Package 'archivist'

September 21, 2014

Version 1.1

Type Package

Title Tools for storing, restoring and searching for R objects

Description Data exploration and modelling is a process in which a lot of data artifacts are produced. Artifacts like: subsets, data aggregates, plots,statistical models, different versions of data sets and different versions of results. The more projects we work with the more artifacts are produced and the harder it is to manage these artifacts. Archivist helps to store and manage artifacts created in R. Archivist allows you to store selected artifacts as a binary files together with their metadata and relations. Archivist allows to share artifacts with others, either through shared folder or github. Archivist allows to look for already created artifacts by using it's class, name, date of the creation or other properties. Makes it easy to restore such artifacts. Archivist allows to check if new artifact is the exact copy that was produced some time ago. That might be useful either for testing or caching.

Repository CRAN

License GPL-2

LazyLoad yes

LazyData yes

Depends R (>= 3.0.0), RSQLite, DBI, lubridate, jsonlite

Imports RCurl, digest, httr

BugReports `https://github.com/pbiecek/archivist/issues`

Author Przemyslaw Biecek [aut, cre],Marcin Kosinski [aut]

Maintainer Przemyslaw Biecek <przemyslaw.biecek@gmail.com>

NeedsCompilation no

Date/Publication 2014-09-21 07:43:45

R topics documented:

`archivist-package` *Tools for storing, restoring and searching for R objects*

Description

Data exploration and modelling is a process in which a lot of data artifacts are produced. Artifacts like: subsets, data aggregates, plots, statistical models, different versions of data sets and different versions of results. The more projects we work with the more artifacts are produced and the harder it is to manage these artifacts.

Archivist helps to store and manage artifacts created in R.

Archivist allows you to store selected artifacts as binary files together with their metadata and relations. Archivist allows you to share artifacts with others, either through a shared folder or github. Archivist allows you to look for already created artifacts by using its class, name, date of creation or other properties. It also facilitates restoring such artifacts. Archivist allows to check if a new artifact is the exact copy of the one that was produced some time ago. This might be useful either for testing or caching.

The list of main use cases is available here https://github.com/pbiecek/archivist[1].

Details

For more detailed information visit **archivist** `wiki` on Github[2].

Author(s)

Przemyslaw Biecek [aut, cre] <przemyslaw.biecek@gmail.com>
Marcin Kosinski [aut] <m.p.kosinski@gmail.com>

[1] https://github.com/pbiecek/archivist
[2] https://github.com/pbiecek/archivist/wiki

See Also

Other archivist: Repository; Tags; copyGithubRepo, copyLocalRepo; createEmptyRepo; deleteRepo; getTagsGithub, getTagsLocal; loadFromGithubRepo, loadFromLocalRepo; md5hash; rmFromRepo; saveToRepo; searchInGithubRepo, searchInLocalRepo; showGithubRepo, showLocalRepo; summaryGithubRepo, summaryLocalRepo; tarGithubRepo, tarLocalRepo

---

copyLocalRepo *Copy an Existing Repository to Another Repository*

Description

copyRepo copies artifact from one Repository to another Repository. It adds new files to exising gallery folder in repoTo Repository. copyLocalRepo copies local Repository, where copyGithubRepo copies Github Repository.

Usage

```
copyLocalRepo(repoFrom, repoTo, md5hashes)

copyGithubRepo(repoTo, md5hashes, user, repo, branch = "master",
  repoDirGit = FALSE)
```

Arguments

repoFrom	A character that specifies the directory of the Repository from which artifacts will be copied. Works only on copyLocalRepo.
repoTo	A character that specifies the directory of the Repository in which artifacts will be copied.
md5hashes	A character or character vector containing md5hashes of artifacts to be copied.
repo	Only if coping a Github repository. A character containing a name of a Github repository on which the repoFrom-Repository is archived.
user	Only if coping a Github repository. A character containing a name of a Github user on whose account the repoFrom is created.
branch	Only if coping with a Github repository. A character containing a name of Github Repository's branch on which a repoFrom-Repository is archived. Default branch is master.
repoDirGit	Only if working with a Github repository. A character containing a name of a directory on Github repository on which the Repository is stored. If the Repository is stored in main folder on Github repository, this should be set to repoDirGit = FALSE as default.

Details

copyRepo copies artifact from one Repository to another Repository. Functions copyLocalRepo and copyGithubRepo copy artifacts from the archivist Repositories stored in a local folder or on a Github. Both of them take md5hash as a parameter, which is a result from saveToRepo function. For every artifacts, md5hash is a unique string of length 32 that comes out as a result of digest function, which uses a cryptographical MD5 hash algorithm. For more information see md5hash.

Author(s)

Marcin Kosinski, <m.p.kosinski@gmail.com>

See Also

Other archivist: Repository; Tags; archivist-package; createEmptyRepo; deleteRepo; getTagsGithub, getTagsLocal; loadFromGithubRepo, loadFromLocalRepo; md5hash; rmFromRepo; saveToRepo; searchInGithubRepo, searchInLocalRepo; showGithubRepo, showLocalRepo; summaryGithubRepo, summaryLocalRepo; tarGithubRepo, tarLocalRepo

Examples

```
## Not run:


# creating example Repository - that examples will work

exampleRepoDir <- tempdir()
hashes <- searchInGithubRepo( pattern="name", user="pbiecek", repo="archivist", fixed=FALSE

createEmptyRepo( exampleRepoDir )

copyGithubRepo( repoTo = exampleRepoDir , md5hashes= hashes, user="pbiecek", repo="archivist

# removing an example Repository

deleteRepo( exampleRepoDir )

rm( exampleRepoDir )

# many archivist-like Repositories on one Github repository

dir <- paste0(getwd(), "/ex1")
createEmptyRepo( dir )
copyGithubRepo(repoTo = dir , md5hashes = "ff575c261c949d073b2895b05d1097c3",
              user="MarcinKosinski", repo="Museum",
              branch="master", repoDirGit="ex2")
deleteRepo( dir )


## End(Not run)
```

createEmptyRepo *Create an Empty Repository in a Given Directory*

Description

createEmptyRepo creates an empty Repository in a given directory in which archived artifacts will be stored.

