData,

Imports AnnotationHub, AnnotationHubData,
Biobase, Biostrings, BiocGenerics,
data.table, dplyr,
foreach,
GenomeInfoDb, GenomicRanges, glue,
httr,
jsonlite,
magrittr,
readr, rtracklayer, rvest,
utils,
xml2

Suggests AnnotationDbi, AnnotationForge,
BiocManager, BiocStyle, BSgenome,
curl,
desc, devtools,
GenomicFeatures, GO.db,
KEGGREST, knitr,
OrganismDbi,
RCurl, reactome.db, RSQLite,
S4Vectors, stringr, testthat, tidyr

biocViews AnnotationData, AnnotationHub, DataImport, EuPathDB

License Artistic-2.0

URL https://github.com/khughitt/EuPathDB

BugReports https://github.com/khughitt/EuPathDB/issues

RoxygenNote 7.1.1

VignetteBuilder knitr

Collate 'check_csv.R'
'check_files.R'
'check_s3.R'
'clean_pkg.R'
'copy_s3_file.R'
'download_eupath_metadata.R'
'download_uniprot_annotations.R'
'eupathdb.R'
'expand_list_columns.R'
'extract_eupath_orthologs.R'
'extract_gene_locations.R'
'get_all_metadata.R'
'get_eupath_annotation_columns.R'
'get_eupath_entry.R'
'get_eupath_fields.R'
'get_eupath_pkgnames.R'
'get_kegg_orgn.R'
'kegg_vector_to_df.R'
'get_versions.R'
'load_ah_annotations.R'
'load_eupath_annotations.R'
'load_eupath_go.R'
'load_kegg_annotations.R'
'load_orgdb_annotations.R'
'load_orgdb_go.R'
'logging.R'
'make_eupath_bsgenome.R'
'make_eupath_granges.R'
'make_eupath_organismdbi.R'
'make_eupath_orgdb.R'
'make_eupath_txdb.R'
'make_taxon_names.R'
'move_final_package.R'
'orgdb_from_ah.R'
'post_eupath_annotations.R'
'post_eupath_go.R'
'post_eupath_goslim.R'
'post_eupath_interpro.R'
'post_eupath_linkout.R'
'post_eupath_ortholog.R'
'post_eupath_pathway.R'
'post_eupath_pdb.R'
'post_eupath_pubmed.R'
'post_eupath_table.R'
'prefix_map.R'

'query_s3_file.R'
'query_s3_ah.R'
'write_eupath_metadata.R'
'xref_species.R'
'xref_taxonomy.R'

R topics documented:

`%:::%` *R CMD check is super annoying about :::.*

Description

In a fit of pique, I did a google search to see if anyone else has been annoyed in the same way as I. I was not surprised to see that Yihui Xie was, and in his email to r-devel in 2013 he proposed a game of hide-and-seek; a game which I am repeating here.

Usage

```
pkg %:::% fun
```

Arguments

`pkg`	on the left hand side
`fun`	on the right hand side

Details

This just implements ::: as an infix operator that will not trip check.

check_csv *Check the metadata csv files and write only the 'good' entries.*

Description

While we are at it, put the failed entries into their own csv file so that I can step through and look for why they failed.

Usage

```
check_csv(file_type = "OrgDb", bioc_version = NULL, eu_version = NULL)
```

Arguments

file_type	Is this an OrgDB, GRanges, TxDb, OrganismDbi, or BSGenome dataset?
bioc_version	Which bioconductor version is this for?
eu_version	Which eupathdb version is this for?

check_files *List the directory containing the various sqlite files and make sure they all have entries.*

Description

Any files which do not have csv entries should be deleted, but for the moment I will move them to the current working directory in an attempt to learn about why they went wrong.

Usage

```
check_files(
  file_type = "OrgDb",
  bioc_version = NULL,
  eu_version = NULL,
  verbose = FALSE,
  destination = NULL
)
```

Arguments

file_type	Is this an OrgDB, GRanges, TxDb, OrganismDbi, or BSGenome dataset?
bioc_version	Which bioconductor version is this for?
eu_version	Which eupathdb version is this for?
verbose	Talk while running?
destination	Place to put non-matched files.

check_s3 *Check the metadata csv files and write only the 'good' entries.*

Description

While we are at it, put the failed entries into their own csv file so that I can step through and look for why they failed.

Usage

```
check_s3(file_type = "OrgDb", bioc_version = NULL, eu_version = NULL)
```

Arguments

file_type	Is this an OrgDB, GRanges, TxDb, OrganismDbi, or BSGenome dataset?
bioc_version	Which bioconductor version is this for?
eu_version	Which eupathdb version is this for?

clean_pkg *Cleans up illegal characters in packages generated by make_organismdbi(), make_orgdb(), and make_txdb(). This attempts to fix some of the common problems therein.*

Description

The primary problem this function seeks to solve is derived from the fact that some species names in the eupathdb contain characters which are not allowed in orgdb/txdb/organismdbi instances. Thus this invokes a couple of regular expressions in an attempt to make sure these generated packages are actually installable.

