Package 'AnnotationHub'

April 11, 2018

Type Package

Title Client to access AnnotationHub resources

Version 2.10.1

biocViews Infrastructure, DataImport, GUI, ThirdPartyClient

Maintainer Bioconductor Package Maintainer <maintainer@bioconductor.org>

Description This package provides a client for the Bioconductor AnnotationHub web resource. The AnnotationHub web resource provides a central location where genomic files (e.g., VCF, bed, wig) and other resources from standard locations (e.g., UCSC, Ensembl) can be discovered. The resource includes metadata about each resource, e.g., a textual description, tags, and date of modification. The client creates and manages a local cache of files retrieved by the user, helping with quick and reproducible access.

License Artistic-2.0

Depends BiocGenerics (>= 0.15.10)

Imports utils, methods, grDevices, RSQLite, BiocInstaller, curl, AnnotationDbi (>= 1.31.19), S4Vectors, interactiveDisplayBase, httr, yaml

Suggests IRanges, GenomicRanges, GenomeInfoDb, VariantAnnotation, Rsamtools, rtracklayer, BiocStyle, knitr, AnnotationForge, rBiopaxParser, RUnit, GenomicFeatures, MSnbase, mzR, Biostrings, SummarizedExperiment, ExperimentHub, gdsfmt

Enhances AnnotationHubData

Collate AnnotationHubOption.R AllGenerics.R Hub-class.R db-utils.R readMetadataFromCsv.R AnnotationHub-class.R AnnotationHubResource-class.R BEDResource-class.R ProteomicsResource-class.R EpigenomeResource-class.R EnsDbResource-class.R utilities.R sql-utils.R Hub-utils.R cache-utils.R zzz.R

VignetteBuilder knitr

NeedsCompilation yes

Author Martin Morgan [cre],
Marc Carlson [ctb],
Dan Tenenbaum [ctb],
Sonali Arora [ctb]

R topics documented:

`AnnotationHub-package` *Light-weight AnnotationHub 3.0 Client*

Description

Client for discovery and retrieval of Bioconductor annotation resources.

Author(s)

Martin Morgan mtmorgan@fhcrc.org

See Also

`AnnotationHub-class`

Examples

```
## Not run:
library(AnnotationHub)
hub = AnnotationHub()
hub

## End(Not run)
```

`AnnotationHub-objects` *AnnotationHub objects and their related methods and functions*

Description

Use `AnnotationHub` to interact with Bioconductor's AnnotationHub service. Query the instance to discover and use resources that are of interest, and then easily download and import the resource into R for immediate use.

Use `AnnotationHub()` to retrieve information about all records in the hub. If working offline, add argument `localHub=TRUE` to work with a local, non-updated hub; It will only have resources available that have previously been downloaded.

Discover records in a hub using `mcols()`, `query()`, `subset()`, `[`, and `display()`.

Retrieve individual records using `[[`. On first use of a resource, the corresponding files or other hub resources are downloaded from the internet to a local cache. On this and all subsequent uses the files are quickly input from the cache into the R session.

AnnotationHub records can be added (and sometimes removed) at any time. snapshotDate() restricts hub records to those available at the time of the snapshot. possibleDates() lists snapshot dates valid for the current version of Bioconductor.

The location of the local cache can be found (and updated) with getAnnotationHubCache and setAnnotationHubCache; removeCache removes all cache resources.

Constructors

AnnotationHub(..., hub=getAnnotationHubOption("URL"), cache=getAnnotationHubOption("CACHE'

Create an AnnotationHub instance, possibly updating the current database of records.

Accessors

In the code snippets below, x and object are AnnotationHub objects.

hubCache(x): Gets the file system location of the local AnnotationHub cache.

hubUrl(x): Gets the URL for the online hub.

length(x): Get the number of hub records.

names(x): Get the names (AnnotationHub unique identifiers, of the form AH12345) of the hub records.

fileName(x): Get the file path of the hub records as stored in the local cache (AnnotationHub files are stored as unique numbers, of the form 12345). NA is returned for those records which have not been cached.

mcols(x): Get the metadata columns describing each record. Columns include:

title Record title, frequently the file name of the object.

dataprovider Original provider of the resource, e.g., Ensembl, UCSC.

species The species for which the record is most relevant, e.g., 'Homo sapiens'.

taxonomyid NCBI taxonomy identifier of the species.

genome Genome build relevant to the record, e.g., hg19.

description Textual description of the resource, frequently automatically generated from file path and other information available when the record was created.

tags Single words added to the record to facilitate identification, e.g,. TCGA, Roadmap.

rdataclass The class of the R object used to represent the object when imported into R, e.g., GRanges, VCFFile.

sourceurl Original URL of the resource.

sourectype Format of the original resource, e.g., BED file.

dbconn(x): Return an open connection to the underyling SQLite database.

dbfile(x): Return the full path the underyling SQLite database.

