# Package 'OhdsiSharing'

July 27, 2021

**Type** Package

**Title** Package for sharing the results of the OHDSI tools

**Version** 0.2.2

**Date** 2020-06-18

**Maintainer** Lee Evans <evans@ohdsi.org>

**Description** Package for sharing the results of the OHDSI tools, with functions for encrypting results, sending results through SFTP to a central site, and uploading results into a postgresql database.

**URL** https://ohdsi.github.io/OhdsiSharing, https://github.com/OHDSI/OhdsiSharing

**BugReports** https://github.com/OHDSI/OhdsiSharing/issues

**Depends** R (>= 4.0.0)

**Imports** rJava,
ParallelLogger,
RJSONIO,
DatabaseConnector (>= 4.0.0),
SqlRender (>= 1.7.0),
dplyr (>= 1.0.0),
magrittr (>= 2.0.1),
readr (>= 1.4.0),
zip (>= 2.2.0),
rlang

**Suggests** testthat,
covr

**License** Apache License

**RoxygenNote** 7.1.1

**Encoding** UTF-8

## R topics documented:

---

`compressAndEncryptFolder` *Compress and encrypt a folder*

---

## Description

Compress and encrypt a folder

## Usage

```
compressAndEncryptFolder(sourceFolder, targetFileName, publicKeyFileName)
```

## Arguments

| | |
|---|---|
| `sourceFolder` | Name of the folder that must be encrypted. |
| `targetFileName` | Name of the file that will hold the encrypted data. |
| `publicKeyFileName` | Name of the file where the public key is stored. |

## Details

Compresses all files in a folder and its subfolders, and encrypts using the provided public key.

## Examples

```
## Not run:
generateKeyPair("public.key", "private.key")

# Create a folder with some data
dir.create("test")
data <- data.frame(x = runif(1000), y = 1:1000)
saveRDS(data, "test/data1.rds")
saveRDS(data, "test/data2.rds")

compressAndEncryptFolder("test", "data.zip.enc", "public.key")
decryptAndDecompressFolder("data.zip.enc", "test2", "private.key")

## End(Not run)
```

---

`compressFolder` *Compress a folder*

---

## Description

Compress a folder

## Usage

```
compressFolder(sourceFolder, targetFileName)
```

## Arguments

| | |
|---|---|
| `sourceFolder` | Name of the folder that must be compressed. |
| `targetFileName` | Name of the file that will hold the compressed data. |

## Details

Compresses all files in a folder and its subfolders, and stores it in a single zip file.

## Examples

```
## Not run:
# Create a folder with some data
dir.create("test")
data <- data.frame(x = runif(1000), y = 1:1000)
saveRDS(data, "test/data1.rds")
saveRDS(data, "test/data2.rds")

compressFolder("test", "data.zip")
decompressFolder("data.zip", "test2")

## End(Not run)
```

`createResultsDataModel` *Create the results data model tables on a database server.*

## Description

Create the results data model tables on a database server.

## Usage

```
createResultsDataModel(
  connection = NULL,
  connectionDetails = NULL,
  schema,
  sql
)
```

## Arguments

| | |
|---|---|
| `connection` | An object of type `connection` as created using the `connect` function in the DatabaseConnector package. Can be left NULL if `connectionDetails` is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes. |
| `connectionDetails` | An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided. |
| `schema` | The schema on the postgres server where the tables will be created. |
| `sql` | The postgres sql with the results data model DDL. |

## Details

Only PostgreSQL servers are supported.

---

`decompressFolder` *Decompress a folder*

## Description

Decompress a folder

## Usage

```
decompressFolder(sourceFileName, targetFolder)
```

## Arguments

| | |
|---|---|
| `sourceFileName` | Name of the file that must be decompressed. |
| `targetFolder` | Name of the folder that will hold the extracted data. |

### Details

Extracts all compressed files to a folder.

### Examples

```
## Not run:
# Create a folder with some data
dir.create("test")
data <- data.frame(x = runif(1000), y = 1:1000)
saveRDS(data, "test/data1.rds")
saveRDS(data, "test/data2.rds")

compressFolder("test", "data.zip")
decompressFolder("data.zip", "test2")

## End(Not run)
```

---

## decryptAndDecompressFolder

*Decrypt and decompress a folder*

---

### Description

Decrypt and decompress a folder

### Usage

```
decryptAndDecompressFolder(sourceFileName, targetFolder, privateKeyFileName)
```

### Arguments

| | |
|---|---|
| sourceFileName | Name of the file that must be decrypted. |
| targetFolder | Name of the folder that will hold the unencrypted data. |
| privateKeyFileName | Name of the file where the private key is stored. |

### Details

Decrypts the data using the provided private key and extracts all files to a folder.

