Package 'AnnotationHubData'

May 20, 2019

Type Package

Title Transform public data resources into Bioconductor Data Structures

Version 1.15.0

Encoding UTF-8

Maintainer Bioconductor Package Maintainer <maintainer@bioconductor.org>

Depends R (>= 3.2.2), methods, utils, S4Vectors (>= 0.7.21), IRanges (>= 2.3.23), GenomicRanges, AnnotationHub (>= 2.15.15)

Suggests RUnit, knitr, BiocStyle, grasp2db, GenomeInfoDbData

Imports GenomicFeatures, Rsamtools, rtracklayer, BiocGenerics, jsonlite, BiocManager, biocViews, AnnotationDbi, Biobase, Biostrings, DBI, GenomeInfoDb (>= 1.15.4), OrganismDbi, RSQLite, rBiopaxParser, AnnotationForge, futile.logger (>= 1.3.0), XML, RCurl

Description These recipes convert a wide variety and a growing number of public bioinformatic data sets into easily-used standard Bioconductor data structures.

License Artistic-2.0

LazyLoad yes

biocViews DataImport

VignetteBuilder knitr

Collate Message-class.R ImportPreparer-class.R makeAnnotationHubResource.R HubMetadata-class.R AnnotationHubMetadata-class.R utils.R updateResources.R ahmToJson.R webAccessFunctions.R makeBioPaxImporter.R makeChEA.R makedbSNPVCF.R makeEncodeDCC.R makeEnsemblGtfToGRanges.R makeEnsemblFasta.R makeEpigenomeRoadmap.R makeGencodeFasta.R makeGencodeGFF.R makeGrasp2Db.R makeHaemCode.R makeInparanoid8ToDbs.R makeNCBIToOrgDbs.R makeStandardOrgDbsToSqlite.R makeStandardTxDbsToSqlite.R makeRefNet.R makeUCSCChain.R makeUCSC2Bit.R makeUCSCTracks.R trackWithAuxiliaryTableToGRangesRecipe.R UCSCTrackUpdateChecker.R makeEnsemblTwoBit.R validationFunctions.R

git_url https://git.bioconductor.org/packages/AnnotationHubData

git_branch master

git_last_commit 58b8ffa

git_last_commit_date 2019-05-02

Date/Publication 2019-05-19

Author Martin Morgan [ctb],
Marc Carlson [ctb],
Dan Tenenbaum [ctb],
Sonali Arora [ctb],
Paul Shannon [ctb],
Lori Shepherd [ctb],
Bioconductor Package Maintainer [cre]

R topics documented:

`AnnotationHubData-package`

Transform public data resources into Bioconductor Data Structures

---

Description

These recipes convert a wide variety and a growing number of public bioinformatic data sets into easily-used standard Bioconductor data structures.

Details

This package provides a set of methods which convert bioinformatic data resources into standard Bioconductor data types. For example, a UCSC genome browser track, expressed as a BED file, is converted into a GRanges object. Not every valuable data resource can be transformed quite so easily; some require more elaborate transformation, and hence a more specialized recipe. Every effort is made to limit the number of recipes required. One strategy that helps with the principle of "zero curation": unless absolutely required, the "cooked" version of the data resource produced by a recipe is a simple and unembellished reflection of the original data in its downloaded form.

Author(s)

Dan Tenenbaum, Paul Shannon

See Also

AnnotationHubMetadata-class, makeAnnotationHubMetadata

AnnotationHubMetadata-class
Class "AnnotationHubMetadata" *and methods*

Description

AnnotationHubMetadata is used to represent record(s) in the server data base.

Usage

```
AnnotationHubMetadata(AnnotationHubRoot, SourceUrl, SourceType,
 SourceVersion, SourceLastModifiedDate, SourceMd5 =
 NA_character_, SourceSize, DataProvider, Title,
 Description, Species, TaxonomyId, Genome, Tags,
 Recipe, RDataClass, RDataDateAdded, RDataPath,
 Maintainer, ..., BiocVersion = BiocManager::version(),
 Coordinate_1_based = TRUE, Notes = NA_character_,
 DispatchClass, Location_Prefix =
 "http://s3.amazonaws.com/annotationhub/")

toJson(x)
constructSeqInfo(species, genome)

metadata(x, ...)
hubError(x)
inputFiles(object, ...)
outputFile(object)
ahmToJson(ahm)
deleteResources(id)
getImportPreparerClasses()
makeAnnotationHubResource(objName, makeAnnotationHubMetadataFunction,
  ..., where)
```

Arguments

AnnotationHubRoot	character(1) Absolute path to directory structure containing resources to be added to AnnotationHub. Internal use only.
SourceUrl	character() URL of original resource(s).
SourceType	character() Form of original data, e.g., BED, FASTA, etc. getValidSourceTypes() list currently acceptable values. If nothing seems appropiate for your data reach out to maintainer@bioconductor.org.
SourceVersion	character(1) Version of original file.
SourceLastModifiedDate	POSIXct() The date when the source was last modified.
SourceMd5	character() md5 hash of original file.