Usage

```
clean_pkg(path, removal = "-like", replace = "", sqlite = TRUE)
```

Arguments

path	Location for the original Db/Dbi instance.
removal	String to remove from the instance.
replace	What to replace removal with, when necessary.
sqlite	Also modify the sqlite database?

Details

One thing I should consider is to add some of this logic to my eupath queries rather than perform these clunky modifications to the already-generated packages.

Value

A hopefully cleaner OrgDb/TxDb/OrganismDbi sqlite package.

`copy_s3_file` *Copy the relevant file for each data type into a place which is easy for pickup by s3.*

Description

Copy the relevant file for each data type into a place which is easy for pickup by s3.

Usage

```
copy_s3_file(src_dir, s3_file, type = "bsgenome")
```

Arguments

`src_dir`	Source directory for the package top be copied.
`s3_file`	Where is the final file to be located?
`type`	Which type of package is this?

`download_eupath_metadata` *Returns metadata for all eupathdb organisms.*

Description

Returns metadata for all eupathdb organisms.

Usage

```
download_eupath_metadata(
  overwrite = FALSE,
  webservice = "eupathdb",
  bioc_version = NULL,
  dir = "EuPathDB",
  eu_version = NULL,
  write_csv = FALSE,
  verbose = FALSE
)
```

Arguments

overwrite	Overwrite existing data?
webservice	Optional alternative webservice for hard-to-find species.
bioc_version	Manually set the bioconductor release if desired.
dir	Where to put the json.
eu_version	Choose a specific eupathdb version?
write_csv	Write a csv file in the format expected by AnnotationHubData?
verbose	Print helper message about species matching?
limit_n	Maximum number of valid entries to return.

Value

Dataframe with lots of rows for the various species in eupathdb.

Author(s)

Keith Hughitt

---

download_uniprot_annotations

Make a table of uniprot anntotations when possible.

Description

I have been working on getting stupid v42 eupathdb into annotationhub forever, it is not particularly interesting to me, and for reasons passing all understanding, it seems impossible to get it to work properly for Lori. In the mean time, I had some ideas of fun things to do.

Usage

```
download_uniprot_annotations(gids, entry, dir = "EuPathDB", overwrite = FALSE)
```

Arguments

gids	Gene IDs to cross reference against uniprot.
entry	Eupath entry to cross reference.
dir	Working directory.
overwrite	Overwrite the savefile.

Details

Notably, my other package 'hpgltools' has a nifty uniprot downloader/parser. It may reasonably easily be used to bring into an orgdb the set of uniprot annotations.

This is just a silly initial implementation because I have been sipping on whisky. But I think it gets the idea across.

`download_uniprot_proteome` *Download the txt uniprot data for a given accession/species*

Description

Download the txt uniprot data for a given accession/species

Usage

```
download_uniprot_proteome(
  accession = NULL,
  species = NULL,
  taxonomy = NULL,
  all = FALSE,
  first = FALSE
)
```

Arguments

`accession`	Which accession to grab?
`species`	Or perhaps species?
`taxonomy`	Taxon to query.
`all`	If there are more than 1 hit, grab them all?
`first`	Or perhaps just grab the first hit?

Value

A filename/accession tuple.

`error` *Error-level logging function.*

Description

Error-level logging function.

Usage

```
error(...)
```

Arguments

`...`	One or more strings to be logged.

EuPathDB *EuPathDB: Access EuPathDB annotations using AnnotationHub*

Description

EuPathDB provides an R interface for retrieving annotation resources from the EuPathDB databases: AmoebaDB, CryptoDB, FungiDB, GiardiaDB, MicrosporidiaDB, PiroplasmaDB, PlasmoDB, ToxoDB, TrichDB, and TriTrypDB using the Bioconductor AnnotationHub framework.

Details

There are currently two types of Bioconductor resources which can be retrieved for 194 supported organisms from the various EuPathDB databases:

- OrgDB resources
- GRanges resources

The OrgDB resources provides gene level information including chromosome, location, name, description, orthologs, and associated GO terms.

The GRanges resources provide transcript-level information such as known exons and their corresponding locations.

Each of these resources are generated using information obtained from the EuPathDB GFF files along with queries made through the various EuPathDB web APIs.

For examples of how EuPathDB can be used to query and interact with EuPathDB.org resources, take a look at the vignette: `browseVignettes(package="EuPathDB")`

Use `availableEuPathDB()` to get a vector of available organisms.

Author(s)

Keith Hughitt and Ashton Belew

See Also

AnnotationHub

GRanges

http://eupathdb.org/eupathdb/

`extract_eupath_orthologs` *Given 2 species names from the eupathdb, make orthology tables betwixt them.*

Description

The eupathdb provides such a tremendous wealth of information. For me though, it is difficult sometimes to boil it down into just the bits of comparison I want for 1 species or between 2 species. A singularly common question I am asked is: "What are the most similar genes between species x and y among these two arbitrary parasites?" There are lots of ways to poke at this question: run BLAST/fasta36, use biomart, query the ortholog tables from the eupathdb, etc. However, in all these cases, it is not trivial to ask the next question: What about: a:b and b:a? This function attempts to address that for the case of two eupath species from the same domain. (tritrypdb/fungidb/etc.) It does however assume that the sqlite package has been installed locally, if not it suggests you run the make_organismdbi function in order to do that.