Usage

```
createEmptyRepo(repoDir, force = FALSE)
```

Arguments

repoDir	A character that specifies the directory for the Repository to be made.
force	If force = TRUE function call forces to create repoDir directory if it did not exist. Default set to force = FALSE.

Details

At least one Repository must be initialized before using other functions from the **archivist** package. When working in groups, it is highly recommended to create a Repository on a shared Dropbox/Git folder.

All artifacts desired to be archived are going to be saved in the local Repository, which is an SQLite database stored in a file named backpack. After calling saveToRepo function, every artifact will be archived in a md5hash.rda file. This file will be saved in a folder (under repoDir directory) named gallery. For every artifact, md5hash is a unique string of length 32 that comes out as a result of digest function, which uses a cryptographical MD5 hash algorithm.

To learn more about artifacts visit archivist-package.

Created backpack database is a useful and fundamental tool for remembering artifact's name, class, archiving date etc. (that are remembered as Tags), or for keeping artifact's md5hash.

Besides the backpack database, gallery folder is created in which all artifacts will be archived.

After every saveToRepo call the database is refreshed, so an artifact is available immediately in backpack.db database for other collaborators.

Author(s)

Marcin Kosinski, <m.p.kosinski@gmail.com>

See Also

Other archivist: Repository; Tags; archivist-package; copyGithubRepo, copyLocalRepo; deleteRepo; getTagsGithub, getTagsLocal; loadFromGithubRepo, loadFromLocalRepo; md5hash; rmFromRepo; saveToRepo; searchInGithubRepo, searchInLocalRepo; showGithubRepo, showLocalRepo; summaryGithubRepo, summaryLocalRepo; tarGithubRepo, tarLocalRepo

Examples

```
## Not run:
exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )

# check the state of an empty Repository

summaryLocalRepo(  repoDir = exampleRepoDir )
showLocalRepo( exampleRepoDir )

# creating a Repository in non existing directory

createEmptyRepo( "xyzdd234", force = TRUE )

# removing an example Repositories

deleteRepo( exampleRepoDir )
deleteRepo( "xyzdd234" )

rm( exampleRepoDir )

## End(Not run)
```

`deleteRepo` *Delete an Existing Repository from Given Directory*

Description

`deleteRepo` deletes an existing Repository from a given directory, so all artifacts from `gallery` folder are removed and database `backpack.db` is deleted.

Usage

```
deleteRepo(repoDir)
```

Arguments

`repoDir`	A character that specifies the directory for the Repository to be deleted.

Details

`deleteRepo` deletes an existing Repository from a given directory, so all artifacts from `gallery` folder are removed and database `backpack.db` is deleted.

Author(s)

Marcin Kosinski, <m.p.kosinski@gmail.com>

See Also

Other archivist: `Repository`; `Tags`; `archivist-package`; `copyGithubRepo`, `copyLocalRepo`; `createEmptyRepo`; `getTagsGithub`, `getTagsLocal`; `loadFromGithubRepo`, `loadFromLocalRepo`; `md5hash`; `rmFromRepo`; `saveToRepo`; `searchInGithubRepo`, `searchInLocalRepo`; `showGithubRepo`, `showLocalRepo`; `summaryGithubRepo`, `summaryLocalRepo`; `tarGithubRepo`, `tarLocalRepo`

Examples

```
# objects preparation
## Not run:
# data.frame object
data(iris)

# ggplot/gg object
library(ggplot2)
df <- data.frame(gp = factor(rep(letters[1:3], each = 10)),y = rnorm(30))
library(plyr)
ds <- ddply(df, .(gp), summarise, mean = mean(y), sd = sd(y))
myplot123 <- ggplot(df, aes(x = gp, y = y)) +
  geom_point() +  geom_point(data = ds, aes(y = mean),
                colour = 'red', size = 3)

# lm object
model <- lm(Sepal.Length~ Sepal.Width + Petal.Length + Petal.Width, data= iris)

# agnes (twins) object
library(cluster)
data(votes.repub)
agn1 <- agnes(votes.repub, metric = "manhattan", stand = TRUE)

# fanny (partition) object
x <- rbind(cbind(rnorm(10, 0, 0.5), rnorm(10, 0, 0.5)),
         cbind(rnorm(15, 5, 0.5), rnorm(15, 5, 0.5)),
          cbind(rnorm( 3,3.2,0.5), rnorm( 3,3.2,0.5)))
fannyx <- fanny(x, 2)

# lda object
library(MASS)

Iris <- data.frame(rbind(iris3[,,1], iris3[,,2], iris3[,,3]),
                   Sp = rep(c("s","c","v"), rep(50,3)))
train <- c(8,83,115,118,146,82,76,9,70,139,85,59,78,143,68,
           134,148,12,141,101,144,114,41,95,61,128,2,42,37,
           29,77,20,44,98,74,32,27,11,49,52,111,55,48,33,38,
           113,126,24,104,3,66,81,31,39,26,123,18,108,73,50,
           56,54,65,135,84,112,131,60,102,14,120,117,53,138,5)
lda1 <- lda(Sp ~ ., Iris, prior = c(1,1,1)/3, subset = train)

# qda object
tr <- c(7,38,47,43,20,37,44,22,46,49,50,19,4,32,12,29,27,34,2,1,17,13,3,35,36)
train <- rbind(iris3[tr,,1], iris3[tr,,2], iris3[tr,,3])
```

```
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
qda1 <- qda(train, cl)

# glmnet object
library( glmnet )

zk=matrix(rnorm(100*20),100,20)
bk=rnorm(100)
glmnet1=glmnet(zk,bk)


# creating example Repository - that examples will work

# save examples

exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )
saveToRepo( myplot123, repoDir=exampleRepoDir )
saveToRepo( iris, repoDir=exampleRepoDir )
saveToRepo( model, repoDir=exampleRepoDir )
saveToRepo( agn1, repoDir=exampleRepoDir )
saveToRepo( fannyx, repoDir=exampleRepoDir )
saveToRepo( lda1, repoDir=exampleRepoDir )
saveToRepo( qda1, repoDir=exampleRepoDir )
saveToRepo( glmnet1, repoDir=exampleRepoDir )


# let's see how the Repository look like: show

showLocalRepo( method = "md5hashes", repoDir = exampleRepoDir )
showLocalRepo( method = "tags", repoDir = exampleRepoDir )

# let's get information about that Repository

summaryLocalRepo( repodir = exampleRepoDir )

# now let's delete the Repository

delete( repoDir = exampleRepoDir )

rm( exampleRepoDir )

## End(Not run)
```