`SourceSize` `numeric(1)` Size of original file in bytes.

`DataProvider` `character(1)` Provider of original data, e.g., NCBI, UniProt etc.

`Title` `character(1)` Title for the resource with version or genome build as appropriate.

`Description` `character(1)` Description of the resource. May include details such as data type, format, study origin, sequencing technology, treated vs control, number of samples etc.

`Species` `character(1)` Species name. For help on valid species see `getSpeciesList`, `validSpecies`, or `suggestSpecies`.

`TaxonomyId` `character(1)` NCBI code. There are checks for valid taxonomyId given the Species which produce warnings. See GenomeInfoDb::loadTaxonomyDb() for full validation table.

`Genome` `character(1)` Name of genome build.

`Tags` `character()` Free-form tags that serve as search terms.

`Recipe` `character(1)` Name of recipe function. Only applicable to recipes created by the Bioconductor core team and included in AnnotationHubData base code.

`RDataClass` `character()` Class of derived R object, e.g., GRanges. Length must match the length of `RDataPath`.

`RDataDateAdded` `POSIXct()` Date resource was added to AnnotationHub. The default is today's date and is auto-generated when metadata are constructed. Resources will appear in snapshots with a date greater than or equal to the `RDataDateAdded`.

`RDataPath` `character()` File path to where object is stored in AWS S3 bucket or on the web.This field should be the remainder of the path to the resource. The `Location_Prefix` will be prepended to `RDataPath` for the full path to the resource. If the resource is stored in Bioconductor's AWS S3 buckets, it should start with the name of the package associated with the metadata and should not start with a leading slash. It should include the resource file name. For strongly associated files, like a bam file and its index file, the two files should be separates with a colon `:`. This will link a single hub id with the multiple files.

`Maintainer` `character(1)` Maintainer name and email address, 'A Maintainer `a.maintainer@email.com`'

`BiocVersion` `character(1)`. The first Bioconductor version the resource was made available for. Unless removed from the hub, the resource will be available for all versions greater than or equal to this field.

`Coordinate_1_based` `logical(1)` Do coordinates start with 1 or 0?

`DispatchClass` `character(1)`. Determines how data are loaded into R. The value for this field should be 'Rda' if the data were serialized with `save()` and 'Rds' if serialized with `saveRDS`. The filename should have the appropriate 'rda' or 'rds' extension.

A number of dispatch classes are pre-defined in AnnotationHub/R/AnnotationHubResource-class.R with the suffix 'Resource'. For example, if you have sqlite files, the AnnotationHubResource-class.R defines SQLiteFileResource so the DispatchClass would be SQLiteFile. Contact maintainer@bioconductor.org if you are not sure which class to use. The function `AnnotationHub::DispatchClassList()` will output a matrix of currently implemented DispatchClass and brief description of utility. If a predefine class does not seem appropriate contact maintainer@bioconductor.org.

Location_Prefix
: character(1) URL location of AWS S3 bucket or web site where resource is located.

Notes
: character() Notes about the resource.

ahm
: An instance of class AnnotationHubMetadata.

x
: An instance of class AnnotationHubMetadata.

object
: An AnnotationHubRecipe instance.

species
: character(1) The organism, e.g., "Homo sapiens".

genome
: character(1) The genome build, e.g., "hg19".

id
: An id whose DB record is to be fully deleted.

objName
: character(1) The name of the PreparerClass used for dispatch.

makeAnnotationHubMetadataFunction
: function Function (name) that makes AnnotationHubMetadata objects from the resource(s).

where
: Environment where function definition is defined. Default value is sufficient.

...
: Additional arguments passed to methods.

Value

AnnotationHubMetadata returns an instance of the class.

jsonPath returns a character(1)) representation of the full path to the location of the json file associated with this record.

toJson returns the JSON representation of the record.

fromJson retuns an instance of the class, as parsed from the JSON file.

Objects from the Class

Objects can be created by calls to the constructor, AnnotationHubMetadata().

