

Package: DataSpaceR (via r-universe)

July 16, 2024

Type Package

Title Interface to 'the CAVD DataSpace'

Version 0.7.6

Description Provides a convenient API interface to access immunological data within 'the CAVD DataSpace'(<<https://dataspace.cavd.org>>), a data sharing and discovery tool that facilitates exploration of HIV immunological data from pre-clinical and clinical HIV vaccine studies.

URL <https://docs.ropensci.org/DataSpaceR/>,
<https://github.com/ropensci/DataSpaceR>

BugReports <https://github.com/ropensci/DataSpaceR/issues>

License GPL-3

Encoding UTF-8

Imports utils, R6, Rlabkey (>= 2.2.0), curl, httr, assertthat, digest, jsonlite, data.table

Suggests testthat, covr, knitr, pryr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.1.1

Repository <https://ropensci.r-universe.dev>

RemoteUrl <https://github.com/ropensci/DataSpaceR>

RemoteRef main

RemoteSha 38184e5a19b72c1709fd181d8dbedaa1d374ed1a

Contents

DataSpaceR-package	2
checkNetrc	2
connectDS	3
DataSpaceConnection	4

DataSpaceMab	7
DataSpaceStudy	9
getNetrcPath	12
writeNetrc	13

Index	15
--------------	-----------

DataSpaceR-package	<i>DataSpaceR</i>
--------------------	-------------------

Description

DataSpaceR provides a convenient API for accessing datasets within the DataSpace database.

Details

Uses the Rlabkey package to connect to DataSpace. Implements convenient methods for accessing datasets.

Author(s)

Ju Yeong Kim

See Also

[connectDS](#)

checkNetrc	<i>Check netrc file</i>
------------	-------------------------

Description

Check that there is a netrc file with a valid entry for the CAVD DataSpace.

Usage

```
checkNetrc(netrcFile = getNetrcPath(), onStaging = FALSE, verbose = TRUE)
```

Arguments

netrcFile	A character. File path to netrc file to check.
onStaging	A logical. Whether to check the staging server instead of the production server.
verbose	A logical. Whether to print the extra details for troubleshooting.

Value

The name of the netrc file

See Also

[connectDS writeNetrc](#)

Examples

```
try(checkNetrc())
```

connectDS

Create a connection to DataSpace

Description

Constructor for [DataSpaceConnection](#)

Usage

```
connectDS(login = NULL, password = NULL, verbose = FALSE, onStaging = FALSE)
```

Arguments

login	A character. Optional argument. If there is no netrc file a temporary one can be written by passing login and password of an active DataSpace account.
password	A character. Optional. The password for the selected login.
verbose	A logical. Whether to print the extra details for troubleshooting.
onStaging	A logical. Whether to connect to the staging server instead of the production server.

Details

Instantiates an [DataSpaceConnection](#). The constructor will try to take the values of the various `labkey.*` parameters from the global environment. If they don't exist, it will use default values. These are assigned to 'options', which are then used by the [DataSpaceConnection](#) class.

Value

an instance of [DataSpaceConnection](#)

See Also

[DataSpaceConnection](#)

Examples

```
## Not run:
con <- connectDS()

## End(Not run)

con <- try(connectDS())
if (inherits(con, "try-error")) {
  warning("Read README for more information on how to set up a .netrc file.")
}
```

DataSpaceConnection *The DataSpaceConnection class*

Description

The DataSpaceConnection class

The DataSpaceConnection class

Constructor

[connectDS](#)

Active bindings

`config` A list. Stores configuration of the connection object such as URL, path and username.

`availableStudies` A data.table. The table of available studies.

`availableGroups` A data.table. The table of available groups.

`availablePublications` A data.table. The table of available publications.

`mabGridSummary` A data.table. The filtered grid with updated `n_` columns and `geometric_mean_curve_ic50`.

`mabGrid` A data.table. The filtered mAb grid.

`virusMetadata` A data.table. Metadata about all viruses in the DataSpace.

`virusNameMappingTables` A list of data.table objects. This list contains ‘virusMetadataAll’, ‘virus-LabId’, and ‘virus_synonym’ which are described in the vignette ‘Virus_Name_Mapping_Tables’.