`getTagsLocal` *Return a Tag Corresponding to md5hash*

Description

`getTagsLocal` and `getTagsGithub` return a `Tag` (see Tags) related to md5hash of an artifact. To learn more about artifacts visit archivist-package.

Usage

```
getTagsLocal(md5hash, repoDir, tag = "name")

getTagsGithub(md5hash, user, repo, branch = "master", repoDirGit = FALSE,
  tag = "name")
```

Arguments

repoDir	A character denoting an existing directory in which an artifact is stored.
md5hash	A character containing md5hash of artifacts which Tag is desired to be returned.
tag	A type of a Tags. Default tag = "name".
repo	Only if working with a Github repository. A character containing a name of a Github repository on which the Repository is archived.
user	Only if working with a Github repository. A character containing a name of a Github user on whose account the repo is created.
branch	Only if working with a Github repository. A character containing a name of Github repository's branch in which Repository is archived. Default branch is master.
repoDirGit	Only if working with a Github repository. A character containing a name of a directory on Github repository on which the Repository is stored. If the Repository is stored in main folder on Github repository, this should be set to repoDirGit = FALSE as default.

Details

getTagsLocal and getTagsGithub return (see Tags) related to md5hash of an artifact. To learn more about artifacts visit archivist-package.

Value

The character is returned, which is a Tag (see Tags) related to md5hash of an artifact.

Author(s)

Marcin Kosinski, <m.p.kosinski@gmail.com>

See Also

Other archivist: Repository; Tags; archivist-package; copyGithubRepo, copyLocalRepo; createEmptyRepo; deleteRepo; loadFromGithubRepo, loadFromLocalRepo; md5hash; rmFromRepo; saveToRepo; searchInGithubRepo, searchInLocalRepo; showGithubRepo, showLocalRepo; summaryGithubRepo, summaryLocalRepo; tarGithubRepo, tarLocalRepo

Other archivist: Repository; Tags; archivist-package; copyGithubRepo, copyLocalRepo; createEmptyRepo; deleteRepo; loadFromGithubRepo, loadFromLocalRepo; md5hash; rmFromRepo; saveToRepo; searchInGithubRepo, searchInLocalRepo; showGithubRepo, showLocalRepo; summaryGithubRepo, summaryLocalRepo; tarGithubRepo, tarLocalRepo

Examples

```
## Not run:
library(dplyr)
exampleRepoDir <- tempdir()
createEmptyRepo( exampleRepoDir )

data(mtcars)
hash <- mtcars %.%
 group_by(cyl, am) %.%
 select(mpg, cyl, wt, am) %.%
 summarise(avgmpg = mean(mpg), avgwt = mean(wt)) %.%
 filter(avgmpg > 20) %>%
 saveToRepo( exampleRepoDir )

getTagsLocal( md5hash = hash, exampleRepoDir )

deleteRepo( exampleRepoDir )
rm( exampleRepoDir )

getTagsGithub( "3db63bc63b8defaf42c0bde19160f242",
   user="pbiecek", repo="archivist")

# many archivist-like Repositories on one Github repository

getTagsGithub("ff575c261c949d073b2895b05d1097c3", user="MarcinKosinski",
repo="Museum", branch="master", repoDirGit="ex1")


## End(Not run)
```

---

loadFromLocalRepo *Load Artifact Given as a md5hash from a Repository*

Description

`loadFromLocalRepo` loads an artifact from a local Repository into the workspace. `loadFromGithubRepo` loads an artifact from a Github Repository into the workspace. To learn more about artifacts visit archivist-package.

Usage

```
loadFromLocalRepo(md5hash, repoDir, value = FALSE)

loadFromGithubRepo(md5hash, repo, user, branch = "master",
  repoDirGit = FALSE, value = FALSE)
```

Arguments

repoDir	A character denoting an existing directory from which an artifact will be loaded.
md5hash	A character assigned to the artifact as a result of a cryptographical hash function with MD5 algorithm, or it's abbreviation.
repo	Only if working with a Github repository. A character containing a name of a Github repository on which the Repository is archived.
user	Only if working with a Github repository. A character containing a name of a Github user on whose account the repo is created.
branch	Only if working with a Github repository. A character containing a name of Github Repository's branch on which the Repository is archived. Default branch is master.
repoDirGit	Only if working with a Github repository. A character containing a name of a directory on Github repository on which the Repository is stored. If the Repository is stored in main folder on Github repository, this should be set to repoDirGit = FALSE as default.
value	If FALSE (default) then artifacts are loaded into the Global Environment with their original names, if TRUE then artifacts are returned as a list of values (if there is more than one artifact) or as a single value (if there is only one arfifact that matches md5hash).