Usage

```
extract_eupath_orthologs(
  db,
  master = "GID",
  query_species = NULL,
  id_column = "ORTHOLOGS_GID",
  org_column = "ORTHOLOGS_ORGANISM",
  group_column = "ORTHOLOGS_GROUP_ID",
  name_column = "ORTHOLOGS_PRODUCT",
  count_column = "ORTHOLOGS_COUNT",
  print_speciesnames = FALSE,
  webservice = "eupathdb"
)
```

Arguments

`db`	Species name (subset) from one eupath database.
`master`	Primary keytype to use for indexing the various tables.
`query_species`	A list of exact species names to search for. If uncertain about them, add print_speciesnames=TRUE and be ready for a big blob of text. If left null, then it will pull all species.
`id_column`	What column in the database provides the set of ortholog IDs?
`org_column`	What column provides the species name?
`count_column`	Name of the column with the count of species represented.
`print_speciesnames`	Dump the species names for diagnostics?
`webservice`	Which eupathdb project to query?
`url_column`	What column provides the orthomcl group ID?

Details

One other important caveat: this function assumes queries in the format 'table_column' where in this particular instance, the table is further assumed to be the ortholog table.

Value

A big table of orthoMCL families, the columns are:

1. GID: The gene ID
2. ORTHOLOG_ID: The gene ID of the associated ortholog.
3. ORTHOLOG_SPECIES: The species of the associated ortholog.
4. ORTHOLOG_URL: The OrthoMCL group ID's URL.
5. ORTHOLOG_COUNT: The number of all genes from all species represented in this group.
6. ORTHOLOG_GROUP: The family ID
7. QUERIES_IN_GROUP: How many of the query species are represented in this group?
8. GROUP_REPRESENTATION: ORTHOLOG_COUNT / the number of possible species.

Author(s)

atb

---

extract_gene_locations

Clean up the gene location field from eupathdb derived gene location data.

Description

The eupathdb encodes its location data for genes in a somewhat peculiar format: chromosome:start..end(strand), but I would prefer to have these snippets of information as separate columns so that I can do things like trivially perform rpkm().

Usage

```
extract_gene_locations(annot_df, location_column = "annot_gene_location_text")
```

Arguments

annot_df	Data frame resulting from load_orgdb_annotations()
location_column	Name of the column to extract the start/end/length/etc from.

Value

Somewhat nicer data frame.

Author(s)

atb

`get_all_metadata` *Invoke download_eupathdb_metadata() using all the sub-projects of the EuPathDB.*

Description

This just iterates over a list of existing EuPathDB web resources and attempts to download the metadata from them.

Usage

```
get_all_metadata(webservice = "all")
```

Arguments

`webservice`	Assume all services are desired.

Value

Dataframe of the various species metadata.

`get_eupath_entry` *Search the eupathdb metadata for a given species substring.*

Description

When querying the eupathdb, it can be difficult to hit the desired species. This is confounded by the fact that there are very similar named species across different EupathDB projects. Thus function seeks to make it a bit easier to find the actual dataset desired. If the specific species is not found, look for a reasonable approximation. stop() if nothing is found.

Usage

```
get_eupath_entry(
  species = "Leishmania major",
  webservice = "eupathdb",
  column = "TaxonUnmodified",
  metadata = NULL,
  ...
)
```

Arguments

species	String containing some reasonably unique text in the desired species name.
webservice	The EuPathDB webservice to query.
column	Which column to use for getting the species name?
metadata	Optional dataframe of already downloaded metadata.
...	Parameters passed to download_eupath_metadata()

Value

A single row from the eupathdb metadata.

Author(s)

atb

---

get_eupath_fields *Extract query-able fields from the EupathDb.*

Description

This parses the result of a query to Eupath's webservice: 'GenesByMolecularWeight' and uses it to get a list of fields which are acquireable elsewhere.

Usage

```
get_eupath_fields(webservice, excludes = NULL)
```

Arguments

webservice	Eupathdb, tritrypdb, fungidb, etc...
excludes	List of fields to ignore.

Value

List of parameters.

get_eupath_gene_types *Attempt to get a list of sequence types.*

Description

Attempt to get a list of sequence types.

Usage

```
get_eupath_gene_types(webservice = NULL)
```

Arguments

webservice	choose a service to download from.

get_eupath_pkgnames *Generate standardized package names for the various eupathdb species.*

Description

This is a surprisingly difficult problem. Many species names in the eupathdb have odd characters in the species suffix which defines the strain ID. Many of these peculiarities result in packages which are non-viable for installation. Thus this function attempts to filter them out and result in consistent, valid package names. They are not exactly the same in format as other orgdb/txdb/etc packages, as I include in them a field for the eupathdb version used; but otherwise they should be familiar to any user of the sqlite based organism packages.

Usage

```
get_eupath_pkgnames(entry, eu_version = NULL, column = "TaxonUnmodified")
```

Arguments

entry	A metadatum entry.
eu_version	Choose a specific version of the eupathdb, only really useful when downloading files.
column	Which column to query to get the species name?

Details

The default argument for this function shows the funniest one I have found so far thanks to the hash character in the strain definition.