### Examples

```
## Not run:
generateKeyPair("public.key", "private.key")

# Create a folder with some data
dir.create("test")
data <- data.frame(x = runif(1000), y = 1:1000)
saveRDS(data, "test/data1.rds")
saveRDS(data, "test/data2.rds")
```

```
compressAndEncryptFolder("test", "data.zip.enc", "public.key")
decryptAndDecompressFolder("data.zip.enc", "test2", "private.key")

## End(Not run)
```

---

`decryptFile` *Decrypt a data file*

---

## Description

Decrypt a data file

## Usage

```
decryptFile(sourceFileName, targetFileName, privateKeyFileName)
```

## Arguments

| | |
|---|---|
| `sourceFileName` | Name of the file that must be decrypted. |
| `targetFileName` | Name of the file that will hold the unencrypted data. |
| `privateKeyFileName` | Name of the file where the private key is stored. |

## Details

Decrypts the data using the provided private key.

## Examples

```
## Not run:
generateKeyPair("public.key", "private.key")
data <- data.frame(x = runif(1000), y = 1:1000)
saveRDS(data, "data.rds")
encryptFile("data.rds", "data.rds.enc", "public.key")
decryptFile("data.rds.enc", "data2.rds", "private.key")

## End(Not run)
```

---

`deleteAllRecordsForDatabaseId`
*Delete all records for database id*

---

## Description

Delete all records for database id

## Usage

```
deleteAllRecordsForDatabaseId(connection, schema, tableName, databaseId)
```

### Arguments

| | |
|---|---|
| connection | An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes. |
| schema | The schema on the postgres server where the tables will be created. |
| tableName | TODO: write description for parameter |
| databaseId | TODO: write description for parameter |

### Details

Only PostgreSQL servers are supported.

---

## deleteFromServer *Delete from server*

---

### Description

Delete from server

### Usage

```
deleteFromServer(connection, schema, tableName, keyValues)
```

### Arguments

| | |
|---|---|
| connection | An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes. |
| schema | The schema on the postgres server where the tables will be created. |
| tableName | TODO: write description for parameter |
| keyValues | TODO: write description for parameter |

### Details

Only PostgreSQL servers are supported.

## encryptFile *Encrypt a data file*

### Description

Encrypt a data file

### Usage

```
encryptFile(sourceFileName, targetFileName, publicKeyFileName)
```

### Arguments

| | |
|---|---|
| sourceFileName | Name of the file that must be encrypted. |
| targetFileName | Name of the file that will hold the encrypted data. |
| publicKeyFileName | Name of the file where the public key is stored. |

### Details

Encrypts the data using the provided public key.

### Examples

```
## Not run:
generateKeyPair("public.key", "private.key")
data <- data.frame(x = runif(1000), y = 1:1000)
saveRDS(data, "data.rds")
encryptFile("data.rds", "data.rds.enc", "public.key")

## End(Not run)
```

## generateKeyPair *Create a public-private key pair*

### Description

Create a public-private key pair

### Usage

```
generateKeyPair(publicKeyFileName, privateKeyFileName)
```

### Arguments

| | |
|---|---|
| publicKeyFileName | Name of the file where the public key should be stored. |
| privateKeyFileName | Name of the file where the private key should be stored. |

### Details

Creates an RSA 4096-bit public-private key pair. The public key can be used to encrypt data, and only with the private key can the data be decrypted.

### Examples

```
## Not run:
generateKeyPair("public.key", "private.key")

## End(Not run)
```

---

## preMergeResultsFiles *Pre-merge results files*

---

### Description

This function combines diagnostics results from one or more databases into a single file. The result is a single file that can be used as input for the Diagnostics Explorer Shiny app.

It also checks whether the results conform to the results data model specifications.

### Usage

```
preMergeResultsFiles(dataFolder, tempFolder = tempdir(), specifications)
```

### Arguments

| | |
|---|---|
| dataFolder | folder where the exported results zip files are stored. Zip files containing results from multiple databases may be placed in the same folder. |
| tempFolder | A folder on the local file system where the zip files are extracted to. Will be cleaned up when the function is finished. Can be used to specify a temp folder on a drive that has sufficient space if the default system temp space is too limited. |
| specifications | TODO: write description of parameter |

---

## sftpCd *Change the current working director*

---

### Description

Change the current working director

### Usage

```
sftpCd(sftpConnection, remoteFolder)
```

### Arguments

| | |
|---|---|
| sftpConnection | An SftpConnection object as created by the sftpConnect function. |
| remoteFolder | The folder on the server to change to. |

## sftpConnect *Connect to the OHDSI SFTP server*

### Description

Connect to the OHDSI SFTP server

### Usage

```
sftpConnect(privateKeyFileName, userName)
```

### Arguments

| | |
|---|---|
| privateKeyFileName | A character string denoting the path to an RSA private key. |
| userName | A character string containing the user name. |

### Value

An SftpConnection object

## sftpDisconnect *Disconnect from the OHDSI SFTP server.*

### Description

Disconnect from the OHDSI SFTP server.

### Usage

```
sftpDisconnect(sftpConnection)
```

### Arguments

| | |
|---|---|
| sftpConnection | An SftpConnection object as created by the sftpConnect function. |

## sftpGetFiles *Get one or more files from the SFTP server*

### Description

Get one or more files from the SFTP server

### Usage

```
sftpGetFiles(
  sftpConnection,
  remoteFileNames,
  localFolder = getwd(),
  localFileNames = file.path(localFolder, remoteFileNames)
)
```

### Arguments

| | |
|---|---|
| sftpConnection | An SftpConnection object as created by the sftpConnect function. |
| remoteFileNames | The name of the file(s) to get from the server. |
| localFolder | The path of a local folder where all files will be stored. Is ignored if localFileNames is provided. |
| localFileNames | The name the file(s) should have locally. If not provided, the files will be given the same names as on the server. |

## sftpLs *List the files in folder on the server.*

### Description

List the files in folder on the server.