Details

Functions loadFromLocalRepo and loadFromGithubRepo load artifacts from the archivist Repositories stored in a local folder or on Github. Both of them take md5hash as a parameter, which is a result from saveToRepo function. For every artifact, md5hash is a unique string of length 32 that comes out as a result of digest function, which uses a cryptographical MD5 hash algorithm. For more information see md5hash.

Important: instead of giving the whole md5hash character, the user can simply give first few characters of the md5hash. For example, a09dd instead of a09ddjdkf9kj33dcjdnfjgos9jd9jkcv. All artifacts with the same corresponing md5hash abbreviation will be loaded from Repository.

Note that user and repo should be used only when working with a Github repository and should be omitted in the local mode. repoDir should only be used when working on a local Repository and should be omitted in the Github mode.

One may notice that loadFromGithubRepo and loadFromLocalRepo load artifacts to the Global Environment with their original names. Alternatively, a parameter value = TRUE might be specified so that the functions return artifacts as a result so that they can be attributed to new names. Note that, when an abbreviation of md5hash was given a list of artifacts corresponding to this abbreviation will be loaded.

Note

You can specify one md5hash (or its abbreviation) per function call.

Author(s)

Marcin Kosinski , <m.p.kosinski@gmail.com>

See Also

Other archivist: Repository; Tags; archivist-package; copyGithubRepo, copyLocalRepo; createEmptyRepo; deleteRepo; getTagsGithub, getTagsLocal; md5hash; rmFromRepo; saveToRepo; searchInGithubRepo, searchInLocalRepo; showGithubRepo, showLocalRepo; summaryGithubRepo, summaryLocalRepo; tarGithubRepo, tarLocalRepo

Examples

```
# objects preparation
## Not run:
# data.frame object
data(iris)

# ggplot/gg object
library(ggplot2)
df <- data.frame(gp = factor(rep(letters[1:3], each = 10)),y = rnorm(30))
library(plyr)
ds <- ddply(df, .(gp), summarise, mean = mean(y), sd = sd(y))
myplot123 <- ggplot(df, aes(x = gp, y = y)) +
  geom_point() +  geom_point(data = ds, aes(y = mean),
                             colour = 'red', size = 3)

# lm object
model <- lm(Sepal.Length~ Sepal.Width + Petal.Length + Petal.Width, data= iris)
model2 <- lm(Sepal.Length~ Sepal.Width + Petal.Width, data= iris)
model3 <- lm(Sepal.Length~ Sepal.Width, data= iris)

# agnes (twins) object
library(cluster)
data(votes.repub)
agn1 <- agnes(votes.repub, metric = "manhattan", stand = TRUE)

# fanny (partition) object
x <- rbind(cbind(rnorm(10, 0, 0.5), rnorm(10, 0, 0.5)),
           cbind(rnorm(15, 5, 0.5), rnorm(15, 5, 0.5)),
           cbind(rnorm( 3,3.2,0.5), rnorm( 3,3.2,0.5)))
fannyx <- fanny(x, 2)

# creating example Repository - that examples will work

exampleRepoDir <- tempdir()
createEmptyRepo(repoDir = exampleRepoDir)
myplo123Md5hash <- saveToRepo(myplot123, repoDir=exampleRepoDir)
irisMd5hash <- saveToRepo(iris, repoDir=exampleRepoDir)
modelMd5hash  <- saveToRepo(model, repoDir=exampleRepoDir)
agn1Md5hash <- saveToRepo(agn1, repoDir=exampleRepoDir)
fannyxMd5hash <- saveToRepo(fannyx, repoDir=exampleRepoDir)

# let's see how the Repository look like: show

showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)
showLocalRepo(method = "tags", repoDir = exampleRepoDir)
```

```
# let's see how the Repository look like: summary

summaryLocalRepo( exampleRepoDir )

# load examples

# first let's remove objects from Global Environment
rm(myplot123)
rm(iris)
rm(agn1)

# if those objects were archivised, they can be loaded
# from Repository, when knowing their tags

loadFromLocalRepo(myplo123Md5hash, repoDir = exampleRepoDir)
loadFromLocalRepo(irisMd5hash, repoDir = exampleRepoDir)


# if one can not remembers the object's md5hash but
# remembers the object's name this object can still be
# loaded.

agnesHash <- searchInLocalRepo( "name:agn1", repoDir = exampleRepoDir)
loadFromLocalRepo(agnesHash, repoDir = exampleRepoDir)

# one can check that object has his own unique md5hash
agnesHash == agn1Md5hash

# object can be loadad as a value

newData <- loadFromLocalRepo(irisMd5hash, repoDir = exampleRepoDir, value = TRUE)

# object can be also loaded from it's abbreviation
# modelMd5hash <- saveToRepo( model , repoDir=exampleRepoDir)
rm( model )
# "cd6557c6163a6f9800f308f343e75e72"
# so example abbreviation might be : "cd6557c"
loadFromLocalRepo("cd6557c", repoDir = exampleRepoDir)

# and can be loaded as a value from it's abbreviation
newModel  <- loadFromLocalRepo("cd6557c", repoDir = exampleRepoDir, value = TRUE)
# note that "model" was not deleted

#
#GitHub Version
#

# check the state of the Repository
summaryGithubRepo( user="pbiecek", repo="archivist" )
showGithubRepo( user="pbiecek", repo="archivist" )
showGithubRepo( user="pbiecek", repo="archivist", method = "tags" )
```

```
rm( model )
rm( myplot123 )
rm( qda1 )
(VARmd5hash <- searchInGithubRepo( "varname:Sepal.Width",
                    user="pbiecek", repo="archivist" ))
(NAMEmd5hash <- searchInGithubRepo( "name:qda1",
                    user="pbiecek", repo="archivist", branch="master" ))
(CLASSmd5hash <- searchInGithubRepo( "class:ggplot",
                    user="pbiecek", repo="archivist", branch="master" ))


loadFromGithubRepo( "ff575c261c", user="pbiecek", repo="archivist")
NewObjects <- loadFromGithubRepo( NAMEmd5hash, user="pbiecek", repo="archivist", value = TRU
loadFromGithubRepo( DATEmd5hash, user="pbiecek", repo="archivist")


# removing an example Repository

deleteRepo( exampleRepoDir )

rm( exampleRepoDir )

# many archivist-like Repositories on one Github repository

loadFromGithubRepo( "ff575c261c949d073b2895b05d1097c3",
user="MarcinKosinski", repo="Museum", branch="master", repoDirGit="ex2")


loadFromGithubRepo( "ff575c261c949d073b2895b05d1097c3",
                     user="MarcinKosinski", repo="Museum", branch="master",
                     repoDirGit="ex1")

## End(Not run)
```