Value

List of package names and some booleans to see if they have already been installed.

Author(s)

atb

`get_kegg_orgn` *Search KEGG identifiers for a given species name.*

Description

KEGG identifiers do not always make sense. For example, how am I supposed to remember that Leishmania major is lmj? This takes in a human readable string and finds the KEGG identifiers that match it.

Usage

```
get_kegg_orgn(species = "Leishmania", short = TRUE)
```

Arguments

`species`	Search string (Something like 'Homo sapiens').
`short`	Only pull the orgid?

Value

Data frame of possible KEGG identifier codes, genome ID numbers, species, and phylogenetic classifications.

See Also

RCurl

Examples

```
## Not run:
 fun = get_kegg_orgn('Canis')
 ## >     Tid     orgid      species                   phylogeny
 ## >  17 T01007   cfa Canis familiaris (dog) Eukaryotes;Animals;Vertebrates;Mammals

## End(Not run)
```

`get_versions` *Figure out the current bioconductor release and eupathdb.org release version numbers.*

Description

Figure out the current bioconductor release and eupathdb.org release version numbers.

Usage

```
get_versions(eu_version = NULL, bioc_version = NULL)
```

Arguments

`eu_version`	When null, query tritrypdb to find the current release version number.
`bioc_version`	When null, ask BiocManager for the current bioconductor release.

`info` *Info-level logging function.*

Description

Info-level logging function.

Usage

```
info(...)
```

Arguments

`...`	One or more strings to be logged.

`kegg_vector_to_df` *Convert a potentially non-unique vector from kegg into a normalized data frame.*

Description

This function seeks to reformat data from KEGGREST into something which is rather easier to use.

Usage

```
kegg_vector_to_df(vector, final_colname = "first", flatten = TRUE)
```

Arguments

`vector`	Information from KEGGREST
`final_colname`	Column name for the new information
`flatten`	Flatten nested data?

Details

This could probably benefit from a tidyr-ish revisitation.

Value

A normalized data frame of gene IDs to whatever.

Author(s)

atb

`load_ah_annotations` *Shortcut for loading annotation data from AnnotationHub, making some EupathDB assumptions.*

Description

Shortcut for loading annotation data from AnnotationHub, making some EupathDB assumptions.

Usage

```
load_ah_annotations(
  species = "Leishmania major strain Friedlin",
  service = "TriTrypDB",
  type = "OrgDb",
  eu_version = NULL,
  wanted_fields = NULL
)
```

Arguments

species	String containing a unique portion of the desired species.
service	Which eupath webservice is desired?
type	Data type to load.
eu_version	Gather data from a specific eupathdb version?
wanted_fields	If not provided, this will gather all columns starting with 'annot'.

Value

Big huge data frame of annotation data.

---

load_eupath_annotations

Shortcut for loading annotation data from a eupathdb-based orgdb.

Description

Every time I go to load the annotation data from an orgdb for a parasite, it takes me an annoyingly long time to get the darn flags right. As a result I wrote this to shortcut that process. Ideally, one should only need to pass it a species name and get out a nice big table of annotation data.

Usage

```
load_eupath_annotations(
  query,
  webservice = "tritrypdb",
  eu_version = NULL,
  wanted_fields = NULL
)
```

Arguments

webservice	Which eupath webservice is desired?
eu_version	Gather data from a specific eupathdb version?
wanted_fields	If not provided, this will gather all columns starting with 'annot'.
species	String containing a unique portion of the desired species.

Value

Big huge data frame of annotation data.

load_eupath_go *Shortcut for loading annotation data from a eupathdb-based orgdb.*

Description

Every time I go to load the annotation data from an orgdb for a parasite, it takes me an annoyingly long time to get the darn flags right. As a result I wrote this to shortcut that process. Ideally, one should only need to pass it a species name and get out a nice big table of annotation data.

Usage

```
load_eupath_go(
  query,
  webservice = "tritrypdb",
  eu_version = NULL,
  wanted_fields = NULL,
  gene_ids = NULL,
  columns = c("go", "evidence")
)
```

Arguments

webservice	Which eupath webservice is desired?
eu_version	Gather data from a specific eupathdb version?
wanted_fields	If not provided, this will gather all columns starting with 'annot'.
species	String containing a unique portion of the desired species.

Value

Big huge data frame of annotation data.

load_eupath_pkg *Loads a pkg into the current R environment.*

Description

Loads a pkg into the current R environment.

Usage

```
load_eupath_pkg(name, webservice = "eupathdb")
```

load_kegg_annotations *Create a data frame of pathways to gene IDs from KEGGREST*

Description

This seeks to take the peculiar format from KEGGREST for pathway<->genes and make it easier to deal with.

Usage

```
load_kegg_annotations(species = "coli", abbreviation = NULL, flatten = TRUE)
```

Arguments

species	String to use to query KEGG abbreviation.
abbreviation	If you already know the abbreviation, use it.
flatten	Flatten nested tables?

Value

dataframe with rows of KEGG gene IDs and columns of NCBI gene IDs and KEGG paths.

Author(s)

atb

load_orgdb_annotations *Load organism annotation data from an orgdb sqlite package.*

Description

Creates a dataframe gene and transcript information for a given set of gene ids using the AnnotationDbi interface.

