Package 'AnnotationHubData'

April 22, 2016

Type Package

Title Transform public data resources into Bioconductor Data Structures

Version 1.0.2

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

Suggests RUnit, knitr,RMySQL, BiocStyle, grasp2db

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

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 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 makeGenecodeGFF.R makeGrasp2Db.R makeGSE62944ToExpressionSet.R makeHaemCode.R makeInparanoid8ToDbs.R makeNCBIToOrgDbs.R makeRecentOrgPkgsToDbs.R makePazar.R makeRefNet.R makeUCSCChain.R makeUCSC2Bit.R makeUCSCTracks.R trackWithAuxiliaryTableToGRangesRecipe.R UCSCTrackUpdateChecker.R

NeedsCompilation no

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

Maintainer Bioconductor Package Maintainer <maintainer@bioconductor.org>

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)
postProcessMetadata(ahm)
constructSeqInfo(species, genome)

inputFiles(object, ...)
outputFile(object)
ahmToJson(ahm)
deleteResources(id)
getImportPreparerClasses()
makeAnnotationHubResource(objName, makeAnnotationHubMetadataFunction,
                          ..., where)
tracksToUpdate()
updateResources(AnnotationHubRoot, BiocVersion,
                preparerClasses=getImportPreparerClasses(),
                insert=FALSE, metadataOnly=TRUE,
                justRunUnitTest=FALSE)
```

Arguments

AnnotationHubRoot

character(1) Absolute path to directory structure containing resources to be added to AnnotationHub

`SourceUrl` `character()` URL where resource(s) can be found

`SourceType` `character()` which indicates what kind of resource was initially processed. The preference is to name the type of resource if it's a single file type and to name where the resources came from if it is a compound resource. So Typical answers would be like: 'BED','FASTA' or 'Inparanoid' etc.

`SourceVersion` `character(1)` Version of original file

`SourceLastModifiedDate`
`POSIXct()` The date when the source was last modified. Leaving this blank should allow the values to be retrieved for you (if your sourceURL is valid).

`SourceMd5` `character()` md5 hash of original file

`SourceSize` `numeric(1)` Number of bytes in original file

`DataProvider` `character(1)` Where did this resource come from?

`Title` `character(1)` Title for this resource

`Description` `character(1)` Description of the resource

`Species` `character(1)` Species name

`TaxonomyId` `character(1)` NCBI code

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

`Tags` `character()` Free-form tags

`Recipe` `character(1)` Name of recipe function

`RDataClass` `character(1)` Class of derived object (e.g. 'GRanges')

`RDataDateAdded` `POSIXct()` Date added to AnnotationHub. Used to determine snapshots.

`RDataPath` `character(1)` file path to serialized form

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

`...` Additional arguments, passed directly to the class initialize method.

`BiocVersion` `character(1)` Under which resource was built

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

`DispatchClass` `character(1)` string used to indicate which code should be called by the client when the resource is downloaded. This is often the same as the RDataClass. But it is allowed to be a different value so that the client can do something different internally if required.

`Location_Prefix`
`character(1)` This was added for resources where the metadata only is stored and the resource itself comes from a third party web site. The location prefix says the base path where the resource is coming from, and the default value will be from our own site.

`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)` representing the value of `AnnotationHubRoot` to be added to the returned instance.

`originalFile` A`character(1)`

`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 (from gamay - not production)

`objName` `character(1)` The name of the preparerClass object that you intend to have be used for dispatch. You can call it whatever you want as long as you don't use an existing preparerClass name

`makeAnnotationHubMetadataFunction` `function` This is not a string, but just the name of the function that makes AHMs out of your resource of choice.

`preparerClasses` `character()` vector of preparerClasses (as strings) to try and update. Normally just one kind.

`insert` `logical()` insert into the database or no?

`metadataOnly` `logical()` process only the metadata? Or also run any associated recipe?

`justRunUnitTest` `logical()` For supporting functions, this is a flag that can help to make testing much faster, normally FALSE, but when supported, set this to TRUE so that you only process the 1st few resources to make sure that recipe works.

`where` the environment in which to store the definition. Default value is sufficient.

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")
```

`ensemblFastaToFaFile` *ensemblFastaToFaFile*

Description

Transform an Ensembl fasta file to a Bioconductor Rsamtools fa file.

Usage

```
ensemblFastaToFaFile(ahm)
```

Arguments

`ahm`	an `AnnotationHub` object.

Details

The recipe takes the source file as specified in `metadata(ahmeta)$SourceFile`, indexes the file, and returns the path of the index file.

Value

File path of the created RData file, `metadata(ahmeta)$RDataPath`.

Author(s)

Martin Morgan

See Also

ensemblGtfToGRanges, extendedBedWithAuxiliaryTableToGRanges, extendedBedToGRanges, AnnotationHubRecipe

`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 `mtmorgan@fhcrc.org`

See Also

`AnnotationHubMetadata`.

Examples

```
names(getClassDef("ImportPreparer")@subclasses)
```

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=getOption("ANNOTATION_HUB_BUCKET_NAME", "annotationhub"),
    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