Package ‘HubPub’

July 23, 2025

Title Utilities to create and use Bioconductor Hubs

Version 1.17.1

Description HubPub provides users with functionality to help with the Bioconductor Hub structures. The package provides the ability to create a skeleton of a Hub style package that the user can then populate with the necessary information. There are also functions to help add resources to the Hub package metadata files as well as publish data to the Bioconductor S3 bucket.

License Artistic-2.0

Imports available, usethis, biocthis, dplyr, aws.s3, fs, BiocManager, utils

Suggests AnnotationHubData, ExperimentHubData, GenomeInfoDbData, testthat, knitr, rmarkdown, BiocStyle,

biocViews DataImport, Infrastructure, Software, ThirdPartyClient

BugReports `https://github.com/Bioconductor/HubPub/issues`

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

VignetteBuilder knitr

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

git_branch devel

git_last_commit aa64d97

git_last_commit_date 2025-04-30

Repository Bioconductor 3.22

Date/Publication 2025-07-23

Author Kayla Interdonato [aut, cre],
Martin Morgan [aut],
Lori Shepherd [ctb]

Maintainer Kayla Interdonato <`kayla.morrell16@gmail.com`>

Contents

`add_resource` *Add a hub resource*

Description

This function adds a hub resource to the AH or EH package metadata.csv file. It can be used while creating a new hub package or for adding data to an existing package.

Usage

```
add_resource(package, fields, metafile = "metadata.csv")
```

Arguments

`package`	A `character(1)` with the name of an existing hub package or the path to a newly created (not yet submitted/accepted) hub package.
`fields`	A named list with the data to be added to the resource. Elements and content of the list are described in `?hub_metadata`.
`metafile`	A `character(1)` with the name of the metadata csv file. The default file name is 'metadata.csv'.

Value

Path to metadata file where resource was added

Examples

```
## create a mock package
pkgdir <- tempdir()
create_pkg(file.path(pkgdir, "recordPkg"), "ExperimentHub")

## create a metadata record
meta <- hub_metadata(
    Title = "ENCODE",
    Description = "a test entry",
    BiocVersion = "4.1",
    Genome = NA_character_,
    SourceType = "JSON",
    SourceUrl = "https://www.encodeproject.org",
    SourceVersion = "x.y.z",
    Species = NA_character_,
    TaxonomyId = as.integer(9606),
    Coordinate_1_based = NA,
    DataProvider = "ENCODE Project",
    Maintainer = "tst person <tst@email.com>",
```

```
    RDataClass = "data.table",
    DispatchClass = "Rda",
    Location_Prefix = "s3://annotationhub/",
    RDataPath = "ENCODExplorerData/encode_df_lite.rda",
    Tags = "ENCODE:Homo sapiens"
)

## add the record to the metadata
add_resource(file.path(pkgdir, "recordPkg"), meta)
```

`create_pkg` *Create a Bioconductor Hub package*

Description

This function creates the skeleton of a package that follow the guidelines for Bioconductor type packages. It is expected of the user to go through and make any necessary changes or improvements once the package begins to take shape. For examples, the DESCRIPTION contains very basic requirements, but the developer should go back and fill in the 'Title:' and 'Description:' fields.

Usage

```
create_pkg(package, type = c("AnnotationHub", "ExperimentHub"), use_git = TRUE)
```

Arguments

`package`	A `character(1)` with the path of the package to be created.
`type`	A `character(1)` to indicate what type of hub package is to be created. Either `AnnotationHub` or `ExperimentHub` are acceptable.
`use_git`	A `logical(1)` indicating whether to set up `git` using `usethis::use_git()`. Default is set to TRUE.

Value

Path to package location

Examples

```
fl <- tempdir()
create_pkg(file.path(fl, "tstPkg"), "AnnotationHub")
```

`hub_metadata` *Create and validate metadata*

Description

This functions makes a list of values that can be used to add as a resource to a 'metadata.csv' file in a Hub package. The type of each argument indicates the expected value, e.g., `Title = character(1)` indicates that it is expected to be a character vector of length 1. See individual parameters for more information.