`md5hash` *md5hash*

Description

`Repository` stores specific values of an artifact, different for various artifact's classes, and artifact themselves. Artifacts are archived with a special attribute named `md5hash`. To learn more about artifacts visit archivist-package.

Details

For every artifact, `md5hash` is a unique string of length 32 that comes out as a result of digest function, which uses a cryptographical MD5 hash algorithm. The `md5hash` of every artifact, that is archived to the Repository, is also saved to the Repository with this artifact's `Tags` - see Tags. It enables to distinguish artifacts in Repository and facilitates searching and loading them.

See Also

Functions that take `md5hash` as a parameter are:

- rmFromRepo,
- loadFromLocalRepo,
- loadFromGithubRepo,
- copyLocalRepo,
- copyGithubRepo.

Functions returning `md5hash` as a value are:

- searchInLocalRepo,
- searchInGithubRepo,
- saveToRepo.

Functions returning `md5hahes` as a `data.frame` are:

- showLocalRepo,
- showGithubRepo.

Learn more about `md5hashes` at **archivist** `wiki` webpage on Github[3].

Other archivist: `Repository`; `Tags`; `archivist-package`; `copyGithubRepo`, `copyLocalRepo`; `createEmptyRepo`; `deleteRepo`; `getTagsGithub`, `getTagsLocal`; `loadFromGithubRepo`, `loadFromLocalRepo`; `rmFromRepo`; `saveToRepo`; `searchInGithubRepo`, `searchInLocalRepo`; `showGithubRepo`, `showLocalRepo`; `summaryGithubRepo`, `summaryLocalRepo`; `tarGithubRepo`, `tarLocalRepo`

`Repository` *Repository*

Description

`Repository` stores specific values of an artifact, different for various artifact's classes, and artifacts themselves. To learn more about artifacts visit archivist-package.

Details

`Repository` is folder with an SQLite database stored in a file named `backpack` and a subdirectory named `gallery` with collection of artifact saved as `.rda` files.

[3] `https://github.com/pbiecek/archivist/wiki/archivist-package-md5hash`

See Also

Functions using `Repository` are:

- saveToRepo,
- rmFromRepo,
- loadFromLocalRepo,
- loadFromGithubRepo,
- searchInLocalRepo,
- searchInGithubRepo,
- summaryLocalRepo,
- summaryGithubRepo.

Function creating `Repository` is:

- createEmptyRepo.

Functions coping `Repository` are:

- copyLocalRepo,
- copyGithubRepo.

Function deleting `Repository` is:

- deleteRepo.

Learn more about `Repository` at **archivist** `wiki` webpage on Github[4].

Other archivist: `Tags`; `archivist-package`; `copyGithubRepo`, `copyLocalRepo`; `createEmptyRepo`; `deleteRepo`; `getTagsGithub`, `getTagsLocal`; `loadFromGithubRepo`, `loadFromLocalRepo`; `md5hash`; `rmFromRepo`; `saveToRepo`; `searchInGithubRepo`, `searchInLocalRepo`; `showGithubRepo`, `showLocalRepo`; `summaryGithubRepo`, `summaryLocalRepo`; `tarGithubRepo`, `tarLocalRepo`

`rmFromRepo` *Remove an Artifact Given as md5hash from a Repository*

Description

`rmFromRepo` removes an artifact given as `md5hash` from a Repository. To learn more about artifacts visit archivist-package.

Usage

```
rmFromRepo(md5hash, repoDir, removeData = FALSE, removeMiniature = FALSE,
  force = FALSE)
```

[4] `https://github.com/pbiecek/archivist/wiki/archivist-package-Repository`

Arguments

md5hash	A character assigned to the artifact as a result of a cryptographical hash function with MD5 algorithm, or it's abbreviation. This object will be removed.
repoDir	A character denoting an existing directory from which an artifact will be removed.
removeData	A logical value denoting whether to remove a data with the artifact specified by the md5hash. Defualt FALSE.
removeMiniature	A logical value denoting whether to remove a miniature with the artifact specified by the md5hash. Defualt FALSE.
force	A logical value denoting whether to remove data if it is related to more than 1 artifact. Defualt FALSE.

Details

rmFromRepo removes an artifact given as md5hash from a Repository, which is a SQLite database named backpack - created by a createEmptyRepo call. For every artifact, md5hash is a unique string of length 32 that comes out as a result of digest function, which uses a cryptographical MD5 hash algorithm.

Also this function removes a md5hash.rda file, where md5hash is the artifact's hash as above.

Important: instead of giving the whole md5hash character, the user can simply give first few characters of the md5hash. For example, a09dd instead of a09ddjdkf9kj33dcjdnfjgos9jd9jkcv. All artifacts with the same corresponing md5hash abbreviation will be removed from the Repository and from the gallery folder.

rmFromRepo provides functionality that enables to delete miniatures of the artifacts (.txt or .png files) while removing .rda files. To disable this functionality use removeMiniature = FALSE. Also, if the data from the artifact were archived, the data will be removed by default but there is a possibility not to delete this data while performing rmFromRepo - simply use removeData = TRUE.

rmFromRepo provides functionality that enables to delete miniatures of the artifacts (.txt or .png files) while removing .rda files. To delete miniature use removeMiniature = TRUE. Also if the data from the artifact was archived, there is a possibility to delete this data while removing artifact that uses this data. Simply use removeData = TRUE.