Usage

```
load_orgdb_annotations(
  orgdb = NULL,
  gene_ids = NULL,
  include_go = FALSE,
  keytype = "gid",
  location_column = "annot_location_text",
  type_column = "annot_gene_type",
  name_column = "annot_gene_product",
  fields = NULL,
  sum_exon_widths = FALSE
)
```

Arguments

orgdb	OrganismDb instance.
gene_ids	Search for a specific set of genes?
include_go	Ask the Dbi for gene ontology information?
keytype	Primary key of the tables, 'gid' for EuPathDB data.
location_column	Which column contains the location data for the genes?
type_column	Use this column to identify the gene type.
name_column	Use this column to identify the gene name.
fields	Columns included in the output.
sum_exon_widths	Perform a sum of the exons in the data set?

Details

This defaults to a few fields which I have found most useful, but the brave or pathological can pass it 'all'.

Value

Table of geneids, chromosomes, descriptions, strands, types, and lengths.

Author(s)

atb

See Also

AnnotationDbi GenomicFeatures BiocGenerics columns keytypes select exonsBy

Examples

```
## Not run:
 one_gene <- load_orgdb_annotations(org, c("LmJF.01.0010"))

## End(Not run)
```

load_orgdb_go *Retrieve GO terms associated with a set of genes.*

Description

AnnotationDbi provides a reasonably complete set of GO mappings between gene ID and ontologies. This will extract that table for a given set of gene IDs.

Usage

```
load_orgdb_go(orgdb = NULL)
```

Arguments

orgdb	OrganismDb instance.
gene_ids	Identifiers of the genes to retrieve annotations.
keytype	Which column is the master key? (GID)
columns	The set of columns to request.

Details

This is a nice way to extract GO data primarily because the Orgdb data sets are extremely fast and flexible, thus by changing the keytype argument, one may use a lot of different ID types and still score some useful ontology data.

Value

Data frame of gene IDs, go terms, and names.

Author(s)

I think Keith provided the initial implementation of this, but atb messed with it pretty extensively.

See Also

AnnotationDbi GO.db magrittr select tbl_df

Examples

```
## Not run:
 go_terms <- load_go_terms(org, c("a","b"))

## End(Not run)
```

load_uniprot_annotations *Read a uniprot text file and extract as much information from it as possible.*

Description

I spent entirely too long fighting with Uniprot.ws, finally got mad and wrote this.

Usage

```
load_uniprot_annotations(file = NULL, savefile = TRUE)
```

Arguments

file	Uniprot file to read and parse
savefile	Do a save?

Value

Big dataframe of annotation data.

make_eupath_bsgenome *Generate a BSgenome package from the eupathdb.*

Description

Since we go to the trouble to try and generate nice orgdb/txdb/organismdbi packages, it seems to me that we ought to also be able to make a readable genome package. I should probably use some of the logic from this to make the organismdbi generator smarter.

Usage

```
make_eupath_bsgenome(
  entry,
  eu_version = NULL,
  workdir = "EuPathDB",
  copy_s3 = FALSE,
  installp = TRUE,
  reinstall = FALSE,
  ...
)
```

Arguments

entry	Single eupathdb metadata entry.
eu_version	Which version of the eupathdb to use for creating the BSGenome?
workdir	Working directory.
copy_s3	Copy the 2bit file into an s3 staging directory for copying to AnnotationHub?
installp	Install the resulting package?
reinstall	Rewrite an existing package directory.
...	Extra arguments for downloading metadata when not provided.

Value

List of package names generated (only 1).

Author(s)

atb

---

make_eupath_granges *Generate a GRanges rda savefile from a gff file.*

Description

There is not too much else to say. This uses import.gff from rtracklayer. I should probably steal my code from hpgltools to make this work for any version of a gff file, but the eupathdb is good about keeping consistent on this front.

Usage

```
make_eupath_granges(
  entry,
  workdir = "EuPathDB",
  eu_version = NULL,
  copy_s3 = FALSE
)
```

Arguments

entry	Metadatum entry.
workdir	Place to put the resulting file(s).
eu_version	Optionally request a specific version of the gff file.
copy_s3	Copy the 2bit file into an s3 staging directory for copying to AnnotationHub?

make_eupath_organismdbi *Create an organismDbi instance for an eupathdb organism.*

Description

The primary goal of an organismdbi instance is to provide a series of links between an orgdb, txdb, and other relevant annotation packages (reactome/go/etc). In its current iteration, this function brings together a couple columns from the orgdb, txdb, GO.db, and reactome.db.

Usage

```
make_eupath_organismdbi(
  entry = NULL,
  eu_version = NULL,
  workdir = "EuPathDB",
  installp = TRUE,
  reinstall = FALSE,
  kegg_abbreviation = NULL,
  exclude_join = "ENTREZID",
  copy_s3 = FALSE
)
```

Arguments

entry	A row from the eupathdb metadataframe.
eu_version	Which version of the eupathdb to use for creating this package?
workdir	Directory in which to build the packages.
installp	Install the resulting package?
reinstall	Overwrite existing data files?
kegg_abbreviation	For when we cannot automagically find the kegg species id.
exclude_join	I had a harebrained idea to automatically set up the joins between columns of GO.db/reactome.db/orgdb/txdb objects. This variable is intended to exclude columns with common IDs that might multi-match spuriously – I think in the end I killed the idea though, perhaps this should be removed or resurrected.
copy_s3	Copy the 2bit file into an s3 staging directory for copying to AnnotationHub?