Methods**Public methods:**

- [DataSpaceConnection\\$new\(\)](#)
- [DataSpaceConnection\\$print\(\)](#)
- [DataSpaceConnection\\$getStudy\(\)](#)
- [DataSpaceConnection\\$getGroup\(\)](#)
- [DataSpaceConnection\\$filterMabGrid\(\)](#)
- [DataSpaceConnection\\$resetMabGrid\(\)](#)

- [DataSpaceConnection\\$getMab\(\)](#)
- [DataSpaceConnection\\$downloadPublicationData\(\)](#)
- [DataSpaceConnection\\$refresh\(\)](#)
- [DataSpaceConnection\\$clone\(\)](#)

Method `new()`: Initialize a `DataSpaceConnection` object. See [connectDS](#).

Usage:

```
DataSpaceConnection$new(  
  login = NULL,  
  password = NULL,  
  verbose = FALSE,  
  onStaging = FALSE  
)
```

Arguments:

`login` A character. Optional argument. If there is no netrc file a temporary one can be written by passing login and password of an active DataSpace account.

`password` A character. Optional. The password for the selected login.

`verbose` A logical. Whether to print the extra details for troubleshooting.

`onStaging` A logical. Whether to connect to the staging server instead of the production server.

Returns: A new 'DataSpaceConnection' object.

Method `print()`: Print the `DataSpaceConnection` object.

Usage:

```
DataSpaceConnection$print()
```

Method `getStudy()`: Create a [DataSpaceStudy](#) object.

Usage:

```
DataSpaceConnection$getStudy(studyName)
```

Arguments:

`studyName` A character. Name of the study to retrieve.

Method `getGroup()`: Create a [DataSpaceStudy](#) object.

Usage:

```
DataSpaceConnection$getGroup(groupId)
```

Arguments:

`groupId` An integer. ID of the group to retrieve.

Method `filterMabGrid()`: Filter rows in the mAb grid by specifying the values to keep in the columns found in the `mabGrid` field. It takes the column and the values and filters the underlying tables.

Usage:

```
DataSpaceConnection$filterMabGrid(using, value)
```

Arguments:

using A character. Name of the column to filter.
 value A character vector. Values to keep in the mAb grid.

Method resetMabGrid(): Reset the mAb grid to the unfiltered state.

Usage:

```
DataSpaceConnection$resetMabGrid()
```

Method getMab(): Create a [DataSpaceMab](#) object.

Usage:

```
DataSpaceConnection$getMab()
```

Method downloadPublicationData(): Download publication data for a chosen publication.

Usage:

```
DataSpaceConnection$downloadPublicationData(
  publicationId,
  outputDir = getwd(),
  unzip = TRUE,
  verbose = TRUE
)
```

Arguments:

publicationId A character/integer. ID for the publication to download data for.
 outputDir A character. Path to directory to download publication data.
 unzip A logical. If TRUE, unzip publication data to outputDir.
 verbose A logical. Default TRUE.

Method refresh(): Refresh the connection object to update available studies and groups.

Usage:

```
DataSpaceConnection$refresh()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
DataSpaceConnection$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

[connectDS DataSpaceR-package](#)

Examples

```
## Not run:
# Create a connection (Initiate a DataSpaceConnection object)
con <- connectDS()
con
```

```
# Connect to cvd408
# https://dataspace.cavd.org/cds/CAVD/app.view#learn/learn/Study/cvd408?q=408
cvd408 <- con$getStudy("cvd408")

# Connect to all studies
cvd <- con$getStudy("cvd408")

# Connect to the NYVAC durability comparison group
# https://dataspace.cavd.org/cds/CAVD/app.view#group/groupsummary/220
nyvac <- con$getGroup(220)

# Refresh the connection object to update available studies and groups
con$refresh()

## End(Not run)
```

DataSpaceMab

The DataSpaceMab class

Description

The DataSpaceMab class

The DataSpaceMab class

Constructor

DataSpaceConnection\$getMab()

Active bindings

config A list. Stores configuration of the connection object such as URL, path and username.

studyAndMabs A data.table. The table of available mAbs by study.

mabs A data.table. The table of available mAbs and their attributes.

nabMab A data.table. The table of mAbs and their neutralizing measurements against viruses.

studies A data.table. The table of available studies.

assays A data.table. The table of assay status by study.

variableDefinitions A data.table. The table of variable definitions.

Methods

Public methods:

- [DataSpaceMab\\$new\(\)](#)
- [DataSpaceMab\\$print\(\)](#)
- [DataSpaceMab\\$refresh\(\)](#)
- [DataSpaceMab\\$getLanlMetadata\(\)](#)

- [DataSpaceMab\\$clone\(\)](#)

Method `new()`: Initialize DataSpaceMab object. See [DataSpaceConnection](#).

Usage:

