Package 'AnnotationHubData'

April 14, 2017

Type Package

Title Transform public data resources into Bioconductor Data Structures

Version 1.4.1

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

Suggests RUnit, knitr,RMySQL, BiocStyle, grasp2db

Imports GenomicFeatures, Rsamtools, rtracklayer, BiocGenerics, jsonlite, BiocInstaller, httr, AnnotationDbi, Biobase, Biostrings, DBI, GEOquery, GenomeInfoDb, OrganismDbi, RSQLite, rBiopaxParser, AnnotationForge, futile.logger (>= 1.3.0), XML, xml2, curl

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 readMetadataFromCsv.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 makeRecentOrgPkgsToDbs.R makePazar.R makeRefNet.R makeUCSCChain.R makeUCSC2Bit.R makeUCSCTracks.R trackWithAuxiliaryTableToGRangesRecipe.R UCSCTrackUpdateChecker.R makeEnsemblTwoBit.R

NeedsCompilation no

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

R topics documented:

`AnnotationHubRecipes-package`
Transform public data resources into Bioconductor Data Structures ~~ package title ~~

Description

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

Details

Package:	AnnotationHubRecipes
Type:	Package
Version:	1.0
Date:	2012-11-30
License:	Artistic-2.0

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

`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 = biocVersion(),
                 Coordinate_1_based = TRUE, Notes = NA_character_,
                 DispatchClass, Location_Prefix =
                 "http://s3.amazonaws.com/annotationhub/")

jsonPath(x)
toJson(x)
constructAnnotationHubMetadataFromSourceFilePath(ahroot, originalFile)
constructMetadataFromJsonPath(ahroot, jsonpath)
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.
`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.
`TaxonomyId`	`character(1)` NCBI code
`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(1)` Class of derived R object, e.g., GRanges.
`RDataDateAdded`	`POSIXct()` Date resource was added to AnnotationHub; used to determine snapshots.
`RDataPath`	`character(1)` File path to serialized form. Internal use only.
`Maintainer`	`character(1)` Maintainer name and email address, e.g., ‘YourName `YourName@email.com`’
`BiocVersion`	`character(1)` The first Biocondcutor version the resource will be made available under. Default is the current version, specified with biocVersion().
`Coordinate_1_based`	`logical(1)` Do coordinates start with 1 or 0?
`DispatchClass`	`character(1)` String used in AnnotationHub:::.get1() method dispatch when loading the resource from AnnotationHub. This value is often the same as the RDataClass. Dispatch classes are defined on the .get1() methods in AnnotationHub. If one exists for your data type, simply use the name of the data type, e.g., if the data are saved as ExpressionSet objects, `DispatchClass` should be "ExpressionSet"; when the resource is downloaded from AnnotationHub with '[[' the "Resources" extension is (automatically) added to make the new class "ExpressionSetResource". If the data are saved as .rda files `DispatchClass` should be "Rda" and when downloaded in AnnotationHub the class name becomes "RdaResource" and so on. If there is not a .get1() method in AnnotationHub for your object type, contact maintainer@bioconductor.org for assistance.
`Location_Prefix`	`character(1)` Base path where the resource is coming from, e.g., a web site or Amazon S3.
`Notes`	`character()` Notes about the resource.
`ahm`	An instance of class `AnnotationHubMetadata`.
`x`	An instance of class `AnnotationHubMetadata`.
`jsonpath`	`character(1)` Full path to a JSON representation of `AnnotationHubMetadata`-class.

`ahroot`	A `character(1)` `AnnotationHubRoot` to be added to the path / location of the returned instance.
`originalFile`	A `character(1)` Original file name.
`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

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 metadata.csv file located in the "inst/extdata/" package directory of an AnnotationHub package.

Usage

```
makeAnnotationHubMetadata(pathToPackage, fileName="metadata.csv")
```

Arguments

pathToPackage	Full path to data package including the package name; no trailing slash
fileName	Name of metadata file with csv extension.

Details

- makeAnnotationHubMetadata: Reads the resource metadata in the metadata.csv file 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.

Value

A list of `AnnotationHubMetadata` objects.

See Also

- `updateResources`
- `readMetadataFromCsv`
- `AnnotationHubMetadata` class

Examples

```
## Not run:
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 = biocVersion())

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

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` vector of Bioconductor versions the resources should be available for.
`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.sanger.ac.uk/pub/gencode/",
                      species=c("Human", "Mouse"), release,
                      justRunUnitTest=FALSE,
                      BiocVersion=biocVersion())

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.sanger.ac.uk/pub/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)
```

`readMetadataFromCsv` *Reads a csv file of resource metadata into a data.frame*

Description

Reads the metadata file located in the "inst/extdata" package directory into a data.frame.

Usage

```
readMetadataFromCsv(pathToPackage, fileName="metadata.csv")
```

Arguments

`pathToPackage`	Full path to data package including package name; no trailing slash.
`fileName`	Name of metadata file with csv extension.

Details

- readMetadataFromCsv: Reads the metadata.csv file describing the metadata for resources to be added to one of the ’Hubs’ (AnnotationHub, ExperimentHub). The file should be located in the inst/extdata of the package. `readMetadataFromCsv` performs checks for required columns and data types and can be used by package authors to validate their metadata.csv. The function is used internally by `makeAnnotationHubMetadata`.

  The rows of metadata.csv 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.
- 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.
- TaxonomyId: `character(1)`. Taxonomy ID.
- 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.
- DispatchClass: `character(1)`. Class used to determine how to load the data into R. The value for this field should be 'Rda' with the assumption that the data have been serialized with `save()` and the filename has the 'rda' extension. Contact maintainer@bioconductor.org if you data do not meet these assumptions and we can determine the most appropriate dispatch class.
- ResourceName: `character(1)`. Exact file name.
- 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`. When adding multiple 'Tags' to the metadata data.frame, protect with `I`, e.g.,
  `data.frame(Tags=I(list(c("tag1", "tag2", "tag3"))))`
  For AnnotationHub resources, 'Tags' are a free-form field of search terms defined by the user. The package name is automatically added as one of the 'Tags' before the metadata are finalized.
  For ExperimentHub resources, 'Tags' are not specified by the user but are instead automatically generated from the 'biocViews' in the DESCRIPTION. 'Tags' values supplied by the user will be ignored. This approach was taken in order to impose more structured vocabulary on the values giving the user a defined set of values on which to search. This approach may change in the future.

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 data.frame with one row per resource and columns for the Required Fields described above. Additional auto-generated columns, e.g., RDataDateAdded, RDataPath and PreparerClass may also be present and are used by internal functions when generating the final metadata.

See Also

- `makeAnnotationHubMetadata`

Examples

```
## Each resource needs a separate row of metadata. This example is for a
## single resource. If you have 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 = "3.3",
    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.
## csv
write.csv(meta, file="metadata.csv", row.names=FALSE)
## Test metadata.csv with readMetadataCsv():
readMetadataFromCsv("path/to/mypackage")

## End(Not run)
```

`updateResources` *updateResources*

Description

Add new resources to AnnotationHub

Usage

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

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

pushMetadata(allAhms, url)
```

Arguments

`AnnotationHubRoot`	Local path where files will be downloaded.
`hubroot`	Local path where files will be downloaded. (same as AnnotationHubRoot; should be renamed to match)
`BiocVersion`	A character vector of Bioconductor versions the resources should be available for.
`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`	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. If all files are successfully pushed to their
## final resting place then metadata are inserted in the AnnotationHub db.

meta <- updateResources("local/path",
                        BiocVersion = c("3.2", "3.3"),
                        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)
```

Index