If one wants to remove all artifact created between two dates, it is suggested to perform:

- obj2rm <- searchInLocalRepo(tag = list(dateFrom,dateTo), repoDir =)
- sapply(obj2rm,rmFromRepo, repoDir =)

Note

md5hash can be a result of the searchInLocalRepo function proceeded with tag = NAME argument, where NAME is a tag that describes the property of the objects to be deleted.

For more information about Tags check Tags.

Author(s)

Marcin Kosinski , <m.p.kosinski@gmail.com>

See Also

Other archivist: Repository; Tags; archivist-package; copyGithubRepo, copyLocalRepo; createEmptyRepo; deleteRepo; getTagsGithub, getTagsLocal; loadFromGithubRepo, loadFromLocalRepo; md5hash; saveToRepo; searchInGithubRepo, searchInLocalRepo; showGithubRepo, showLocalRepo; summaryGithubRepo, summaryLocalRepo; tarGithubRepo, tarLocalRepo

Examples

```
# objects preparation
## Not run:
data.frame object
data(iris)

# ggplot/gg object
library(ggplot2)
df <- data.frame(gp = factor(rep(letters[1:3], each = 10)),y = rnorm(30))
library(plyr)
ds <- ddply(df, .(gp), summarise, mean = mean(y), sd = sd(y))
myplot123 <- ggplot(df, aes(x = gp, y = y)) +
  geom_point() +  geom_point(data = ds, aes(y = mean),
                             colour = 'red', size = 3)

# lm object
model <- lm(Sepal.Length~ Sepal.Width + Petal.Length + Petal.Width, data= iris)
model2 <- lm(Sepal.Length~ Sepal.Width + Petal.Width, data= iris)
model3 <- lm(Sepal.Length~ Sepal.Width, data= iris)

# agnes (twins) object
library(cluster)
data(votes.repub)
agn1 <- agnes(votes.repub, metric = "manhattan", stand = TRUE)

# fanny (partition) object
x <- rbind(cbind(rnorm(10, 0, 0.5), rnorm(10, 0, 0.5)),
           cbind(rnorm(15, 5, 0.5), rnorm(15, 5, 0.5)),
           cbind(rnorm( 3,3.2,0.5), rnorm( 3,3.2,0.5)))
fannyx <- fanny(x, 2)

# creating example Repository - that examples will work

exampleRepoDir <- tempdir()
createEmptyRepo(repoDir = exampleRepoDir)
myplo123Md5hash <- saveToRepo(myplot123, repoDir=exampleRepoDir)
irisMd5hash <- saveToRepo(iris, repoDir=exampleRepoDir)
modelMd5hash  <- saveToRepo(model, repoDir=exampleRepoDir)
agn1Md5hash <- saveToRepo(agn1, repoDir=exampleRepoDir)
fannyxMd5hash <- saveToRepo(fannyx, repoDir=exampleRepoDir)

# let's see how the Repository look like: show
showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)
showLocalRepo(method = "tags", repoDir = exampleRepoDir)
```

```
# let's see how the Repository look like: summary
summaryLocalRepo( exampleRepoDir )

# remove examples

rmFromRepo(fannyxMd5hash, repoDir = exampleRepoDir, removeData= FALSE)
# removeData = FALSE provides from removing archived "fannyxMd5hash object"-data from
# a Repository and gallery

rmFromRepo(irisMd5hash, repoDir = exampleRepoDir)
# note that also files in gallery folder, created in exampleRepoDir
# directory will be removed

# let's see how the Repository look like: show
showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)
showLocalRepo(method = "tags", repoDir = exampleRepoDir)

# let's see how the Repository look like: summary
summaryLocalRepo( exampleRepoDir )


# one can have the same object archived 3 different times
# there will appear a warning message
agn1Md5hash2 <- saveToRepo(agn1, repoDir=exampleRepoDir)
agn1Md5hash3 <- saveToRepo(agn1, repoDir=exampleRepoDir)

# md5hashes are the same for that same object (agn1)
agn1Md5hash == agn1Md5hash2
agn1Md5hash2 == agn1Md5hash3



# but there are 3 times more rows in Repository database (backpack.db).

# let's see how the Repository look like: show
showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)
showLocalRepo(method = "tags", repoDir = exampleRepoDir)

# let's see how the Repository look like: summary
summaryLocalRepo( exampleRepoDir )

# one easy call removes them all but this call will result in error
rmFromRepo(agn1Md5hash, repoDir = exampleRepoDir, removeData = TRUE,
            removeMiniature = TRUE)

# soultion to that is
rmFromRepo(agn1Md5hash, repoDir = exampleRepoDir, removeData = TRUE,
            removeMiniature = TRUE, force = TRUE)
# removeMiniature = TRUE removes miniatures from gallery folder

# rest of artifacts can be removed e.g. like this
# looking for dates of creation and then removing all objects
```

```
# from specific date

obj2rm <- searchInLocalRepo( pattern = list(dateFrom = Sys.Date(), dateTo = Sys.Date()),
                             repoDir = exampleRepoDir )
sapply(obj2rm, rmFromRepo, repoDir = exampleRepoDir)
# above example removed all objects from this example

# let's see how the Repository look like: show
showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)
showLocalRepo(method = "tags", repoDir = exampleRepoDir)

# one can also remove objects from only specific class
modelMd5hash  <- saveToRepo(model, repoDir=exampleRepoDir)
model2Md5hash  <- saveToRepo(model2, repoDir=exampleRepoDir)
model3Md5hash  <- saveToRepo(model3, repoDir=exampleRepoDir)
showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)

objMd5hash <- searchInLocalRepo("class:lm", repoDir = exampleRepoDir)
sapply(objMd5hash, rmFromRepo, repoDir = exampleRepoDir, removeData = TRUE, force = TRUE)
showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)
summaryLocalRepo( exampleRepoDir )


# once can remove object specifying only its md5hash abbreviation
(myplo123Md5hash <- saveToRepo(myplot123, repoDir=exampleRepoDir))
showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)
# "ff751bb5ba34bbb8a7851958b15f2ef7"
# so example abbreviation might be : "ff751"
rmFromRepo("ff751", repoDir = exampleRepoDir)
summaryLocalRepo( repoDir = exampleRepoDir )


# removing an example Repository

deleteRepo( exampleRepoDir )

rm( exampleRepoDir )

## End(Not run)
```