```
DataSpaceMab$new(mabMixture, filters, config)
```

Arguments:

`mabMixture` A character vector.

`filters` A list.

`config` A list.

Method `print()`: Print the DataSpaceMab object summary.

Usage:

```
DataSpaceMab$print()
```

Method `refresh()`: Refresh the DataSpaceMab object to update datasets.

Usage:

```
DataSpaceMab$refresh()
```

Method `getLanlMetadata()`: Applies LANL metadata to mabs table.

Usage:

```
DataSpaceMab$getLanlMetadata()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
DataSpaceMab$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

[connectDS](#) [DataSpaceConnection](#)

Examples

```
## Not run:
# Create a connection (Initiate a DataSpaceConnection object)
con <- connectDS()

# Browse the mAb Grid
con$mabGridSummary

# Filter the grid by viruses
con$filterMabGrid(using = "virus", value = c("242-14", "Q23.17", "6535.3", "BaL.26", "DJ263.8"))

# Filter the grid by donor species (llama)
con$filterMabGrid(using = "donor_species", value = "llama")
```



```

# Check the updated grid
con$mabGridSummary

# Retrieve available viruses in the filtered grid
con$mabGrid[, unique(virus)]

# Retrieve available clades for 1H9 mAb mixture in the filtered grid
con$mabGrid[mab_mixture == "1H9", unique(clade)]

# Create a DataSpaceMab object that contains the filtered mAb data
mab <- con$getMab()
mab

# Inspect the `nabMab` field
mab$nabMab

## End(Not run)

```

DataSpaceStudy

The DataSpaceStudy class

Description

The DataSpaceStudy class

The DataSpaceStudy class

Constructor

DataSpaceConnection\$getStudy() DataSpaceConnection\$getGroup()

Active bindings

`study` A character. The study name.

`config` A list. Stores configuration of the connection object such as URL, path and username.

`availableDatasets` A data.table. The table of datasets available in the DataSpaceStudy object.

`cache` A list. Stores the data to avoid downloading the same tables multiple times.

`dataDir` A character. Default directory for storing nonstandard datasets. Set with `setDataDir(dataDir)`.

`treatmentArm` A data.table. The table of treatment arm information for the connected study. Not available for all study connection.

`group` A character. The group name.

`studyInfo` A list. Stores the information about the study.

Methods

Public methods:

- [DataSpaceStudy\\$new\(\)](#)
- [DataSpaceStudy#print\(\)](#)
- [DataSpaceStudy\\$getDataset\(\)](#)
- [DataSpaceStudy\\$clearCache\(\)](#)
- [DataSpaceStudy\\$getDatasetDescription\(\)](#)
- [DataSpaceStudy\\$setDataDir\(\)](#)
- [DataSpaceStudy\\$refresh\(\)](#)
- [DataSpaceStudy\\$clone\(\)](#)

Method `new()`: Initialize `DataSpaceStudy` class. See [DataSpaceConnection](#).

Usage:

```
DataSpaceStudy$new(study = NULL, config = NULL, group = NULL, studyInfo = NULL)
```

Arguments:

`study` A character. Name of the study to retrieve.

`config` A list. Stores configuration of the connection object such as URL, path and username.

`group` An integer. ID of the group to retrieve.

`studyInfo` A list. Stores the information about the study.

Method `print()`: Print `DataSpaceStudy` class.

Usage:

```
DataSpaceStudy#print()
```

Method `getDataset()`: Get a dataset from the connection.

Usage:

```
DataSpaceStudy$getDataset(
  datasetName,
  mergeExtra = FALSE,
  colFilter = NULL,
  reload = FALSE,
  outputDir = NULL,
  ...
)
```

Arguments:

`datasetName` A character. Name of the dataset to retrieve. Accepts the value in either the "name" or "label" field from `availableDatasets`.

`mergeExtra` A logical. If set to `TRUE`, merge extra information. Ignored for non-integrated datasets.

`colFilter` A matrix. A filter as returned by `Rlabkey`'s `makeFilter`.

`reload` A logical. If set to `TRUE`, download the dataset, whether a cached version exist or not.