Value

The result of attempting to install the organismDbi package.

Author(s)

Keith Hughitt, modified by atb.

make_eupath_orgdb *Create an orgdb SQLite database from the tables in eupathdb.*

Description

This function has passed through multiple iterations as the preferred method(s) for accessing data in the eupathdb has changed. It currently uses my empirically defined set of queries against the eupathdb webservices. As a result, I have made some admittedly bizarre choices when creating the queries. Check through eupath_webservices.r for some amusing examples of how I have gotten around the idiosyncrasies in the eupathdb. Final note, I confirmed with Cristina that it is not possible to acquire data specific to a given version of the eupathdb.

Usage

```
make_eupath_orgdb(
  entry,
  build_dir = "EuPathDB",
  install = TRUE,
  kegg_abbreviation = NULL,
  reinstall = FALSE,
  overwrite = FALSE,
  verbose = FALSE,
  copy_s3 = FALSE,
  godb_source = NULL
)
```

Arguments

entry	If not provided, then species will get this, it contains all the information.
build_dir	Where to put all the various temporary files.
install	Install the resulting package?
kegg_abbreviation	If known, provide the kegg abbreviation.
reinstall	Re-install an already existing orgdb?
overwrite	Overwrite a partial installation?
copy_s3	Copy the 2bit file into an s3 staging directory for copying to AnnotationHub?
godb_source	Which table to use for the putative union of the GO tables.

Value

Currently only the name of the installed package. This should probably change.

Author(s)

Keith Hughitt with significant modifications by atb.

`make_eupath_txdb` *Generate an EuPathDB organism TxDb package.*

Description

This will hopefully create a txdb package and granges savefile for a single species in the eupathdb. This depends pretty much entirely on the successful download of a gff file from the eupathdb.

Usage

```
make_eupath_txdb(
  entry = NULL,
  workdir = "EuPathDB",
  eu_version = NULL,
  reinstall = FALSE,
  installp = TRUE,
  copy_s3 = FALSE
)
```

Arguments

`entry`	One row from the organism metadata.
`workdir`	Base directory for building the package.
`eu_version`	Which version of the eupathdb to use for creating this package?
`reinstall`	Overwrite an existing installed package?
`installp`	Install the resulting package?
`copy_s3`	Copy the 2bit file into an s3 staging directory for copying to AnnotationHub?

Value

TxDb instance name.

Author(s)

Keith Hughitt with significant modifications by atb.

make_taxon_names *Iterate through the various ways of representing taxon names*

Description

Spend some time making sure they are valid, too. Thus we want to get rid of weird characters like hash marks, pipes, etc.

Usage

```
make_taxon_names(entry, column = "TaxonUnmodified")
```

Arguments

entry	An entry of the eupathdb metadata.
column	Which column should be used to query the species name?

Value

A list of hopefully valid nomenclature names to be used elsewhere in this family.

Author(s)

atb

move_final_package *Move a package file to its final location for collation by AnnotationHubData.*

Description

Move a package file to its final location for collation by AnnotationHubData.

Usage

```
move_final_package(pkgname, type = "orgdb", workdir = "EuPathDB")
```

Arguments

pkgname	Name of package to move to its final home.
type	Which type of package is this?
workdir	base working directory.

orgdb_from_ah *Get an orgdb from an AnnotationHub taxonID.*

Description

Ideally, annotationhub will one day provide a one-stop shopping source for a tremendous wealth of curated annotation databases, sort of like a non-obnoxious biomart. But for the moment, this function is more fragile than I would like.

Usage

```
orgdb_from_ah(ahid = NULL, title = NULL, species = NULL, type = "OrgDb")
```

Arguments

ahid	TaxonID from AnnotationHub
title	Title for the annotation hub instance
species	Species to download
type	Datatype to download

Value

An Orgdb instance

Author(s)

atb

See Also

AnnotationHub S4Vectors

Examples

```
## Not run:
 orgdbi <- mytaxIdToOrgDb(taxid)

## End(Not run)
```

post_eupath_annotations *Create a series of POST requests which download all the annotation data for a species.*

Description

The only way I have figured out how to download mass data from the eupathdb is to ask for a raw dump of all available data using the GenesByGeneType WADL. Therefore, this function iterates over the various sequence types that I have noticed at the eupathdb and does that for each type.

Usage

```
post_eupath_annotations(
  entry = NULL,
  overwrite = FALSE,
  build_dir = "EuPathDB"
)
```

Arguments

entry	Eupathdb annotation entry.
overwrite	Overwrite existing data if it exists?
workdir	Location to dump the resulting data.

post_eupath_go_table *Use the POST interface to get GO data from the EuPathDB.*

Description

Use the POST interface to get GO data from the EuPathDB.

