`Rlabkey-package` *Import/export data between a labkey database and R*

Description

This package allows the transfer of data between a labkey database and an R session. Data can be imported from a labkey database into R by specifying the query schema information (`labkey.selectRows`) or by using sql commands (`labkey.executeSql`). From an R session, existing data can be updated (`labkey.updateRows`), new data can be inserted (`labkey.insertRows`) or data can be deleted from the labkey database (`labkey.deleteRows`).

The user must have the appropriate authorization on the labkey server in order to modify the database through the use of these functions.

Details

Package:	Rlabkey
Type:	Package
Version:	0.0.4
Date:	2008-10-27
License:	Apache License 2.0
LazyLoad:	yes

Using this package to access a password protected labkey data base requires that the user has their login information in a netrc file. The netrc file contains configuration and autologin information for the File Transfer Protocol client (ftp) and other programs such as CURL.

On a UNIX system this file should be named .netrc (dot netrc) and on windows it sould be named _netrc (underscore netrc). The file should be located in the users home directory and the permissions on the file should be unreadable for everybody except the owner.

To create the _netrc on a windows machine, first create an environment variable called 'HOME' that is set to your home directory (c:/Users/<User-Name> on Vista) or any directory you want to use. In that directory, create a text file named _netrc (note that it's underscore netrc, not dot netrc like it is on UNIX).

The following three lines must be included in the .netrc or _netrc file either separated by white space (spaces, tabs, or newlines) or commas.

machine <remote-machine-name>
login <user-email>
password <user-password>

One example would be:
machine atlas.scharp.org
login vobencha@fhcrc.org
password mypassword

Another example would be:
machine atlas.scharp.org login vobencha@fhcrc.org password mypassword

Author(s)

Valerie Obenchain

References

http://www.omegahat.org/RCurl/,
http://dssm.unipa.it/CRAN/web/packages/rjson/rjson.pdf,
https://www.labkey.org/project/home/begin.view

See Also

`labkey.selectRows`, `labkey.executeSql`, `makeFilter`, `labkey.insertRows`, `labkey.updateRows`, `labkey.deleteRows`

`labkey.deleteRows` *Delete rows of data from a labkey database*

Description

Specify rows of data to be deleted from the database.

Usage

```
labkey.deleteRows(baseUrl, folderPath, schemaName, queryName, toDelete,
stripAllHidden = TRUE)
```

Arguments

`baseUrl`	a string specifying the `baseUrl`for labkey server
`folderPath`	a string specifying the `folderPath`
`schemaName`	a string specifying the `schemaName` for the query
`queryName`	a string specifying the `queryName`
`toDelete`	a data frame containing a single column of data containing the data identifiers of the rows to be deleted
`stripAllHidden`	(optional) a logical value indicating whether or not to return data columns that would normally be hidden from user veiw. If no value is specified, no hidden columns are returned.

Details

A single row or multiple rows of data can be deleted. The `toDelete` data frame should consist of a single column of data containing the data identifiers of the rows to be deleted (e.g., key or lsid). The name of the data in the data frame must be the column name from the labkey database. The data frame must be created with the `stringsAsFactors` set to FALSE.

NOTE: Each variable in a dataset has both a column label and a column name. The column label is visible at the top of each column on the web page and is longer and more descriptive. The column name is shorter and is used "behind the scenes" for database manipulation. It is the column name that must be used in the Rlabkey functions when a column name is expected. To identify a particular column name in a dataset on a web site, use the "export to R script" option available as a drop down option under the "views" tab for each dataset.

Value

A list is returned with named categories of **command**, **rowsAffected**, **rows**, **queryName**, **containerPath** and **schemaName**. The **schemaName**, **queryName** and **containerPath** properties contain the same schema, query and folder path used in the request. The **rowsAffected** property indicates he number of rows affected by the API action. This will typically be the same number as passed in the request. The **rows** property contains a list of rows corresponding to the rows deleted.