Author(s)

Dan Tenenbaum and Marc Carlson

Examples

```
getClass("AnnotationHubMetadata")
```

---

flog *flog*

Description

Write logging message to console and a file.

Usage

```
flog(level, ...)
```

Arguments

level	A characater(1) string object.
...	Further arguments.

Details

Writes the message to the console and to a file.

Value

None.

Author(s)

Dan Tenenbaum

See Also

futile.logger

ImportPreparer-class *Class* ImportPreparer *and generic* newResources

---

Description

The ImportPreparer and derived classes are used for dispatch during data discovery (see newResources). There is one ImportPreparer class for each data source for AnnotationHubMetadata.

newResources is a generic function; with methods implemented for each ImportPreparer.

Author(s)

Martin Morgan

See Also

AnnotationHubMetadata.

Examples

```
getImportPreparerClasses()
```

makeAnnotationHubMetadata
Make AnnotationHubMetadata objects from csv file of metadata

Description

Make AnnotationHubMetadata objects from .csv files located in the "inst/extdata/" package directory of an AnnotationHub package.

Usage

```
makeAnnotationHubMetadata(pathToPackage, fileName=character())
```

Arguments

pathToPackage	Full path to data package including the package name; no trailing slash
fileName	Name of metadata file(s) with csv extension. If none are provided, all files with .csv extension in "inst/extdata" will be processed.

Details

- makeAnnotationHubMetadata: Reads the resource metadata from .csv files into a AnnotationHubMetadata object. The AnnotationHubMetadata is inserted in the AnnotationHub database. Intended for internal use or package authors checking the validity of package metadata.

- Formatting metadata files:

  makeAnnotationHubMetadata reads .csv files of metadata located in "inst/extdata". Internal functions perform checks for required columns and data types and can be used by package authors to validate their metadata before submitting the package for review.

  The rows of the .csv file(s) represent individual Hub resources (i.e., data objects) and the columns are the metadata fields. All fields should be a single character string of length 1.

  Required Fields in metadata file:

  - Title: character(1). Name of the resource. This can be the exact file name (if self-describing) or a more complete description.
  - Description: character(1). Brief description of the resource, similar to the 'Description' field in a package DESCRIPTION file.
  - BiocVersion: character(1). The first Bioconductor version the resource was made available for. Unless removed from the hub, the resource will be available for all versions greater than or equal to this field.
  - Genome: character(1). Genome.
  - SourceType: character(1). Format of original data, e.g., FASTA, BAM, BigWig, etc. getValidSourceTypes() list currently acceptable values. If nothing seems appropiate for your data reach out to maintainer@bioconductor.org.
  - SourceUrl: character(1). Optional location of original data files. Multiple urls should be provided as a comma separated string.
  - SourceVersion: character(1). Version of original data.
  - Species: character(1). Species. For help on valid species see getSpeciesList,validSpecies,or suggestSpecies.

- TaxonomyId: `character(1)`. Taxonomy ID. There are checks for valid taxonomyId given the Species which produce warnings. See GenomeInfoDb::loadTaxonomyDb() for full validation table.
- Coordinate_1_based: `logical`. TRUE if data are 1-based.
- DataProvider: `character(1)`. Name of company or institution that supplied the original (raw) data.
- Maintainer: `character(1)`. Maintainer name and email in the following format: Maintainer Name <username@address>.
- RDataClass: `character(1)`. R / Bioconductor class the data are stored in, e.g., GRanges, SummarizedExperiment, ExpressionSet etc.
- DispatchClass: `character(1)`. Determines how data are loaded into R. The value for this field should be 'Rda' if the data were serialized with `save()` and 'Rds' if serialized with `saveRDS`. The filename should have the appropriate 'rda' or 'rds' extension.
  A number of dispatch classes are pre-defined in AnnotationHub/R/AnnotationHubResourceclass.R with the suffix 'Resource'. For example, if you have sqlite files, the AnnotationHubResourceclass.R defines SQLiteFileResource so the DispatchClass would be SQLiteFile. Contact maintainer@bioconductor.org if you are not sure which class to use. The function `AnnotationHub::DispatchClassList()` will output a matrix of currently implemented DispatchClass and brief description of utility. If a predefine class does not seem appropriate contact maintainer@bioconductor.org.
- Location_Prefix: `character(1)`. Do not include this field if data are stored in the Bioconductor AWS S3; it will be generated automatically.
  If data will be accessed from a location other than AWS S3 this field should be the base url.
- RDataPath: `character()`.This field should be the remainder of the path to the resource. The `Location_Prefix` will be prepended to `RDataPath` for the full path to the resource. If the resource is stored in Bioconductor's AWS S3 buckets, it should start with the name of the package associated with the metadata and should not start with a leading slash. It should include the resource file name. For strongly associated files, like a bam file and its index file, the two files should be separates with a colon `:`. This will link a single hub id with the multiple files.
- Tags: `character() vector`. 'Tags' are search terms used to define a subset of resources in a `Hub` object, e.g, in a call to `query`.
  For ExperimentHub resources, 'Tags' are automatically generated from the 'biocViews' in the DESCRIPTION. 'Tags' values supplied by the user are not be entered in the database and are not part of the formal metadata. This 'controlled vocabulary' approach was taken to limit the search terms to a well defined set and may change in the future.
  'Tags' for AnnotationHub resources are a free-form field of search terms defined by the user. The package name is added as one of the 'Tags' before the metadata are finalized. Multiple 'Tags' are specified as a colon separated string, e.g., tags for two resources would look like this:

```
Tags=c("tag1:tag2:tag3", "tag1:tag3")
```