.db_close(conn): Close the SQLite connection conn returned by dbconn(x).

Subsetting and related operations

In the code snippets below, x is an AnnotationHub object.

x$name: Convenient reference to individual metadata columns, e.g., x$species.

x[i]: Numerical, logical, or character vector (of AnnotationHub names) to subset the hub, e.g., x[x$species == "Homo sapiens"].

x[[i]]: Numerical or character scalar to retrieve (if necessary) and import the resource into R.

`query(x, pattern, ignore.case=TRUE, pattern.op= `&`)`: Return an AnnotationHub subset containing only those elements whose metadata matches `pattern`. Matching uses `pattern` as in `grepl` to search the `as.character` representation of each column, performing a logical `&` across columns. e.g., `query(x, c("Homo sapiens", "hg19", "GTF"))`.

`pattern` A character vector of patterns to search (via `grepl`) for in any of the `mcols()` columns.

`ignore.case` A logical(1) vector indicating whether the search should ignore case (TRUE) or not (FALSE).

`pattern.op` Any function of two arguments, describing how matches across pattern elements are to be combined. The default `&` requires that only records with *all* elements of `pattern` in their metadata columns are returned. `&`, `|` and `!` are most notably available. See `"?&"` or `?base::Ops` for more information.

`subset(x, subset)`: Return the subset of records containing only those elements whose metadata satisfies the *expression* in `subset`. The expression can reference columns of `mcols(x)`, and should return a logical vector of length `length(x)`. e.g., `subset(x, species == "Homo sapiens" & genome=="GRCh38")`.

`display(object)`: Open a web browser allowing for easy selection of hub records via interactive tabular display. Return value is the subset of hub records identified while navigating the display.

`recordStatus(hub, record)`: Returns a `data.frame` of the record id and status. `hub` must be a `Hub` object and `record` must be a `character(1)`. Can be used to discover why a resource was removed from the hub.

Cache and hub management

In the code snippets below, `x` is an AnnotationHub object.

`snapshotDate(x)` and `snapshotDate(x) <- value`: Gets or sets the date for the snapshot in use. `value` should be one of `possibleDates()`.

`possibleDates(x)`: Lists the valid snapshot dates for the version of Bioconductor that is being run (e.g., BiocInstaller::biocVersion()).

`cache(x)` and `cache(x) <- NULL`: Adds (downloads) all resources in `x`, or removes all local resources corresponding to the records in `x` from the cache. In this case, `x` would typically be a small subset of AnnotationHub resources.

`hubUrl(x)`: Gets the URL for the online AnnotationHub.

`hubCache(x)`: Gets the file system location of the local AnnotationHub cache.

`removeCache(x)`: Removes local AnnotationHub database and all related resources. After calling this function, the user will have to download any AnnotationHub resources again.

Coercion

In the code snippets below, `x` is an AnnotationHub object.

`as.list(x)`: Coerce x to a list of hub instances, one entry per element. Primarily for internal use.

`c(x, ...)`: Concatenate one or more sub-hub. Sub-hubs must reference the same AnnotationHub instance. Duplicate entries are removed.