Author(s)

Valerie Obenchain

References

http://www.omegahat.org/RCurl/,
http://dssm.unipa.it/CRAN/web/packages/rjson/rjson.pdf,
https://www.labkey.org/project/home/begin.view

See Also

`labkey.selectRows`, `labkey.executeSql`, `makeFilter`, `labkey.insertRows`, `labkey.updateRows`

Examples

```
## Insert, update and delete
# Note that users must have the necessary permissions in the database
# to be able to modify data through the use of these functions

### Not run
#newrow <- data.frame(name="Frank", age=11, stringsAsFactors=FALSE)
#labkey.insertRows(
#baseUrl="https://atlas-test.scharp.org/cpas",
#folderPath="/CHAVI/Analysis/vobencha",
#schemaName="lists",
#queryName="testlist",
#toInsert=newrow)
#
#updaterow=data.frame(Key=4,name="Patty",age=11, stringsAsFactors=FALSE)
```

```
#mydata <- labkey.updateRows(
#baseUrl="https://atlas-test.scharp.org/cpas",
#folderPath="/CHAVI/Analysis/vobencha",
#schemaName="lists",
#queryName="testlist",
#toUpdate=updaterow)
#
#deleterow <- data.frame(Key=1, stringsAsFactors=FALSE)
#mydata <- labkey.deleteRows(
#baseUrl="https://atlas-test.scharp.org/cpas",
#folderPath="/CHAVI/Analysis/vobencha",
#schemaName="lists",
#queryName="testlist",
#toDelete=deleterow)
```

`labkey.executeSql` *Retrieve data from a labkey database using SQL commands*

Description

Use Sql commands to specify data to be imported into R. Prior to import, data can be manipulated through standard SQL commands supported in labkey SQL.

Usage

```
labkey.executeSql(baseUrl, folderPath, schemaName, sql, maxRows = NULL,
rowOffset = NULL, stripAllHidden = TRUE)
```

Arguments

`baseUrl`	a string specifying the `baseUrl`for the labkey server
`folderPath`	a string specifying the `folderPath`
`schemaName`	a string specifying the `schemaName` for the query
`sql`	a string containing the `sql` commands to be executed
`maxRows`	(optional) an integer specifying the maximum number of rows to return. If no value is specified, all rows are returned.
`rowOffset`	(optional) an integer specifying which row of data should be the first row in the retrieval. If no value is specified, rows will begin at the start of the result set.
`stripAllHidden`	(optional) a logical value indicating whether or not to return data columns that would normally be hidden from user veiw. If no value is specified, no hidden columns are returned.

Details

A full dataset or any portion of a dataset can be imported into an R data frame using the `labkey.executeSql` function. Function arguments are components of the url that identify the location of the data and the SQL actions that should be taken on the data prior to import.

NOTE: Each variable in a dataset has both a column label and a column name. The column label is visible at the top of each column on the web page and is longer and more descriptive. The column name is shorter and is used "behind the scenes" for database manipulation. It is the column name that must be used in the Rlabkey functions when a column name is expected. To identify a particular column name in a dataset on a web site, use the "export to R script" option available as a drop down option under the "views" tab for each dataset.

Value

The requested data are returned in a data frame with column names as they appear on the website.

Author(s)

Valerie Obenchain

References

http://www.omegahat.org/RCurl/,
http://dssm.unipa.it/CRAN/web/packages/rjson/rjson.pdf,
https://www.labkey.org/project/home/begin.view

See Also

`labkey.selectRows`, `makeFilter`, `labkey.insertRows`, `labkey.updateRows`, `labkey.deleteRows`

Examples

```
## These example datasets are located at https://www.labkey.org/project/home/Study/demo/begi

## Select participants who meet acute status requirements
getacute <- labkey.executeSql(
baseUrl="https://www.labkey.org",
folderPath="/home/Study/demo",
schemaName="study",
sql = 'select "Status Assessment".ParticipantId, "Status Assessment".StatusMeetCriteria
from "Status Assessment" where "Status Assessment".StatusMeetCriteria=\'Yes\'')

## Compute average ages over different gender groups, use column alias "Number" to rename th
getage <- labkey.executeSql(
baseUrl="https://www.labkey.org",
folderPath="/home/Study/demo",
schemaName="study",
sql = "select Demographics.Gender, avg(Demographics.Age) as Number from Demographics
group by Demographics.Gender")

## Get a list of participants with partner information
```

```
getpartners <- labkey.executeSql(
baseUrl="https://www.labkey.org",
folderPath="/home/Study/demo",
schemaName="study",
sql = 'select "Status Assessment".ParticipantID, "Status Assessment".StatusPartner1
from "Status Assessment" where "Status Assessment".StatusPartner1 is not null')
```

`labkey.insertRows` *Insert new rows of data into a labkey database*

Description

Insert new rows of data into the database.