NOTE: The metadata file can have additional columns beyond the 'Required Fields' listed above. These values are not added to the Hub database but they can be used in package functions to provide an additional level of metadata on the resources.

Value

A named list the length of `fileName`. Each element is a list of of `AnnotationHubMetadata` objects created from the .csv file.

See Also

- updateResources
- AnnotationHubMetadata class

Examples

```
## Each row of the metadata file represents a resource added to one of
## the 'Hubs'. This example creates a metadata.csv file for a single resource.
## In the case of multiple resources, the arguments below would be character
## vectors that produced multiple rows in the data.frame.

meta <- data.frame(
    Title = "RNA-Sequencing dataset from study XYZ",
    Description = paste0("RNA-seq data from study XYZ containing 10 normal ",
 "and 10 tumor samples represented as a",
 "SummarizedExperiment"),
    BiocVersion = "3.4",
    Genome = "GRCh38",
    SourceType = "BAM",
    SourceUrl = "http://www.path/to/original/data/file",
    SourceVersion = "Jan 01 2016",
    Species = "Homo sapiens",
    TaxonomyId = 9606,
    Coordinate_1_based = TRUE,
    DataProvider = "GEO",
    Maintainer = "Your Name <youremail@provider.com>",
    RDataClass = "SummarizedExperiment",
    DispatchClass = "Rda",
    ResourceName = "FileName.rda"
)

## Not run:
## Write the data out and put in the inst/extdata directory.
write.csv(meta, file="metadata.csv", row.names=FALSE)

## Test the validity of metadata.csv
makeAnnotationHubMetadata("path/to/mypackage")

## End(Not run)
```

---

makeEnsemblFasta *Functions to convert Ensembl FASTA files to FaFile and TwoBitFile for inclusion in AnnotationHub.*

Description

Transform an Ensembl FASTA file to a Bioconductor FaFile or ToBitFile.

Usage

```
makeEnsemblFastaToAHM(currentMetadata, baseUrl = "ftp://ftp.ensembl.org/pub/",
                      baseDir = "fasta/", release,
```

```
                        justRunUnitTest = FALSE,
                        BiocVersion = BiocManager::version())

makeEnsemblTwoBitToAHM(currentMetadata, baseUrl = "ftp://ftp.ensembl.org/pub/",
                        baseDir = "fasta/", release,
                        justRunUnitTest = FALSE,
                        BiocVersion = BiocManager::version())

ensemblFastaToFaFile(ahm)

ensemblFastaToTwoBitFile(ahm)
```

Arguments

currentMetadata	Currently not used. Intended to be a list of metadata to filter, i.e., records that do not need to be processed again. Need to remove or fix.
baseUrl	ftp file location.
baseDir	ftp file directory.
release	Integer version number, e.g., "84".
justRunUnitTest	A logical. When TRUE, a small number of records (usually 5) are processed instead of all.
BiocVersion	A character(1) Bioconductor version. The resource will be available in Bioconductor >= to this version. Default value is the current version, specified with BiocManager::version().
ahm	List of AnnotationHubMetadata instances.

Details

makeEnsemblFastaToAHM and makeEnsemblTwoBitToAHM process metadata into a list of AnnotationHubMetadata objects.

ensemblFastaToFaFile unzips a .gz files, creates and index and writes out .rz and .rz.fai files to disk. ensemblFastaToTwoBit converts a fasta file to twobit format and writes the .2bit file out to disk.

