Package 'EuPathDB'

August 15, 2019

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),
Biobase,
GenomicRanges,
GenomeInfoDbData,
AnnotationHub

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

Suggests AnnotationDbi, AnnotationForge, BiocInstaller, BiocManager,
BiocStyle, BSgenome, BiocInstaller, 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 6.1.1

VignetteBuilder knitr

Collate 'check_csv.R'
'check_files.R'
'clean_pkg.R'
'copy_s3_file.R'
'download_eupath_metadata.R'
'eupathdb.R'
'extract_eupath_orthologs.R'
'extract_gene_locations.R'
'get_all_metadata.R'
'get_eupath_entry.R'
'get_eupath_fields.R'

'get_eupath_pkgnames.R'
'get_kegg_orgn.R'
'get_orthologs_all_genes.R'
'get_orthologs_one_gene.R'
'kegg_vector_to_df.R'
'load_eupath_annotations.R'
'load_kegg_annotations.R'
'load_orgdb_annotations.R'
'load_orgdb_go.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_table.R'
'post_eupath_interpro.R'
'post_eupath_linkout.R'
'post_eupath_ortholog_table.R'
'post_eupath_pathway_table.R'
'post_eupath_pubmed.R'
'post_eupath_raw.R'
'post_eupath_snps.R'
'post_eupath_table.R'
'prefix_map.R'
'write_eupath_metadata.R'
'xref_species.R'
'xref_taxonomy.R'
'zzz.R'

R topics documented:

---

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 = "v3.9",
  eu_version = "v44")
```

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 = "v3.9",
  eu_version = "v44", verbose = TRUE, 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.

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.

Author(s)

atb

`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?

Value

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

Author(s)

Keith Hughitt

`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_ORTHOLOG", org_column = "ORTHOLOGS_ORGANISM",
  url_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?
`url_column`	What column provides the orthomcl group ID?
`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?

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_eupath_metadata() using all the sub-projects of the EuPathDB.*

Description

Invoke download_eupath_metadata() using all the sub-projects of the EuPathDB.

Usage

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

Arguments

`webservice`	Assume all services are desired.

`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")
```

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?

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_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_orthologs_all_genes

Query ortholog tables from the eupathdb one gene at a time.

Description

Deprecated. I think this is no longer needed.

Usage

```
get_orthologs_all_genes(entry = NULL, dir = "EuPathDB",
  gene_ids = NULL, overwrite = TRUE, species_list = NULL)
```

Arguments

`entry`	An entry from the eupathdb metadata to use for other parameters.
`dir`	Directory to which to save intermediate data (currently unused).
`gene_ids`	List of gene IDs to query.
`overwrite`	Overwrite the savefile?
`species_list`	When provided, use this to subset the possible species for ortholog queries, otherwise grab them all.

Details

Querying the full ortholog table at eupathdb.org fails mysteriously. This is a horrible brute-force approach to get around this.

get_orthologs_one_gene

This peculiar and slow querying of orthologs is due to me crashing the eupathdb web servers.

Description

Therefore, I wrote this, which queries one gene at a time. I think it would be nice to change this to query multiple genes at a time.

Usage

```
get_orthologs_one_gene(entry = NULL, gene = "LmjF.01.0010",
  dir = "EuPathDB", species_list = NULL)
```

Arguments

`entry`	Metadata entry.
`gene`	What gene to query?
`dir`	Where to put the checkpoint file?
`species_list`	When provided, use this to subset the possible species for ortholog queries, otherwise grab them all.

Value

table of orthologs for our one gene.

`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_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(species = "Leishmania major",
  webservice = "tritrypdb", eu_version = NULL, wanted_fields = NULL)
```

Arguments

`species`	String containing a unique portion of the desired species.
`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'.

Value

Big huge data frame of annotation data.

`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",
  strand_column = "annot_cdsstrand", start_column = "annot_cdsstart",
  end_column = "annot_cdsend", chromosome_column = "annot_cdschrom",
  type_column = "annot_gene_type", name_column = "annot_cdsname",
  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`	mmm the key type used?
`strand_column`	There are a few fields I want to gather by default: start, end, strand, chromosome, type, and name; but these do not necessarily have consistent names, use this column for the chromosome strand.
`start_column`	Use this column for the gene start.
`end_column`	Use this column for the gene end.
`chromosome_column`	Use this column to identify the chromosome.
`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

Tested in test_45ann_organdb.R 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, gene_ids = NULL, keytype = "gid",
  columns = c("go", "goall", "goid"))
```

Arguments

`orgdb`	OrganismDb instance.
`gene_ids`	Identifiers of the genes to retrieve annotations.
`keytype`	The mysterious keytype returns yet again to haunt my dreams.
`columns`	The set of columns to request.

Details

Tested in test_45ann_organdb.R 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)
```

---

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, dir = "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?
`dir`	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 = NULL, dir = "EuPathDB",
  eu_version = NULL, copy_s3 = FALSE)
```

Arguments

entry	Metadatum entry.
dir	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,
  dir = "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?
dir	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.