Usage

```
post_eupath_go_table(entry = NULL, build_dir = "EuPathDB", overwrite = FALSE)
```

Arguments

entry	The full annotation entry.
overwrite	Overwrite intermediate savefiles in case of incomplete install?
workdir	Location to write savefiles.

Value

A big honking table.

---

post_eupath_goslim_table *Use the POST interface to get GO data from the EuPathDB.*

Description

Use the POST interface to get GO data from the EuPathDB.

Usage

```
post_eupath_goslim_table(
  entry = NULL,
  build_dir = "EuPathDB",
  overwrite = FALSE
)
```

Arguments

entry	The full annotation entry.
overwrite	Overwrite intermediate savefiles in case of incomplete install?
workdir	Location to write savefiles.

Value

A big honking table.

---

post_eupath_interpro_table *Use the post interface to get interpro data.*

Description

Use the post interface to get interpro data.

Usage

```
post_eupath_interpro_table(
  entry = NULL,
  build_dir = "EuPathDB",
  overwrite = FALSE
)
```

Arguments

entry	The full annotation entry.
overwrite	Overwrite the savefile when attempting a redo?
workdir	Location to which to save intermediate savefile.

Value

A big honking table.

---

post_eupath_linkout_table

Use the post interface to get linkout data.

Description

Use the post interface to get linkout data.

Usage

```
post_eupath_linkout_table(
  entry = NULL,
  build_dir = "EuPathDB",
  overwrite = FALSE
)
```

Arguments

entry	The full annotation entry.
overwrite	Overwrite the savefile when attempting a redo?
workdir	Location to which to save intermediate savefile.

Value

A big honking table.

post_eupath_ortholog_table *Use the post interface to get ortholog data.*

Description

The folks at the EuPathDB kindly implemented the table 'OrthologsLite' which makes it possible for me to use this function without trouble.

Usage

```
post_eupath_ortholog_table(
  entry = NULL,
  ortholog_table = NULL,
  build_dir = "EuPathDB",
  gene_ids = NULL,
  overwrite = FALSE
)
```

Arguments

entry	The full annotation entry.
gene_ids	When provided, ask only for the orthologs for these genes.
overwrite	Overwrite incomplete savefiles?
workdir	Location to which to save an intermediate savefile.
table	This defaults to the 'OrthologsLite' table, but that does not exist at all eupathdb subprojects.

Value

A big honking table.

post_eupath_pathway_table *Use the post interface to get pathway data.*

Description

Use the post interface to get pathway data.

Usage

```
post_eupath_pathway_table(
  entry = NULL,
  build_dir = "EuPathDB",
  overwrite = FALSE
)
```

Arguments

entry	The full annotation entry.
overwrite	If trying again, overwrite the savefile?
workdir	Location to which to save intermediate savefile.

Value

A big honking table.

---

post_eupath_pdb_table *Use the post interface to get linkout data.*

Description

Use the post interface to get linkout data.

Usage

```
post_eupath_pdb_table(entry = NULL, build_dir = "EuPathDB", overwrite = FALSE)
```

Arguments

entry	The full annotation entry.
overwrite	Overwrite the savefile when attempting a redo?
workdir	Location to which to save intermediate savefile.

Value

A big honking table.

post_eupath_pubmed_table

Use the post interface to get linkout data.

Description

Use the post interface to get linkout data.

Usage

```
post_eupath_pubmed_table(
  entry = NULL,
  build_dir = "EuPathDB",
  overwrite = FALSE
)
```

Arguments

entry	The full annotation entry.
overwrite	Overwrite the savefile when attempting a redo?
workdir	Location to which to save intermediate savefile.

Value

A big honking table.

post_eupath_table

Queries one of the EuPathDB APIs using a POST request.

Description

This should return a dataframe representation of one table at the eupathdb. It should also simplify the column names into something a bit more consistent.

Usage

```
post_eupath_table(entry, tables = "GOTerms", table_name = NULL, minutes = 30)
```

Arguments

entry	The single metadatum containing the base url of the provider, species, etc.
table_name	The name of the table to extract, this is provided to make for prettier labeling.
minutes	A timeout when querying the eupathdb.
query_body	String of additional query arguments

Value

list containing response from API request.

More information ——————- 1. https://tritrypdb.org/tritrypdb/serviceList.jsp

Author(s)

Keith Hughitt

`prefix_map` *A few webservices at the eupathdb are not what one would expect.*

Description

This maps the service name to the correct hostname of the webserver.

Usage

```
prefix_map(prefix)
```

Arguments

`prefix`	Webservice to query.

`query_s3_ah` *As yet another test, this function will download all the AH data one species at a time.*

Description

This may be run after the data has been uploaded to the annotationhub to ensure that all the uploaded files are actually functional.

Usage

```
query_s3_ah(
  testing = TRUE,
  file_type = "OrgDb",
  cachedir = "~/scratch/eupathdb/cache",
  csv = "inst/extdata/OrgDb_biocv3.10_eupathdbv46_metadata.csv"
)
```

Arguments

`testing`	Use the annotationHub TESTING service rather than production.
`file_type`	Type of data to query.
`cachedir`	Place to put the downloaded files, useful for if one's homedirectory is too small.