Value

makeEnsemblFastaToAHM and makeEnsemblTwoBitToAHM return a list of AnnotationHubMetadata objects.

ensemblFastaToFaFile write out .rz and .rz.fai files to disk. ensemblFastaToTwoBit writes out a .2bit file to disk.

Author(s)

Bioconductor Core Team

See Also

- updateResources
- AnnotationHubMetadata

Examples

```
## updateResources() generates metadata, process records and
## pushes files to AWS S3 buckets. See ?updateResources for details.

## 'release' is passed to makeEnsemblFastaToFaFile.
## Not run:
meta <- updateResources("/local/path",
                        BiocVersion = c("3.2", "3.3"),
                        preparerClasses = "EnsemblFastaImportPreparer",
                        metadataOnly = TRUE, insert = FALSE,
                        justRunUnitTest = FALSE, release = "83")

## End(Not run)
```

`makeGencodeFasta` *Recipe to add Gencode FASTA resources to AnnotationHub*

Description

Create metadata and process raw Gencode FASTA files for inclusion in AnnotationHub

Usage

```
makeGencodeFastaToAHM(currentMetadata,
                      baseUrl="ftp://ftp.ebi.ac.uk/pub/databases/gencode/",
                      species=c("Human", "Mouse"), release,
                      justRunUnitTest=FALSE,
                      BiocVersion=BiocManager::version())

gencodeFastaToFaFile(ahm)
```

Arguments

`currentMetadata`	Currently not used. Intended to be a list of metadata to filter, i.e., records that do not need to be processed again. Need to remove or fix.
`baseUrl`	ftp file location.
`species`	A `character(1)` of the species. Currently "Human" and "Mouse" are supported.
`release`	A `character` string of the release number.
`justRunUnitTest`	A `logical`. When TRUE, a small number of records (usually 5) are processed instead of all.
`BiocVersion`	A `character` vector of Bioconductor versions the resources should be available for.
`ahm`	List of `AnnotationHubMetadata` instances.

Details

- Documentation: http://www.gencodegenes.org/releases/
- File download location: ftp://ftp.ebi.ac.uk/pub/databases/gencode/. Gencode_human and Gencode_mouse are used.
- Files downloaded: Code is currently specific for human and mouse. Files chosen for download are described in AnnotationHubData:::.gencodeDescription().

Value

`makeGencodeFastaAHM` returns a list of `AnnotationHubMetadata` instances. `gencodeFastaToFaFile` returns nothing.

Author(s)

Bioconductor Core Team.

See Also

- updateResources
- AnnotationHubMetadata

Examples

```
## updateResources() generates metadata, process records and
## pushes files to AWS S3 buckets.

## To run the GencodeFasta recipe specify
## 'preparerClasses = GencodeFastaImportPreparer'. The 'species' and 'release'
## arguments are passed to makeGencodeFastaAHM().
## Not run:
meta <- updateResources("/local/path",
                        BiocVersion = c("3.2", "3.3"),
                        preparerClasses = "GencodeFastaImportPreparer",
                        metadataOnly = TRUE, insert = FALSE,
                        justRunUnitTest = FALSE)


## End(Not run)
```

---

makeStandardOrgDbs *Functions to add OrgDb and TxDb sqlite files to AnnotationHub*

Description

Add OrgDb and TxDb sqlite files to AnnotationHub

Usage

```
makeStandardOrgDbsToAHM(currentMetadata, justRunUnitTest = FALSE,
                        BiocVersion = BiocManager::version(),
                        downloadOrgDbs = TRUE)

makeStandardTxDbsToAHM(currentMetadata, justRunUnitTest = FALSE,
                       BiocVersion = BiocManager::version(), TxDbs)

makeNCBIToOrgDbsToAHM(currentMetadata, justRunUnitTest = FALSE,
                      BiocVersion = BiocManager::version(),
                      baseUrl = "ftp://ftp.ncbi.nlm.nih.gov/gene/DATA/")
```

Arguments

currentMetadata	Historically was intended to be a list of metadata to filter, i.e., records that do not need to be processed again. In some recipes this is used as a way to pass additional arguments. Need to remove or make consistent.
baseUrl	A character(). The file location.
justRunUnitTest	A logical. When TRUE, a small number of records (usually <= 5) are processed instead of all.
BiocVersion	A character(1). The resource will be available for Bioconductor versions greater than and equal to this version. Default is BiocManaer::version().
TxDbs	Character vector of the TxDb names; generally includes TxDbs that were new or updated for the current release.
downloadOrgDbs	A logical. Indicates if all OrgDb packages in the Bioconductor repo should be downloaded and installed. This should be TRUE the first time the recipe is run and can be FALSE for subsequent runs when testing.