`saveToRepo` *Save an Artifact into a Repository*

Description

`saveToRepo` function saves desired artifacts to the local Repository in a given directory. To learn more about artifacts visit archivist-package.

Usage

```
saveToRepo(artifact, repoDir, archiveData = TRUE, archiveTags = TRUE,
  archiveMiniature = TRUE, force = TRUE, rememberName = TRUE,
  chain = FALSE, ...)
```

Arguments

artifact	An arbitrary R artifact to be saved. For supported artifacts see details.
...	Graphical parameters denoting width and height of a miniature. See details.
archiveData	A logical value denoting whether to archive the data from the artifact.
archiveTags	A logical value denoting whether to archive tags from the artifact.
archiveMiniature	A logical value denoting whether to archive a miniature of the artifact.
repoDir	A character denoting an existing directory in which an artifact will be saved.
force	A logical value denoting whether to archive artifact if it was already archived in a Repository.
rememberName	A logical value. Should not be changed by an user. It is a technical parameter.
chain	A logical value. Should the result be (default chain = FALSE) the md5hash of an stored artifact or should the result be the input artifact (chain = TRUE), so that chaining code can be used. See examples.

Details

saveToRepo function saves desired artifacts to the local Repository in a given directory. Artifacts are saved in the local Repository, which is a SQLite database named backpack. After every saveToRepo call the database is refreshed, so the artifact is available immediately in the database for other collaborators. Every artifact is archived in a md5hash.rda file. This file will be saved in a folder (under repoDir directory) named gallery. For every artifact, md5hash is a unique string of length 32 that comes out as a result of digest function, which uses a cryptographical MD5 hash algorithm.

By default, a miniature of an artifact and (if possible) a data set needed to compute this artifact are extracted. They are also going to be saved in a file named by their md5hash in the gallery folder that exists in the directory specified in the repoDir argument. Moreover, a specific Tag-relation is going to be added to the backpack dataset in case there is a need to load the artifact with it's related data set - see loadFromLocalRepo or loadFromGithubRepo. Default settings may be changed by using the archiveData, archiveTag or archiveMiniature arguments with the FALSE value.

Tags are artifact's attributes, different for various artifact's classes. For more detailed information check Tags

Archived artifact can be searched in the backpack dataset by using the searchInLocalRepo or searchInGithubRepo functions. Artifacts can be searched by their Tags, names, classes or archiving date.

Graphical parameters.

If the artifact is of class data.frame or archiveData = TRUE, it is possible to specify how many rows of that data should be archived by adding the argument firstRows with the n specified

number of rows. Note that, the date can be extracted only from the artifacts that are supported by the **archivist** package; see Tags.

If the artifact is of class `lattice` or `ggplot`, and `archiveMiniature = TRUE`, then it is possible to set the miniature's `width` and `height` parameters. By default they are set to `width = 800`, `height = 600`.

Supported artifact's classes are (so far):

- `lm`,
- `data.frame`,
- `ggplot`,
- `htest`,
- `trellis`,
- `twins (result of agnes, diana or mona function)`,
- `partition (result of pam, clara or fanny fuction)`,
- `lda`,
- `qda`,
- `glmnet`,
- `survfit`.

To check what `Tags` will be extracted for various artifacts see Tags.

Value

As a result of this function a character string is returned, which determines the `md5hash` of the artifact that was used in the `saveToRepo` function. If `archiveData` was `TRUE`, the result also has an attribute, named `data`, that determines `md5hash` of the data needed to compute the artifact.

Note

One can specify his own `Tags` for artifacts by setting artifact's attribute before call of the `saveToRepo` function like this: `attr(x, "tags" ) = c( "name1", "name2" )`, where `x` is artifact and `name1, name2` are `Tags` specified by an user.

Important: if one want to archive data from arftifacts that class is one of: `survfit, glmnet, qda, lda, trellis,` and this dataset set is transformed only in the artifact's formula the `saveToRepo` will not archive this dataset. `saveToRepo` only archives datasets that already exists in any of R environments.

Example: here data set will not be archived.

- `z <- lda(Sp ~ ., Iris, prior = c(1,1,1)/3,subset = train[,-8])`
- `saveToRepo( z, repoDir )`

Example: here data set will be archived.

- `train2 <- train[,-8]`
- `z <- lda(Sp ~ .,Iris, prior = c(1,1,1)/3, subset = train2)`
- `saveToRepo( z, repoDir )`

Author(s)

Marcin Kosinski , <m.p.kosinski@gmail.com>

See Also

For more detailed information check the **archivist** package Use Cases[5]. The list of supported artifacts and their tags is available on `wiki` on **archivist** Github Repository[6].