Author(s)

Martin Morgan, Marc Carlson, Sonali Arora, and Dan Tenenbaum

Examples

```
## create an AnnotationHub object
library(AnnotationHub)
ah = AnnotationHub()

## Summary of available records
ah

## Detail for a single record
ah[1]

## and what is the date we are using?
snapshotDate(ah)

## how many resources?
length(ah)

## from which resources, is data available?
head(sort(table(ah$dataprovider), decreasing=TRUE))

## from which species, is data available ?
head(sort(table(ah$species),decreasing=TRUE))

## what web service and local cache does this AnnotationHub point to?
hubUrl(ah)
hubCache(ah)

### Examples ###

## One can  search the hub for multiple strings
ahs2 <- query(ah, c("GTF", "77","Ensembl", "Homo sapiens"))

## information about the file can be retrieved using
ahs2[1]

## one can further extract information from this show method
## like the sourceurl using:
ahs2$sourceurl
ahs2$description
ahs2$title

## We can download a file by name like this (using a list semantic):
gr <- ahs2[[1]]
## And we can also extract it by the names like this:
res <- ah[["AH28812"]]

## the gtf file is returned as a GenomicRanges object and contains
## data about which organism it belongs to, its seqlevels and seqlengths
seqinfo(gr)

## each GenomicRanges contains a metadata slot which can be used to get
## the name of the hub object and other associated metadata.
metadata(gr)
ah[metadata(gr)$AnnotationHubName]

## And we can also use "[" to restrict the things that are in the
```

```
## AnnotationHub object (by position, character, or logical vector).
## Here is a demo of position:
subHub <- ah[1:3]

if(interactive()) {
  ## Display method involves user interaction through web interface
  ah2 <- display(ah)
}

## recordStatus
recordStatus(ah, "TEST")
recordStatus(ah, "AH7220")
```

`getAnnotationHubOption` *Get and set options for default AnnotationHub behavior.*

Description

These functions get or set options for creation of new 'AnnotationHub' instances.

Usage

```
getAnnotationHubOption(arg)
setAnnotationHubOption(arg, value)
```

Arguments

`arg`	The character(1) hub options to set. see 'Details' for current options.
`value`	The value to be assigned to the hub option.

Details

Supported options include:

"URL": character(1). The base URL of the annotation hub. Default: https://annotationhub.bioconductor.org

"CACHE": character(1). The location of the hub cache. Default: ".AnnotationHub" in the user home directory.

"MAX_DOWNLOADS": numeric(1). The integer number of downloads allowed before triggering an error. This is to help avoid accidental download of a large number of AnnotationHub members.

"PROXY": `request` object returned by `httr::use_proxy()`. The `request` object describes a proxy connection allowing Internet access, usually through a restrictive firewall. Setting this option sends all AnnotationHub requests through the proxy. Default: NULL.

In `setHubOption("PROXY", value)`, `value` can be one of NULL, a `request` object returned by `httr::use_proxy()`, or a well-formed URL as character(1). The URL can be completely specified by `http://username:password@proxy.dom.com:8080`; `username:password` and port (e.g. `:8080`) are optional.

Default values may also be determined by system and global R environment variables visible *before* the package is loaded. Use options or variables preceeded by "ANNOTATION_HUB_", e.g., `options(ANNOTATION_HUB_MAX_DOWNLOADS=10)` prior to package load sets the default number of downloads to 10.

Value

The requested or successfully set option.

Author(s)

Martin Morgan mtmorgan@fhcrc.org

Examples

```
getAnnotationHubOption("URL")
## Not run:
setAnnotationHubOption("CACHE", "~/.myHub")

## End(Not run)
```

---

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

Description

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

Usage

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

Arguments

pathToPackage	Full path to data package including package name; no trailing slash.
fileName	Name of single metadata file located in "inst/extdata". If none is provided the function looks for a file named "metadata.csv".

Details

- readMetadataFromCsv:

  Reads a .csv file of metadata located in "inst/extdata". `readMetadataFromCsv` performs 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 function is used internally by `AnnotationHubData::makeAnnotationHubMetadata`.

  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.
- 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, 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.
- Location_Prefix: `character(1)`. Do not include this field if data are stored in 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(1)`.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.
- 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 data.frame with one row per resource and columns for the Required Fields described above. Additional auto-generated columns, e.g., RDataDateAdded and PreparerClass may also be present and are used by internal functions when generating the final metadata.

See Also

- makeAnnotationHubMetadata

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 with readMetadataCsv():
readMetadataFromCsv("path/to/mypackage")

## End(Not run)
```

`utilities` *Utility functions for discovering package-specific Hub resources.*

Description

List and load resources from ExperimentHub filtered by package name and optional search terms. Not Implemented for AnnotationHub.

Details

Currently `listResources` and `loadResources` are only meaningful for `ExperimentHub` objects.

Index