Details

makeStandardOrgDbsToAHM and makeStandardTxDbsToAHM extracts the sqlite files from the existing OrgDb and TxDb packages in the Bioconductor repositories and generate associated metadata.

makeNCBIToOrgDbsToAHM creates sqlite files and metadata for 1000 organisms with the makeOrgPackageFromNCBI function. These organisms are less 'main stream' than those hosted in the Bioconductor repository (makeStandardOrgDbsToAHM) and the databases are less comprehensive because data only come from one source, NCBI.

Value

List of AnnotationHubMetadata objects.

Author(s)

Bioconductor Core Team

See Also

- updateResources
- AnnotationHubMetadata

Examples

```
## Not run:
## In Bioconductor 3.5, one new TxDb was added and 4 active
## tracks were updated. This piece of code shows how to add these 5
## packages to AnnotationHub.

## Step I: generate metadata
##
## Generate the metadata with the low-level helper for inspection.
TxDbs <- c("TxDb.Ggallus.UCSC.galGal5.refGene",
           "TxDb.Celegans.UCSC.ce11.refGene",
           "TxDb.Rnorvegicus.UCSC.rn5.refGene",
           "TxDb.Dmelanogaster.UCSC.dm6.ensGene",
           "TxDb.Rnorvegicus.UCSC.rn6.refGene")
meta <- makeStandardTxDbsToAHM(currentMetadata=list(AnnotationHubRoot=getwd()),
                               justRunUnitTest=FALSE,
                               TxDbs = TxDbs)

## Once the low-level helper runs with no errors, try generating the
## metadata with the high-level wrapper updateResources(). Setting
## metadataOnly=TRUE will generate metadata only and not push resources
## to S3. insert=FALSE prevents the metadata from being inserted in the
## database.
##
## The metadata generated by updateResources() will be the same as that
## generated by makeStandardTxDbsToAHM(). Both should be a list the same
## length as the number of TxDbs specified.
meta <- updateResources(getwd(),
                        preparerClasses="TxDbFromPkgsImportPreparer",
                        metadataOnly=TRUE, insert = FALSE,
                        justRunUnitTest=FALSE, TxDbs = TxDbs)

INFO [2017-04-11 09:12:09] Preparer Class: TxDbFromPkgsImportPreparer
complete!
> length(meta)
[1] 5

## Step II: push resources to S3
##
## If the metadata looks correct we are ready to push resources to S3.
## Set metadataOnly=FALSE but keep insert=FALSE.

meta <- updateResources(getwd(),
                        BiocVersion="3.5",
                        preparerClasses="TxDbFromPkgsImportPreparer",
                        metadataOnly=FALSE, insert = FALSE,
                        justRunUnitTest=FALSE, TxDbs = TxDbs)

## Step III: insert metadata in AnnotationHub production database
##
## Inserting the metadata in the database is usually done as a separte step
## and with the help of the AnnotationHub docker.
## Set metadataOnly=TRUE and insert=TRUE.
meta <- updateResources(getwd(),
                        BiocVersion="3.5",
                        preparerClasses="TxDbFromPkgsImportPreparer",
```

```
                        metadataOnly=FALSE, insert = FALSE,
                        justRunUnitTest=FALSE, TxDbs = TxDbs)


## End(Not run)
```

---

updateResources *updateResources*

Description

Add new resources to AnnotationHub

Usage

```
updateResources(AnnotationHubRoot, BiocVersion = BiocManager::version(),
                preparerClasses = getImportPreparerClasses(),
                metadataOnly = TRUE, insert = FALSE,
                justRunUnitTest = FALSE, ...)

pushResources(allAhms, uploadToS3 = TRUE, download = TRUE)

pushMetadata(allAhms, url)
```

Arguments

AnnotationHubRoot	Local path where files will be downloaded.
BiocVersion	A character(1) Bioconductor version. The resource will be available in Bioconductor >= to this version. Default value is the current version, specified with BiocManager::version().
preparerClasses	One of the ImportPreparer subclasses defined in getImportPreparer(). This class is used for dispatch during data discovery.
metadataOnly	A logical to specify the processing of metadata only or both metadata and data files. When FALSE, metadata are generated and data files are downloaded, processed and pushed to their final location in S3 buckets. metadata = TRUE produces only metadata and is useful for testing.
insert	NOTE: This option is for inserting metadata records in the production data base (done by Bioconductor core team member) and is for internal use only. A logical to control if metadata are inserted in the AnnotationHub db. By default this option is FALSE which is a useful state in which to test a new recipe and confirm the metadata fields are correct. When insert = TRUE, the "AH_SERVER_POST_URL" global option must be set to the http location of the AnnotationHubServer in the global environment or .Rprofile. Additionally, AWS command line tools must be installed on the local machine to push files to S3 buckets. See https://aws.amazon.com/cli/ for installation instructions.

