

# Package ‘DataSpaceR’

January 8, 2020

**Type** Package

**Title** Interface to 'the CAVD DataSpace'

**Version** 0.7.3

**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

**LazyData** true

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

**Suggests** testthat, covr, knitr, pryr, rmarkdown, DT

**VignetteBuilder** knitr

**RoxygenNote** 7.0.2

**NeedsCompilation** no

**Author** Ju Yeong Kim [aut, cre, cph],  
Sean Hughes [rev] (Sean reviewed the package for ropensci, see  
<<https://github.com/ropensci/software-review/issues/261>>),  
Jason Taylor [aut],  
Helen Miller [aut]

**Maintainer** Ju Yeong Kim <[jkim2345@fredhutch.org](mailto:jkim2345@fredhutch.org)>

**Repository** CRAN

**Date/Publication** 2020-01-08 08:10:02 UTC

## R topics documented:

DataSpaceR-package . . . . .	2
checkNetrc . . . . .	2
connectDS . . . . .	3
DataSpaceConnection . . . . .	4
DataSpaceMab . . . . .	7
DataSpaceStudy . . . . .	9
getNetrcPath . . . . .	12
writeNetrc . . . . .	12

<b>Index</b>	<b>14</b>
--------------	-----------

---

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

**Value**

an instance of DataSpaceConnection

**Constructor**

[connectDS](#)

**Fields**

`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.

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

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

**Methods**

`initialize(login = NULL, password = NULL, verbose = FALSE, onStaging = FALSE)` Initialize a DataSpaceConnection object. See [connectDS](#).

`print()` Print the DataSpaceConnection object.

`getStudy(study, groupId = NULL)` Create a [DataSpaceStudy](#) object.

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

`getGroup(groupId)` Create a [DataSpaceStudy](#) object.  
    *groupId*: An integer. ID of the group to retrieve.

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

`filterMabGrid(using, value)` 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.  
    *using*: A character. Name of the column to filter.  
    *value*: A character vector. Values to keep in the mAb grid.

`getMab()` Create a [DataSpaceMab](#) object.

`resetMabGrid()` Reset the mAb grid to the unfiltered state.

## Methods

### Public methods:

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

### Method `new()`:

*Usage:*

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

### Method `print()`:

*Usage:*

```
DataSpaceConnection$print()
```

### Method `getStudy()`:

*Usage:*

```
DataSpaceConnection$getStudy(study)
```

### Method `getGroup()`:

*Usage:*

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

**Method** filterMabGrid():*Usage:*

DataSpaceConnection\$filterMabGrid(using, value)

**Method** resetMabGrid():*Usage:*

DataSpaceConnection\$resetMabGrid()

**Method** getMab():*Usage:*

DataSpaceConnection\$getMab()

**Method** refresh():*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	<i>The DataSpaceMab class</i>
--------------	-------------------------------

---

### Description

The DataSpaceMab class

The DataSpaceMab class

### Value

an instance of DataSpaceMab

### Constructor

DataSpaceConnection\$getMab()

### Fields

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

initialize(mabMixture, filters, config) Initialize DataSpaceMab object. See [DataSpaceConnection](#).

print() Print DataSpaceMab object summary.

refresh() Refresh the mab object to update datasets.

### Methods

#### Public methods:

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

#### Method new():

*Usage:*

DataSpaceMab\$new(mabMixture, filters, config)

**Method print():***Usage:*

DataSpaceMab\$print()

**Method refresh():***Usage:*

DataSpaceMab\$refresh()

**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

### Value

an instance of DataSpaceStudy

### Constructor

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

### Fields

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 study object.

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

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.

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

### Methods

initialize(study = NULL, config = NULL, group = NULL, studyInfo = NULL) Initialize DataSpaceStudy class. See [DataSpaceConnection](#).

print() Print DataSpaceStudy class.

getDataset(datasetName, mergeExtra = FALSE, colFilter = NULL, reload = FALSE, outputDir = NULL, ...) Get a dataset from the connection.

datasetName: A character. Name of the dataset to retrieve.

mergeExtra: A logical. If set to TRUE, merge extra information.

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](#)

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

`getDatasetDescription(datasetName)` Get variable information.  
datasetName: A character. Name of the dataset to retrieve.

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

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

## Methods

### Public methods:

- `DataSpaceStudy$new()`
- `DataSpaceStudy$print()`
- `DataSpaceStudy$getDataset()`
- `DataSpaceStudy$clearCache()`
- `DataSpaceStudy$getDatasetDescription()`
- `DataSpaceStudy$refresh()`
- `DataSpaceStudy$setDataDir()`
- `DataSpaceStudy$clone()`

### Method `new()`:

*Usage:*

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

### Method `print()`:

*Usage:*

```
DataSpaceStudy$print()
```

### Method `getDataset()`:

*Usage:*

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

### Method `clearCache()`:

*Usage:*

```
DataSpaceStudy$clearCache()
```

### Method `getDatasetDescription()`:

*Usage:*

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

**Method** refresh():

*Usage:*

```
DataSpaceStudy$refresh()
```

**Method** setDataDir():

*Usage:*

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

**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	<i>Get a default netrc file path</i>
--------------	--------------------------------------

---

### 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](#), [4](#), [7–9](#), [11](#)

`DataSpaceMab`, [5](#), [7](#)

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

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

`DataSpaceStudy`, [4](#), [5](#), [9](#)

`getNetrcPath`, [12](#)

`labkey.selectRows`, [9](#)

`makeFilter`, [9](#)

`writeNetrc`, [3](#), [12](#)