Usage

```
labkey.insertRows(baseUrl, folderPath, schemaName, queryName, toInsert,
stripAllHidden = TRUE)
```

Arguments

`baseUrl`	a string specifying the `baseUrl`for the labkey server
`folderPath`	a string specifying the `folderPath`
`schemaName`	a string specifying the `schemaName` for the query
`queryName`	a string specifying the `queryName`
`toInsert`	a data frame containing rows of data to be inserted
`stripAllHidden`	(optional) a logical value indicating whether or not to return data columns that would normally be hidden from user veiw. If no value is specified, no hidden columns are returned.

Details

A single row or multiple rows of data can be inserted. The `toInsert` data frame must contain values for each column in the dataset and must be created with the `stringsAsFactors` option set to FALSE. The names of the data in the data frame must be the column names from the labkey database. When inserting data into a study dataset, the sequence number must be specified.

NOTE: Each variable in a dataset has both a column label and a column name. The column label is visible at the top of each column on the web page and is longer and more descriptive. The column name is shorter and is used "behind the scenes" for database manipulation. It is the column name that must be used in the Rlabkey functions when a column name is expected. To identify a particular column name in a dataset on a web site, use the "export to R script" option available as a drop down option under the "views" tab for each dataset.

Value

A list is returned with named categories of **command**, **rowsAffected**, **rows**, **queryName**, **containerPath** and **schemaName**. The **schemaName**, **queryName** and **containerPath** properties contain the same schema, query and folder path used in the request. The **rowsAffected** property indicates he number of rows affected by the API action. This will typically be the same number as passed in the request. The **rows** property contains a list of row objects corresponding to the rows inserted.

Author(s)

Valerie Obenchain

References

http://www.omegahat.org/RCurl/,
http://dssm.unipa.it/CRAN/web/packages/rjson/rjson.pdf,
https://www.labkey.org/project/home/begin.view

See Also

labkey.selectRows, labkey.executeSql, makeFilter, labkey.updateRows, labkey.deleteRows

Examples

```
## Insert, update and delete
# Note that users must have the necessary permissions in the database
# to be able to modify data through the use of these functions

### Not run
#newrow <- data.frame(name="Frank", age=11, stringsAsFactors=FALSE)
#labkey.insertRows(
#baseUrl="https://atlas-test.scharp.org/cpas",
#folderPath="/CHAVI/Analysis/vobencha",
#schemaName="lists",
#queryName="testlist",
#toInsert=newrow)
#
#updaterow=data.frame(Key=4,name="Patty",age=11, stringsAsFactors=FALSE)
#mydata <- labkey.updateRows(
#baseUrl="https://atlas-test.scharp.org/cpas",
#folderPath="/CHAVI/Analysis/vobencha",
#schemaName="lists",
#queryName="testlist",
#toUpdate=updaterow)
#
#deleterow <- data.frame(Key=1, stringsAsFactors=FALSE)
#mydata <- labkey.deleteRows(
#baseUrl="https://atlas-test.scharp.org/cpas",
#folderPath="/CHAVI/Analysis/vobencha",
#schemaName="lists",
#queryName="testlist",
```

```
#toDelete=deleterow)
```

`labkey.selectRows` *Retrieve data from a labkey database*

Description

Import full datasets or selected rows into R. The data can be sorted and filtered prior to import.