justRunUnitTest
: A logical. When TRUE, a small number of records (usually 5) are processed instead of all.

allAhms
: List of AnnotationHubMetadata objects.

url
: URL of AnnotationHub database where metadata will be inserted.

uploadToS3
: A logical indicating whether resources should be uploaded to the AWS S3 bucket.

download
: A logical indicating whether resources should be downloaded from resource url.

...
: Arguments passed to other methods such as regex, baseUrl, baseDir.

Details

- updateResources:
  updateResources is responsible for creating metadata records and downloading, processing and pushing data files to their final resting place. The
- preparerClasses argument is used in method dispatch to determine which recipe is used.
  By manipulating the metadataOnly, insert and justRunUnitTest arguments one can flexibly test the metadata for a small number of records with or without downloading and processing the data files.
- global options:
  Several recipes look at the "AnnotationHub_Use_Disk" option to determine if the file is written to S3. This is legacy behavior and not clearly documented. If the recipe being run respects this option it must be set before running updateResources,
  When insert = TRUE the "AH_SERVER_POST_URL" option must be set to the https location of the AnnotationHub db.

Value

A list of AnnotationHubMetadata objects.

Author(s)

Martin Morgan, Marc Carlson

See Also

- AnnotationHubMetadata

Examples

```
## Not run:

## ------------------------------------------------------------------------
## Inspect metadata:
## ------------------------------------------------------------------------
## A useful first step in testing a new recipe is to generate and
## inspect a small number of metadata records. The combination of
## 'metadataOnly=TRUE', 'insert=FALSE' and 'justRunUnitTest=TRUE'
## generates metadata for the first 5 records and does not download or
## process any data.
```

```
meta <- updateResources("/local/path",
                        BiocVersion = "3.3",
                        preparerClasses = "EnsemblFastaImportPreparer",
                        metadataOnly = TRUE, insert = FALSE,
                        justRunUnitTest = TRUE,
                        release = "84")

INFO [2015-11-12 07:58:05] Preparer Class: EnsemblFastaImportPreparer
Ailuropoda_melanoleuca.ailMel1.cdna.all.fa.gz
Ailuropoda_melanoleuca.ailMel1.dna_rm.toplevel.fa.gz
Ailuropoda_melanoleuca.ailMel1.dna_sm.toplevel.fa.gz
Ailuropoda_melanoleuca.ailMel1.dna.toplevel.fa.gz
Ailuropoda_melanoleuca.ailMel1.ncrna.fa.gz

## The return value is a list of metadata for the first 5 records:

> names(meta)
[1] "FASTA cDNA sequence for Ailuropoda melanoleuca"
[2] "FASTA DNA sequence for Ailuropoda melanoleuca"
[3] "FASTA DNA sequence for Ailuropoda melanoleuca"
[4] "FASTA DNA sequence for Ailuropoda melanoleuca"
[5] "FASTA ncRNA sequence for Ailuropoda melanoleuca"


## Each record is of class AnnotationHubMetadata:

> class(meta[[1]])
[1] "AnnotationHubMetadata"
attr(,"package")
[1] "AnnotationHubData"

## ------------------------------------------------------------------------
## Insert metadata in the db and process/push data files:
## ------------------------------------------------------------------------
## This next code chunk creates the metadata and downloads and processes
## the data (metadataOnly=FALSE). If all files are successfully pushed to
## to their final resting place, metadata records are inserted in the
## AnnotationHub db (insert=TRUE). Metadata insertion is done by a
## Bioconductor team member; contact maintainer@bioconductor.org for help.

meta <- updateResources("local/path",
                        BiocVersion = "3.5",
                        preparerClasses = "EnsemblFastaImportPreparer",
                        metadataOnly = FALSE, insert = TRUE,
                        justRunUnitTest = FALSE,
                        regex = ".*release-81")

## ------------------------------------------------------------------------
## Recovery helpers:
## ------------------------------------------------------------------------

## pushResources() and pushMetadata() are both called from updateResources()
## but can be used solo for testing or completing a run that
## terminated unexpectedly.

## Download, process and push to S3 the last 2 files in 'meta':
```

```
sub <- meta[length(meta) - 1:length(meta)]
pushResources(sub)

## Insert metadata in the AnotationHub db for the last 2 files in 'meta':

pushMetadata(sub, url = getOption("AH_SERVER_POST_URL"))

## End(Not run)
```

---

upload_to_S3 *Upload a file to Amazon S3*

Description

This function is for uploading a file resource to the S3 cloud.