Usage

```
hub_metadata(
  Title = character(1),
  Description = character(1),
  BiocVersion = package_version("0.0"),
  Genome = character(1),
  SourceType = character(1),
  SourceUrl = character(1),
  SourceVersion = character(1),
  Species = character(1),
  TaxonomyId = integer(1),
  Coordinate_1_based = NA,
  DataProvider = character(1),
  Maintainer = character(1),
  RDataClass = character(1),
  DispatchClass = character(1),
  Location_Prefix = character(1),
  RDataPath = character(1),
  Tags = character()
)
```

Arguments

`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.
`BiocVersion`	The two-digit version of Bioconductor the resource is being introduced into. Could be a character vector `"4.1"` or an object created from `package_version()`, e.g., `package_version("4.1")`.
`Genome`	`character(1)` Name of genome build.
`SourceType`	`character(1)` Form of originial data, e.g., BED, FASTA, etc. `getValidSourceTypes()` list currently acceptable values. If nothing seems appropriate for your data reach out to maintainer@bioconductor.org.
`SourceUrl`	`character(1)` URL of originial resource(s).
`SourceVersion`	`character(1)`. A description of the version of the resource in the original source. Since source version may not follow R / Bioconductor versioning practices, this field is not restricted to a `package_version()` format.

`Species`	`character(1)` Species name. For help on valid species see `getSpeciesList`, `validSpecies`, or `suggestSpecies`.
`TaxonomyId`	`integer(1)` NCBI code. There are checks for valid taxonomyID given the Species which produce warnings. See GenomeInfoDb::loadTaxonomyDb() for full validation table.
`Coordinate_1_based`	`logical(1)` are the genomic coordinates in the resource 0-based, or 1-based? Use NA if genomic coordinates are not present in the resource.
`DataProvider`	`character(1)` Provider of original data, e.g., NCBI, UniProt etc.
`Maintainer`	`character(1)` Maintainer name and email address, `A Maintainer <URL: a. maintainer@email.c`
`RDataClass`	`character(1)` Class of derived R object, e.g., GRanges. Length must match the length of `RDataPath`.
`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.
`RDataPath`	`character(1)` 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 seperates with a colon `:`. This will link a single hub id with multiple files.
`Tags`	`character()` Zero or more tags describing the data, colon `:` separated.

Value

None

Examples

```
hub_metadata()

tst <- hub_metadata(
    Title = "ENCODE",
    Description = "a test entry",
    BiocVersion = package_version("3.9"),
    Genome = NA_character_,
    SourceType = "JSON",
```

```
    SourceUrl = "https://www.encodeproject.org",
    SourceVersion = package_version("0.0"),
    Species = NA_character_,
    TaxonomyId = NA_integer_,
    Coordinate_1_based = NA,
    DataProvider = "ENCODE Project",
    Maintainer = "tst person <tst@email.com>",
    RDataClass = "data.table",
    DispatchClass = "Rda",
    Location_Prefix = NA_character_,
    RDataPath = "ENCODExplorerData/encode_df_lite.rda",
    Tags = c("ENCODE", "Homo sapiens")
)
```

`publish_resource` *A function that publishes resource to the hub S3 bucket*

Description

This function uses functionality from the aws.s3 package to put files or directories on the Bioconductor's test hub S3 bucket. The user should have already contacted the hubs maintainers at hubs@bioconductor.org to get the necessary credentials to access the bucket. These credentials should be delcared in the system environment prior to running this function.

Usage

```
publish_resource(path, object, dry.run = TRUE)
```

Arguments

`path`	A `character(1)` path to the file or the name of the directory to be added to the bucket. If adding a directory, be sure there are no nested directories and only files within it.
`object`	A `character(1)` to indicate how the file should be named on the bucket.
`dry.run`	A boolean to indicate if the resource should in fact be published. The defalut is TRUE, meaning the resource won't be published.

Value

None

Examples

```
pkgdir <- tempfile()
fl1 <- file.path(pkgdir, "mtcars1.csv")
dir.create(dirname(fl1), recursive = TRUE)
write.csv(mtcars, file = file.path(fl1))
fl2 <- file.path(pkgdir, "mtcars2.csv")
write.csv(mtcars, file = file.path(fl2))
publish_resource(pkgdir, "test_dir")

fl3 <- file.path(pkgdir, "mtcars3.csv")
```

```
write.csv(mtcars, file = file.path(fl3))
publish_resource(fl3, "test_dir")
```

Index