Usage

```
make_eupath_orgdb(entry = NULL, dir = "EuPathDB", eu_version = NULL,
  installp = TRUE, kegg_abbreviation = NULL, reinstall = FALSE,
  overwrite = FALSE, copy_s3 = FALSE, do_go = TRUE,
  do_orthologs = TRUE, do_interpro = TRUE, do_linkout = TRUE,
  do_pubmed = TRUE, do_pathway = TRUE, do_kegg = TRUE)
```

Arguments

`entry`	If not provided, then species will get this, it contains all the information.
`dir`	Where to put all the various temporary files.
`eu_version`	Which version of the eupathdb to use for creating this 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?
`do_go`	Create the gene ontology table?
`do_orthologs`	Create the gene ortholog table?
`do_interpro`	Create the interpro table?
`do_linkout`	Create a table of linkout data?
`do_pubmed`	Create a table of pubmed entries?
`do_pathway`	Create the pathway table?
`do_kegg`	Attempt to create the kegg table?

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, dir = "EuPathDB", eu_version = NULL,
  reinstall = FALSE, installp = TRUE, copy_s3 = FALSE)
```

Arguments

`entry`	One row from the organism metadata.
`dir`	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", dir = "EuPathDB")
```

Arguments

`pkgname`	Name of package to move to its final home.
`type`	Which type of package is this?
`dir`	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 *Gather all available annotation data for a given eupathdb species.*

Description

This function fills in the parameters to post_eupath_raw() so that one can download all the available data for a given parasite into one massive table. It should also provide some constraints to the data rather than leaving it all as characters. Caveat: I manually filled in the list 'field_list' to include the variable names and their text associations. This is likely to change in future releases of the tritrypdb. It is probably possible to automagically fill it in. In addition, I am using GenesByMolecularWeight to get the data, which is a bit weird.

Usage

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

Arguments

`entry`	The full annotation entry.
`dir`	A directory into which to write the intermediate savefile.
`overwrite`	If a partial table exists, overwrite it?

Value

A big honking table.

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, dir = "EuPathDB",
  overwrite = FALSE)
```

Arguments

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

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, dir = "EuPathDB",
  overwrite = FALSE)
```

Arguments

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

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, dir = "EuPathDB",
  overwrite = FALSE)
```

Arguments

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

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, dir = "EuPathDB",
  table = "OrthologsLite", gene_ids = NULL, overwrite = FALSE)
```

Arguments

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

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, dir = "EuPathDB",
  overwrite = FALSE)
```

Arguments

`entry`	The full annotation entry.
`dir`	Location to which to save intermediate savefile.
`overwrite`	If trying again, overwrite the 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, dir = "EuPathDB",
  overwrite = FALSE)
```

Arguments

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

Value

A big honking table.

post_eupath_raw

The new eupath system provides 3 output types for downloading data. This uses the raw one.

Description

For the life of me, I could not figure out how to query the big text tables as the tabular format. Every query I sent came back telling me I gave it incorrect parameter despite the fact that I was copy/pasting the example given me by the eupathdb maintainers. So, I got mad and asked it for the raw format, and so this function was born.

Usage

```
post_eupath_raw(entry, question = "GeneQuestions.GenesByMolecularWeight",
  parameters = NULL, table_name = NULL, columns = NULL,
  minutes = 10)
```

Arguments

`entry`	Annotation entry for a given species
`question`	Which query to try? Molecular weight is the easiest, as it was their example.
`parameters`	Query parameters when posting
`table_name`	Used to make sure all columns are unique by prefixing them with the table name.
`columns`	Columns for which to ask.
`minutes`	How long to wait until giving up and throwing an error.

Value

A hopefully huge table of eupath data.

`post_eupath_snp_table` *Use the post interface to get SNP data.*

Description

Use the post interface to get SNP data.

Usage

```
post_eupath_snp_table(entry = NULL, dir = "EuPathDB",
  overwrite = FALSE)
```

Arguments

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

Value

A big honking table.

`post_eupath_table` *Queries one of the EuPathDB APIs using a POST request and returns a dataframe representation of the result. Note: As of 2017/07/13, POST requests are not yet supported on EuPathDB. Note: 2017/07/13 POST queries can only use the new API*

Description

Queries one of the EuPathDB APIs using a POST request and returns a dataframe representation of the result. Note: As of 2017/07/13, POST requests are not yet supported on EuPathDB. Note: 2017/07/13 POST queries can only use the new API

Usage

```
post_eupath_table(query_body, entry, table_name = NULL, minutes = 30)
```

Arguments

`query_body`	String of additional query 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.

Value

list containing response from API request.

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

Author(s)

Keith Hughitt

`start_eupathdb` *Get started with EuPathDB*

Description

Get started with EuPathDB

Usage

```
start_eupathdb()
```

Value

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

Author(s)

Keith Hughitt

Examples

```
start_eupathdb()
```

`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, service = "eupathdb", type = "valid",
  bioc_version = "v3.9", eu_version = "v44")
```

Arguments

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

Value

List containing the filenames written.

---

xref_species *Cross reference the taxonomy data from AnnotationHub-Data::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)
```

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 Eu-PathDB 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)
```

Arguments

`metadata`	Information provided by downloading the metadata from a eupathdb sub project.
`verbose`	Print some information about what is found as this runs?

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().

`%:::%` *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 was as I. I was in no way 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.

Index