`outputDir` A character. Optional, specifies directory to download nonstandard datasets. If `NULL`, data will be downloaded to `dataDir`, set with `setDataDir(dataDir)`. If `dataDir` is not set, and `outputDir` is `NULL`, a `tmp` directory will be used.

... Extra arguments to be passed to [labkey.selectRows](#)

Method `clearCache()`: Clear cache. Remove downloaded datasets.

Usage:

```
DataSpaceStudy$clearCache()
```

Method `getDatasetDescription()`: Get variable information.

Usage:

```
DataSpaceStudy$getDatasetDescription(datasetName, outputDir = NULL)
```

Arguments:

`datasetName` A character. Name of the dataset to retrieve. Accepts the value in either the "name" or "label" field from `availableDatasets`.

`outputDir` A character. Directory path.

Method `setDataDir()`: Set default directory to download non-integrated datasets. If no `dataDir` is set, a `tmp` directory will be used.

Usage:

```
DataSpaceStudy$setDataDir(dataDir)
```

Arguments:

`dataDir` A character. Directory path.

Method `refresh()`: Refresh the study object to update available datasets and treatment info.

Usage:

```
DataSpaceStudy$refresh()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
DataSpaceStudy$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

[connectDS](#) [DataSpaceConnection](#)

Examples

```
## Not run:
# Create a connection (Initiate a DataSpaceConnection object)
con <- connectDS()

# Connect to cvd408 (Initiate a DataSpaceStudy object)
# https://dataspace.cavd.org/cds/CAVD/app.view#learn/learn/Study/cvd408?q=408
cvd408 <- con$getStudy("cvd408")
cvd408
```

```
# Retrieve Neutralizing antibody dataset (NAb) for cvd408 from DataSpace
NAb <- cvd408$getDataset("NAb")

# Get variable information of the NAb dataset
cvd408$getDatasetDescription("NAb")

# Take a look at cvd408's treatment arm information
cvd408$treatmentArm

# Clear cache of a study object
cvd408$clearCache()

# Connect to the NYVAC durability comparison group
# https://dataspace.cavd.org/cds/CAVD/app.view#group/groupsummary/220
nyvac <- con$getGroup(220)

# Connect to all studies
cvd <- con$getStudy("")

# Refresh the study object to update available datasets and treatment info
cvd$refresh()

## End(Not run)
```

getNetrcPath

Get a default netrc file path

Description

Get a default netrc file path

Usage

```
getNetrcPath()
```

Value

A character vector containing the default netrc file path

Examples

```
getNetrcPath()
```

writeNetrc	<i>Write a netrc file</i>
------------	---------------------------

Description

Write a netrc file that is valid for accessing DataSpace.

Usage

```
writeNetrc(  
  login,  
  password,  
  netrcFile = NULL,  
  onStaging = FALSE,  
  overwrite = FALSE  
)
```

Arguments

login	A character. Email address used for logging in on DataSpace.
password	A character. Password associated with the login.
netrcFile	A character. Credentials will be written into that file. If left NULL, netrc will be written into a temporary file.
onStaging	A logical. Whether to connect to the staging server instead of the production server.
overwrite	A logical. Whether to overwrite the existing netrc file.

Details

The database is accessed with the user's credentials. A netrc file storing login and password information is required. See [here](#) for instruction on how to register and set DataSpace credential. By default curl will look for the file in your home directory.

Value

A character vector containing the netrc file path

See Also

[connectDS checkNetrc](#)

Examples

```
# First, create an account in the DataSpace App and read the terms of use
# Next, create a netrc file using writeNetrc()
writeNetrc(
  login = "dataspaceuser@email.com",
  password = "yourSecretPassword"
)
# Specify `netrcFile = getNetrcPath()` to write netrc in the default path
```

Index

`checkNetrc`, [2](#), [13](#)

`connectDS`, [2](#), [3](#), [3](#), [4–6](#), [8](#), [11](#), [13](#)

`DataSpaceConnection`, [3](#), [4](#), [8](#), [10](#), [11](#)

`DataSpaceMab`, [6](#), [7](#)

`DataSpaceR` (`DataSpaceR`-package), [2](#)

`DataSpaceR`-package, [2](#)

`DataSpaceStudy`, [5](#), [9](#)

`getNetrcPath`, [12](#)

`labkey.selectRows`, [11](#)

`makeFilter`, [10](#)

`writeNetrc`, [3](#), [13](#)