Usage

```
labkey.selectRows(baseUrl, folderPath, schemaName, queryName, viewName = NULL,
colSelect = NULL, maxRows = NULL, rowOffset = NULL, colSort = NULL,
colFilter = NULL, stripAllHidden = TRUE)
```

Arguments

`baseUrl`	a string specifying the `baseUrl`for the labkey server
`folderPath`	a string specifying the `folderPath`
`schemaName`	a string specifying the `schemaName` for the query
`queryName`	a string specifying the `queryName`
`viewName`	(optional) a string specifying the `viewName`
`colSelect`	(optional) a vector of comma separated strings specifying which columns of a dataset or view to import
`maxRows`	(optional) an integer specifying how many rows of data to return. If no value is specified, all rows are returned.
`colSort`	(optional) a string including the name of the column to sort preceeded by a "+" or "-" to indicate sort direction
`rowOffset`	(optional) an integer specifying which row of data should be the first row in the retrieval. If no value is specified, the retrieval starts with the first row.
`colFilter`	(optional) a vector or array object created by the `makeFilter` function which contains the column name, operator and value of the filter(s) to be applied to the retrieved data.
`stripAllHidden`	(optional) a logical value indicating whether or not to return data columns that would normally be hiddenfrom user view. If no value is specified, no hidden columns are returned.

Details

A full dataset or any portion of a dataset can be imported into an R data frame using the `labkey.selectRows` function. Function arguments are the components of the url that identify the location of the data and what actions should be taken on the data prior to import (ie, sorting, selecting particular columns or maximum number of rows, etc.).

NOTE: Each variable in a dataset has both a column label and a column name. The column label is visible at the top of each column on the web page and is longer and more descriptive. The column name is shorter and is used "behind the scenes" for database manipulation. It is the column name that must be used in the Rlabkey functions when a column name is expected. To identify a particular column name in a dataset on a web site, use the "export to R script" option available as a drop down option under the "views" tab for each dataset.

Value

The requested data are returned in a data frame with column names as they appear on the website.

Author(s)

Valerie Obenchain

References

http://www.omegahat.org/RCurl/,
http://dssm.unipa.it/CRAN/web/packages/rjson/rjson.pdf,
https://www.labkey.org/project/home/begin.view

See Also

`labkey.executeSql`, `makeFilter`, `labkey.insertRows`, `labkey.updateRows`, `labkey.deleteRows`

Examples

```
## These example datasets are located at https://www.labkey.org/project/home/Study/demo/begi

## Retrieve full HIV Test Results dataset
fulldata <- labkey.selectRows(
baseUrl="https://www.labkey.org",
folderPath="/home/Study/demo",
schemaName="study",
queryName="HIV Test Results")

## Specifying filters, max rows and selecting columns
myfilters<- makeFilter(c("HIVLoadQuant","GREATER_THAN",500), c("HIVRapidTest","EQUALS","Posi
smalldata <- labkey.selectRows(
baseUrl="https://www.labkey.org",
folderPath="/home/Study/demo",
schemaName="study",
queryName="HIV Test Results",
colSelect=c("ParticipantId","HIVDate","HIVLoadQuant","HIVRapidTest"),
maxRows=20,
```

```
colFilter=myfilters)
```

`labkey.updateRows` *Update existing rows of data in a labkey database*

Description

Send data from an R session to update existing rows of data in the database.

Usage

```
labkey.updateRows(baseUrl, folderPath, schemaName, queryName, toUpdate,
stripAllHidden = TRUE)
```

Arguments

`baseUrl`	a string specifying the `baseUrl`for the labkey server
`folderPath`	a string specifying the `folderPath`
`schemaName`	a string specifying the `schemaName`for the query
`queryName`	a string specifying the `queryName`
`toUpdate`	a data frame containing the row(s) of data to be updated
`stripAllHidden`	(optional) a logical value indicating whether or not to return data columns that would normally be hidden from user veiw. If no value is specified, no hidden columns are returned.

Details

A single row or multiple rows of data can be updated. The `toUpdate` data frame should contain the rows of data to be updated and must be created with the `stringsAsFactors` option set to FALSE. The names of the data in the data frame must be the column names from the labkey database.

NOTE: Each variable in a dataset has both a column label and a column name. The column label is visible at the top of each column on the web page and is longer and more descriptive. The column name is shorter and is used "behind the scenes" for database manipulation. It is the column name that must be used in the Rlabkey functions when a column name is expected. To identify a particular column name in a dataset on a web site, use the "export to R script" option available as a drop down option under the "views" tab for each dataset.

Value

A list is returned with named categories of **command**, **rowsAffected**, **rows**, **queryName**, **containerPath** and **schemaName**. The **schemaName**, **queryName** and **containerPath** properties contain the same schema, query and folder path used in the request. The **rowsAffected** property indicates he number of rows affected by the API action. This will typically be the same number as passed in the request. The **rows** property contains a list of row objects corresponding to the rows updated.