### Usage

```
sftpLs(sftpConnection, remoteFolder = "./")
```

### Arguments

| | |
|---|---|
| sftpConnection | An SftpConnection object as created by the sftpConnect function. |
| remoteFolder | The folder on the server. Defaults to the current folder. |

### Value

A data frame with two columns: the file names, and the file types (directory, link, or file).

## sftpMkdir *Make a directory*

### Description

Make a directory

### Usage

```
sftpMkdir(sftpConnection, remoteFolder)
```

### Arguments

| | |
|---|---|
| sftpConnection | An SftpConnection object as created by the sftpConnect function. |
| remoteFolder | The folder on the server to create. |

## sftpPutFile *Put a file on the SFTP server*

### Description

Put a file on the SFTP server

### Usage

```
sftpPutFile(
  sftpConnection,
  localFileName,
  remoteFileName = basename(localFileName)
)
```

### Arguments

| | |
|---|---|
| sftpConnection | An SftpConnection object as created by the sftpConnect function. |
| localFileName | The path to the local file to upload. |
| remoteFileName | The name the file should have on the server. |

## sftpRename *Rename a file or folder*

### Description

Rename a file or folder

### Usage

```
sftpRename(sftpConnection, oldRemoteFilename, newRemoteFilename)
```

### Arguments

| | |
|---|---|
| sftpConnection | An SftpConnection object as created by the sftpConnect function. |
| oldRemoteFilename | The file on the server to rename. |
| newRemoteFilename | The new file name. |

## sftpRm *Remove one or more files*

### Description

Remove one or more files

### Usage

```
sftpRm(sftpConnection, remoteFiles)
```

### Arguments

| | |
|---|---|
| sftpConnection | An SftpConnection object as created by the sftpConnect function. |
| remoteFiles | The file(s) on the server to remove. |

## sftpRmdir *Remove a directory*

### Description

Remove a directory

### Usage

```
sftpRmdir(sftpConnection, remoteFolder)
```

### Arguments

| | |
|---|---|
| sftpConnection | An SftpConnection object as created by the sftpConnect function. |
| remoteFolder | The folder on the server to remove. |

## sftpUploadFile *Upload a single file to the OHDSI SFTP server*

### Description

This function combines calls to the sftpConnect, sftpPutFile, and sftpDisconnect functions. A random string will be prefixed to the file name to prevent overwriting existing files on the server.

### Usage

```
sftpUploadFile(privateKeyFileName, userName, remoteFolder = ".", fileName)
```

### Arguments

| | |
|---|---|
| privateKeyFileName | A character string denoting the path to an RSA private key. |
| userName | A character string containing the user name. |
| remoteFolder | The remote folder to upload the file to. |
| fileName | A character string denoting the path to file to upload. |

## sftPwd *Get the present working directory*

### Description

Get the present working directory

### Usage

```
sftPwd(sftpConnection)
```

### Arguments

| | |
|---|---|
| sftpConnection | An SftpConnection object as created by the sftpConnect function. |

### Value

A character string representing the current remote folder name.

uploadResults *Upload results to the database server.*

## Description

Requires the results data model tables have been created using the `createResultsDataModel` function.

Set the POSTGRES_PATH environmental variable to the path to the folder containing the psql executable to enable bulk upload (recommended).

## Usage

```
uploadResults(
  connectionDetails = NULL,
  schema,
  zipFileName,
  forceOverWriteOfSpecifications = FALSE,
  purgeSiteDataBeforeUploading = TRUE,
  tempFolder = tempdir(),
  specifications
)
```

## Arguments

| | |
|---|---|
| `connectionDetails` | An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. |
| `schema` | The schema on the postgres server where the tables have been created. |
| `zipFileName` | The name of the zip file. |
| `forceOverWriteOfSpecifications` | If TRUE, specifications of the phenotypes, cohort definitions, and analysis will be overwritten if they already exist on the database. Only use this if these specifications have changed since the last upload. |
| `purgeSiteDataBeforeUploading` | If TRUE, before inserting data for a specific databaseId all the data for that site will be dropped. This assumes the input zip file contains the full data for that data site. |
| `tempFolder` | A folder on the local file system where the zip files are extracted to. Will be cleaned up when the function is finished. Can be used to specify a temp folder on a drive that has sufficient space if the default system temp space is too limited. |
| `specifications` | A tibble data frame object with specifications. |

`validateResultsDataModelSpecifications` *Validate format of results data model specifications*

## Description

Validate format of results data model specifications

## Usage

```
validateResultsDataModelSpecifications(specifications)
```

## Arguments

| | |
|---|---|
| `specifications` | A tibble data frame object with specifications |

## Value

TRUE if valid format else FALSE

# Index