Usage

```
upload_to_S3(file, remotename, bucket, profile, acl="public-read")
```

Arguments

file	The file to upload.
remotename	The name this file should have in S3, including any "keys" that are part of the name. This should not start with a slash (if it does, the leading slash will be removed), but can contain forward slashes.
bucket	Name of the S3 bucket to copy to.
profile	Corresponds to a profile set in the config file for the AWS CLI (see the documentation). If this argument is omitted,the default profile is used.
acl	Should be one of private, public-read, or public-read-write.

Details

Uses the AWS Command Line Interface to copy a file to Amazon S3. Assumes the CLI is properly configured and that the aws program is in your PATH. The CLI should be configured with the credentials of a user who has permission to upload to the appropriate bucket. It's recommended to use IAM to set up users with limited permissions.

There is an RAmazonS3 package but it seems to have issues uploading files to S3.

Value

TRUE on success. If the command fails, the function will exit with an error.

Author(s)

Dan Tenenbaum

Examples

```
## Not run:
upload_to_S3("myfile.txt", "foo/bar/baz/yourfile.txt")
# If this is successful, the file should be accessible at
# http://s3.amazonaws.com/annotationhub/foo/bar/baz/yourfile.txt

## End(Not run)
```

---

validationFunctions *ValidationFunctions*

Description

Functions to assist in the validation process of creating the metadata.csv file for Hub Resources

Usage

```
getSpeciesList(verbose=FALSE)

validSpecies(species, verbose=TRUE)

suggestSpecies(query, verbose=FALSE, op=c("|", "&"))

getValidSourceTypes()

checkSpeciesTaxId(txid, species, verbose=TRUE)

validDispatchClass(dc, verbose=TRUE)
```

Arguments

species	species to validate (may be single value or list)
query	terms to query. Whether AND or OR is determined by argument op.
verbose	should additional information and useful tips be displayed
op	Should searching of mulitple terms be conditional OR ("|") or AND ("&")
txid	taxonomy id (single value or list)
dc	Dispatch class to validate (may be single value or list)

Details

- getSpeciesList: Provides a list of valid species as determined by the GenomeInfoDbData package specData.rda file.
- validSpecies: True/False if argument is considered a valid species based on the list generated by getSpeciesList. A species may be deemed invalid if the capitalization mismatches or punctuation mismatches. Use suggestSpecies to find similar terms.
- suggestSpecies: Based on a term or multiple terms suggest possible valid species.
- getValidSourceTypes: returns list of acceptable values for SourceType in metadata.csv. If you think a valid source type should be added to the list please reach out to maintainer@bioconductor.org

- checkSpeciesTaxId: cross validates a list of species and taxonomy ids for expected values based on `GenomeInfoDb::loadTaxonomyDb()`. Warning when there is a mismatch.
- validDispatchClass: TRUE/FALSE if argument is considered a valid DispatchClass based on the currently available methods in AnnotationHub. Use `AnnotationHub::DispatchClassList()` to see the table of currently available methods. If a currently available method is not appropriate for your resource, please reach out to Lori Shepherd `<Lori.Shepherd@roswellpark.org>` to request a new method be added.

Value

- For getSpeciesList: character vector of valid species
- For validSpecies: True/False if all species given as argument are valid
- For suggestSpecies: data.frame of taxonomy id and species name of possible valid species based on given query key words.
- For getValidSourceTypes: character vector of valid source types.
- For checkSpeciesTaxId: NULL if check is verfified, If verbose is ture a table of suggested values along with the warning.
- For validDispatchClass: True/False if all dispatch class given as argument are valid

Author(s)

Lori Shepherd

See Also

- AnnotationHubMetadata
- makeAnnotationHubMetadata

Examples

```
species = getSpeciesList()

# following is TRUE

validSpecies("Homo sapiens")
# followin is FALSE because of starting "h"
validSpecies("homo sapiens")

# can provide multiple, if any are not valid FALSE
# TRUE
validSpecies(c("Homo sapiens", "Canis domesticus"))

suggestSpecies("Canis")

getValidSourceTypes()

checkSpeciesTaxId(1003232, "Edhazardia aedis")
checkSpeciesTaxId(9606, "Homo sapiens")

validDispatchClass("GRanges")
```

Index