Author(s)

Valerie Obenchain

References

http://www.omegahat.org/RCurl/,
http://dssm.unipa.it/CRAN/web/packages/rjson/rjson.pdf,
https://www.labkey.org/project/home/begin.view

See Also

labkey.selectRows, labkey.executeSql, makeFilter, labkey.insertRows, labkey.deleteRows

Examples

```
## Insert, update and delete
# Note that users must have the necessary permissions in the database
# to be able to modify data through the use of these functions

### Not run
#newrow <- data.frame(name="Frank", age=11, stringsAsFactors=FALSE)
#labkey.insertRows(
#baseUrl="https://atlas-test.scharp.org/cpas",
#folderPath="/CHAVI/Analysis/vobencha",
#schemaName="lists",
#queryName="testlist",
#toInsert=newrow)
#
#updaterow=data.frame(Key=4,name="Patty",age=11, stringsAsFactors=FALSE)
#mydata <- labkey.updateRows(
#baseUrl="https://atlas-test.scharp.org/cpas",
#folderPath="/CHAVI/Analysis/vobencha",
#schemaName="lists",
#queryName="testlist",
#toUpdate=updaterow)
#
#deleterow <- data.frame(Key=1, stringsAsFactors=FALSE)
#mydata <- labkey.deleteRows(
#baseUrl="https://atlas-test.scharp.org/cpas",
#folderPath="/CHAVI/Analysis/vobencha",
#schemaName="lists",
#queryName="testlist",
#toDelete=deleterow)
```

---

makeFilter *Builds filters to be used in labkey.selectRows*

Description

This function takes inputs of column name, filter value and filter operator and returns an array of filters to be used in `labkey.selectRows`.

Usage

```
makeFilter(...)
```

Arguments

...	Arguments in c("colname","operator","value") form, used to create a filter.

Details

These filters are applied to the data prior to import into R. The user can specify as many filters as desired. The format for specifying a filter is a vector of characters including the column name, operator and value.

colname a string specifying the name of the column to be filtered

operator a string specifying what operator should be used in the filter (see options below)

value an integer or string specifying the value the columns should be filtered on

Possible operator values are as follows: "EQUALS", "EQUALS_ONE_OF", "NOT_EQUALS", "GREATER_THAN", "GREATER_THAN_OR_EQUAL_TO", "LESS_THAN", "LESS_THAN_OR_EQUAL_TO", "DATE_EQUAL", "DATE_NOT_EQUAL", "NOT_EQUAL_OR_NULL", "IS_MISSING", "IS_NOT_MISSING", "CONTAINS", "DOES_NOT_CONTAIN", "STARTS_WITH", and "DOES_NOT_START_WITH".

When using the "IS_MISSING" or "IS_NOT_MISSING" operators, an empty string should be supplied as the value. See example below.

Value

The function returns either a single string or an array of strings to be use in the `colFilter` argument of the `labkey.selectRows` function.

Author(s)

Valerie Obenchain

References

http://www.omegahat.org/RCurl/,
http://dssm.unipa.it/CRAN/web/packages/rjson/rjson.pdf,
https://www.labkey.org/project/home/begin.view

See Also

`labkey.selectRows`

Examples

```
## These example datasets are located at https://www.labkey.org/project/home/Study/demo/begi

## Two filters:
filter1<- makeFilter(c("HIVLoadQuant","GREATER_THAN",500), c("HIVRapidTest","EQUALS","Positi

## Using "equals one of" operator:
filter2 <- makeFilter(c("HIVLoadIneq","EQUALS_ONE_OF","Equals ; Less than"))

## Using "is not missing" operator:
filter3 <- makeFilter(c("HIVRapidTest","IS_NOT_MISSING",""))

## Apply a filter in labkey.selectRows function
getdata <- labkey.selectRows(
baseUrl="https://www.labkey.org",
folderPath="/home/Study/demo",
schemaName="study",
queryName="HIV Test Results",
colSelect=c("ParticipantId","HIVDate","HIVLoadQuant","HIVRapidTest"),
colFilter=filter3)
```