Package 'AnnotationHub'

April 9, 2015

Type Package

Title A client for retrieving Bioconductor objects from AnnotationHub

Version 1.6.0

Depends S4Vectors (>= 0.2.3), IRanges (>= 1.99.28)

Imports methods, stats, utils, rjson, BiocGenerics, httr, BiocInstaller (>= 1.11.0), AnnotationDbi, GenomicRanges, interactiveDisplay (>= 1.0.23)

Suggests RUnit, RCurl, Rsamtools

Author Marc Carlson, Sonali Arora

Maintainer Marc Carlson <mcarlson@fhcrc.org>

Description A client for retrieving data from the Bioconductor AnnotationHub online services.

Collate AllGenerics.R Constants.R hubUtils.R hubCache.R filter.R completion.R getResource.R AnnotationHub-class.R test_AnnotationHub_package.R dynObjHelp.R display.R zzz.R

License Artistic-2.0

biocViews Annotation, Infrastructure

LazyLoad yes

Video https://www.youtube.com/watch?v=8qvGNTVz3Ik

R topics documented:

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

Description

AnnotationHub is the base class for interacting with the AnnotationHub services using Bioconductor. When using the AnnotationHub package, users will create an instance of this class and then apply filters on it in order to narrow down their annotation options to an amount that a human can reasonably look at.

Once an AnnotationHub is created, the user can see what resources are available by tab completion using the $ argument. But before doing that, it is usually a good idea to use the filters method to set some restrictions.

What kind of restrictions? Well those can be listed by using the columns and keys methods to list them. Possible values for those can be found by using keys.

Once this is determined, a list object can be assigned using filters<-. The list needs to be named after values returned by the columns and keys methods, and to contain character vectors that contain results from the matching keys method.

All AnnotationHub have a snapshot date that by default is set to the most recent one. The snapshotDate method indicates which one is in use, but this value can also be set to previous dates.

Whenever you download a file from AnnotationHub it will automatically put the data in a local cache for future reference. The location of this cache can be found and even changed with the hubCache setters and getters. This cache provides a performance boost for users but it does not mean that you can use this on a plane without needing wireless access.

Usage

```
filters(x, ...)
filters(x, ...) <- value
hubUrl(x, ...)
hubCache(x, ...)
hubCache(x, ...) <- value
hubResource(x, path = character(), ...)
possibleDates(x, ...)
snapshotDate(x, ...)
snapshotDate(x, ...) <- value
snapshotUrl(x, ...)
snapshotUrls(x, ...)
snapshotVersion(x, ...)

columns(x)
keytypes(x)
keys(x, keytype, ...)
## S4 method for signature AnnotationHub
metadata(x, columns, ..., cols)
## S4 method for signature AnnotationHub
```

```
query(x, pattern, ...)
## S4 method for signature AnnotationHub
subset(x, subset, ...)
ahinfo(x, path)
display(object,...)
```

Arguments

x	the AnnotationHub object.
object	the AnnotationHub object.
value	the value to be assigned. For filters method, this is a named list where the names are the kind of filters, and the values are character vectors of acceptable values
path	a path string
columns, cols	character vector of desired columns. cols is deprecated; use columns instead.
keytype	the type of key you want to look up possible values for.
pattern	A character(1) regular expression to query metadata of x.
subset	An expression refering to columns of metadata(x). See Methods for additional detail.
...	other arguments

Methods

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

filters(x, ...) and filters(x, ...) <- value: Methods for listing and setting a list of filters for the AnnotationHub.

hubUrl(x, ...): Gets the URL for the main hub.

hubCache(x, ...) and hubCache(x, ...) <- value: Gets or sets the hub cache location.

possibleDates(x, ...): Lists dates for snapshots that the hub could potentially use.

snapshotDate(x) and snapshotDate(x, ...) <- value: Gets or sets the date for the snapshot in use.

snapshotUrl(x, ...): Gives the base URL for this current snapshot.

snapshotUrls(x, ...): Gives the individual URLS for all the snapshot resources.

snapshotVersion(x, ...): shows the snapshot version in use.

columns(x) and keytypes(x): shows which kinds of filters can be set for an AnnotationHub object.

keys(x, keytype, ...): Lists potential values for the filters of an AnnotationHub object. Use the keytype argument to specify which kind of filter.

metadata(x, columns, ..., cols): Retrieve metadata about all resources in the AnnotationHub x. The optional columns argument is a subset of values returned by columns(x); cols is deprecated and should not be used.

query(x, pattern, ...): 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 OR across columns. Arguments ... are passed to grepl for within-column matching.

`subset(x, subset, ...)`: Return an AnnotationHub subset containing only those elements whose metadata satisfies the *expression* in `subset`. The expression can reference columns of `metadata(x)`, and should return a logical vector of length `lenght(x)`.

`ahinfo(x, path)`: Shows metadata about an AnnotationHub resource and returns a DataFrame with that data.

`as.list(x, ...)`: Returns all the resources in the `AnnotationHub` as elements in a list object.

`display(object, ...)`: Implements the display generic from interactiveDisplay package. Users can select metadata rows from a web interface, the values in the rows selected by the user become filters applied to the AnnotationHub object.

Author(s)

Marc Carlson

Examples

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

## what is the version of this snapshot?
snapshotVersion(ah)

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

## how many resources?
length(ah)

## list currently active filters
filters(ah)

## list values that can be used to filter on:
columns(ah)
keytypes(ah)

## list possible values for one of these filter types
head(keys(ah, keytype="Species"))

## OR retrieve metadata values about several keys at once
## (This approach may not always scale the way you want it to)
metadata(ah, columns = c("Species","RDataPath"))

## create and apply a new filter to only include people
filters(ah) <- list(Species="Homo sapiens")

## now how many resources are there?
length(ah)

## what are the names for these resources?
head(names(ah))
```

```
## What are the URLs for these resources?
head(snapshotUrls(ah))

## what web service is this AnnotationHub pointing to?
hubUrl()
## and more explicitly
snapshotUrl()

## Where are the files that get downloaded being cached?
## (there is also a setter if you wish to assign this to another location)
hubCache(ah)

## Download a resource (using tab completion) and put it into "res"
res <- ah$goldenpath.hg19.encodeDCC.wgEncodeUwTfbs.wgEncodeUwTfbsMcf7CtcfStdPkRep1.narrowPeak_0.0.1.RData

## query and subset
query(ah, "GRCh37")    # GRCh37 anywhere in the metadata
subset(ah, Species == "Homo sapiens" & any(Tags == "FASTA"))

## For brevity, lets define a resource pathname we are interested in
pathName <- "goldenpath.hg19.encodeDCC.wgEncodeUwTfbs.wgEncodeUwTfbsMcf7CtcfStdPkRep1.narrowPeak_0.0.1.RData"

## We can get metadata information about that resource:
ahinfo(ah, pathName)

## Or we can extract it by name like this (using a list semantic):
res <- ah[[pathName]]

## And we can also use "[" restrict the things that are in the
## AnnotationHub object.  Either by position:
subHub <- ah[1:3]

## Or by name
subHub <- ah[pathName]

## So if I have many things to extract, I could do something like this:
pathNames <- c(pathName, "goldenpath.hg19.database.oreganno_0.0.1.RData")
res <- list()
for(i in pathNames){
  res[[i]] <- ah[[i]]
}

## OR those values can be extracted to a list like this (we recommend
## that you only download what you need)
subHub <- ah[pathNames]
res <- as.list(subHub)

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

Index