`query_s3_file` *Perform what should be a completely silly final check on the file which is to be copied to s3.*

Description

This function really should not be needed. But damn. This will do a final check that the data in the s3 staging directory is loadable in R and return the md5 sum of the file. Thus the md5 sum may be added to the metadata.

Usage

```
query_s3_file(row, file_type = "OrgDb", file_column = "OrgdbFile")
```

Arguments

`row`	Metadata row to query.
`file_type`	Currently I have 3 file types of interest.
`file_column`	Name of the column with the locations of the files of interest.

Value

MD5 checksum of the resulting file, or NULL.

`query_s3_granges` *Perform what should be a completely silly final check on the file which is to be copied to s3.*

Description

This function really should not be needed. But damn. This will do a final check that the data in the s3 staging directory is loadable in R and return the md5 sum of the file. Thus the md5 sum may be added to the metadata.

Usage

```
query_s3_granges(file)
```

Arguments

`file`	Filename to query.

Value

MD5 sum of the file or NULL.

`query_s3_orgdb` *Perform what should be a completely silly final check on the file which is to be copied to s3.*

Description

This function really should not be needed. But damn. This will do a final check that the data in the s3 staging directory is loadable in R and return the md5 sum of the file. Thus the md5 sum may be added to the metadata.

Usage

```
query_s3_orgdb(file)
```

Arguments

`file`	Filename to query.

Value

MD5 sum of the file or NULL.

`query_s3_txdb` *Perform what should be a completely silly final check on the file which is to be copied to s3.*

Description

This function really should not be needed. But damn. This will do a final check that the data in the s3 staging directory is loadable in R and return the md5 sum of the file. Thus the md5 sum may be added to the metadata.

Usage

```
query_s3_txdb(file)
```

Arguments

`file`	Filename to query.

Value

MD5 sum of the file or NULL.

remove_eupath_nas *Get rid of spurious NA entries in a table from the eupathdb.*

Description

If these are not removed, creating the sqlite will fail.

Usage

```
remove_eupath_nas(table, name = "annot")
```

Arguments

table	dataframe of SQLite-bound data
name	column prefix, just used for printing for the moment.

start_eupathdb *Get started with EuPathDB*

Description

This function has always been here. To be honest, I am not completely sure of its purpose.

Usage

```
start_eupathdb(type = "GRanges")
```

Arguments

type	Choose this type of metadatum to open.

Value

Used for its side-effect of opening the package vignette. A vector of experiment identifiers.

Author(s)

Keith Hughitt

Examples

```
start_eupathdb()
```

warn *Warning-level logging function.*

Description

Warning-level logging function.

Usage

```
warn(...)
```

Arguments

...	One or more strings to be logged.

write_eupath_metadata *Standardize the writing of csv metadata.*

Description

This function effectively splits the metadata from a single data frame to a set of individual files, one for each data type created.

Usage

```
write_eupath_metadata(
  metadata,
  webservice,
  file_type = "valid",
  bioc_version = NULL,
  eu_version = NULL,
  build_dir = "EuPathDB"
)
```

Arguments

metadata	Set of metadata.
bioc_version	Version of Bioconductor used for this set of metadata.
eu_version	Version of the EuPathDB used for this set of metadata.
service	EupathDB subproject, or the set of all projects named 'eupathdb'.
type	Either valid or invalid, defines the final output filenames.

Value

List containing the filenames written.

xref_species *Cross reference the taxonomy data from AnnotationHubData::getSpeciesList()*

Description

Previously, the logic of this function resided in download_eupath_metadata(), but I want to be able to test and poke at it separately to more effectively ensure as many taxa as possible pass. Therefore, I split it into its own function. The secondary function of this is to set the 'Species' column as appropriately as possible.

Usage

```
xref_species(
  valid,
  invalid,
  verbose = FALSE,
  taxon_column = "TaxonUnmodified",
  species_column = "GenusSpecies"
)
```

Arguments

valid	Dataframe of entries which have thus far been deemed 'valid' by my tests.
invalid	Dataframe of entries which failed.
verbose	Print some information about what is found?

Value

Likely smaller data frame of valid information and larger dataframe of invalid.

xref_taxonomy *Cross reference the taxonomy data from GenomeInfoDb with EuPathDB metadata.*

Description

Previously, the logic of this function resided in download_eupath_metadata(), but I want to be able to test and poke at it separately to more effectively ensure as many taxa as possible pass. Therefore, I split it into its own function. The secondary function of this is to set the 'Species' column as appropriately as possible.

Usage

```
xref_taxonomy(
  metadata,
  verbose = FALSE,
  species_column = "SpeciesName",
  taxon_column = "TaxonomyID"
)
```

Arguments

metadata	Information provided by downloading the metadata from a eupathdb sub project.
verbose	Print some information about what is found as this runs?
species_column	Because the species column name has changed.
taxon_column	Because the taxonomy column name has changed.

Details

The downside is that there is now yet another for loop in this codebase iterating over the metadata. Ideally, we should be collapsing some of these, on the other hand it will be nice to have the metadata separated by taxa which do and do not match GenomeInfoDb.

Value

List containing entries which pass and fail after xrefing against loadTaxonomyDb().

Index