Other archivist: `Repository`; `Tags`; `archivist-package`; `copyGithubRepo`, `copyLocalRepo`; `createEmptyRepo`; `deleteRepo`; `getTagsGithub`, `getTagsLocal`; `loadFromGithubRepo`, `loadFromLocalRepo`; `md5hash`; `rmFromRepo`; `searchInGithubRepo`, `searchInLocalRepo`; `showGithubRepo`, `showLocalRepo`; `summaryGithubRepo`, `summaryLocalRepo`; `tarGithubRepo`, `tarLocalRepo`

Examples

```
# objects preparation
## Not run:
# data.frame object
data(iris)

# ggplot/gg object
library(ggplot2)
df <- data.frame(gp = factor(rep(letters[1:3], each = 10)),y = rnorm(30))
library(plyr)
ds <- ddply(df, .(gp), summarise, mean = mean(y), sd = sd(y))
myplot123 <- ggplot(df, aes(x = gp, y = y)) +
  geom_point() +  geom_point(data = ds, aes(y = mean),
                colour = 'red', size = 3)

# lm object
model <- lm(Sepal.Length~ Sepal.Width + Petal.Length + Petal.Width, data= iris)

# agnes (twins) object
library(cluster)
data(votes.repub)
agn1 <- agnes(votes.repub, metric = "manhattan", stand = TRUE)

# fanny (partition) object
x <- rbind(cbind(rnorm(10, 0, 0.5), rnorm(10, 0, 0.5)),
         cbind(rnorm(15, 5, 0.5), rnorm(15, 5, 0.5)),
          cbind(rnorm( 3,3.2,0.5), rnorm( 3,3.2,0.5)))
fannyx <- fanny(x, 2)

# lda object
library(MASS)

Iris <- data.frame(rbind(iris3[,,1], iris3[,,2], iris3[,,3]),
                   Sp = rep(c("s","c","v"), rep(50,3)))
train <- c(8,83,115,118,146,82,76,9,70,139,85,59,78,143,68,
```

[5] https://github.com/pbiecek/archivist#-the-list-of-use-cases-

[6] https://github.com/pbiecek/archivist/wiki/archivist-package---Tags

```
           134,148,12,141,101,144,114,41,95,61,128,2,42,37,
           29,77,20,44,98,74,32,27,11,49,52,111,55,48,33,38,
           113,126,24,104,3,66,81,31,39,26,123,18,108,73,50,
           56,54,65,135,84,112,131,60,102,14,120,117,53,138,5)
lda1 <- lda(Sp ~ ., Iris, prior = c(1,1,1)/3, subset = train)

# qda object
tr <- c(7,38,47,43,20,37,44,22,46,49,50,19,4,32,12,29,27,34,2,1,17,13,3,35,36)
train <- rbind(iris3[tr,,1], iris3[tr,,2], iris3[tr,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
qda1 <- qda(train, cl)

# glmnet object
library( glmnet )

zk=matrix(rnorm(100*20),100,20)
bk=rnorm(100)
glmnet1=glmnet(zk,bk)


# creating example Repository - that examples will work

# save examples

exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )
saveToRepo( myplot123, repoDir=exampleRepoDir )
saveToRepo( iris, repoDir=exampleRepoDir )
saveToRepo( model, repoDir=exampleRepoDir )
saveToRepo( agn1, repoDir=exampleRepoDir )
saveToRepo( fannyx, repoDir=exampleRepoDir )
saveToRepo( lda1, repoDir=exampleRepoDir )
saveToRepo( glmnet1, repoDir=exampleRepoDir )

# let's see how the Repository look like: show

showLocalRepo( method = "md5hashes", repoDir = exampleRepoDir )
showLocalRepo( method = "tags", repoDir = exampleRepoDir )

# let's see how the Repository look like: summary

summaryLocalRepo( exampleRepoDir )

# one can archived the same artifact twice, but there is a message

saveToRepo( model, repoDir=exampleRepoDir )

# in case not to archive the same artifact twice, use

saveToRepo( lda1, repoDir=exampleRepoDir, force = FALSE )

# one can archive artifact withouth it's database and miniature
```

```
saveToRepo( qda1, repoDir=exampleRepoDir, archiveData = FALSE,
            archiveMiniature = FALSE)

# one can specify his own additional tags to be archived with artifact

attr( model, "tags" ) = c( "do not delete", "my favourite model" )
saveToRepo( model, repoDir=exampleRepoDir )
showLocalRepo( "tags", exampleRepoDir )

# removing an example Repository

# saveToRepo in chaining code
library(dplyr)

data("hflights", package = "hflights")
hflights %>%
  group_by(Year, Month, DayofMonth) %>%
  select(Year:DayofMonth, ArrDelay, DepDelay) %>%
  saveToRepo( exampleRepoDir, chain = TRUE ) %>%
  # here the artifact is stored but chaining is not finished
  summarise(
    arr = mean(ArrDelay, na.rm = TRUE),
    dep = mean(DepDelay, na.rm = TRUE)
  ) %>%
  filter(arr > 30 | dep > 30) %>%
  saveToRepo( exampleRepoDir )
  # chaining code is finished and after last operation the
  # artifact is stored



deleteRepo( exampleRepoDir )

rm( exampleRepoDir )

## End(Not run)
```

`searchInLocalRepo` *Search for an Artifact in a Repository Using Tags*

Description

`searchInRepo` searches for an artifact in a Repository using it's Tags. To learn more about artifacts visit archivist-package.

Usage

```
searchInLocalRepo(pattern, repoDir, fixed = TRUE, realDBname = TRUE)

searchInGithubRepo(pattern, repo, user, branch = "master",
  repoDirGit = FALSE, fixed = TRUE)
```

Arguments

pattern	If fixed = TRUE: a character denoting a Tags to be searched for in the Repository. It is also possible to specify pattern as a list of length 2 with dataFrom and dataTo; see details. If fixed = FALSE: A regular expression specifying the beginning of a Tags, which will be used to search artifacts for.
repoDir	A character denoting an existing directory in which artifacts will be searched.
repo	Only if working with a Github repository. A character containing a name of a Github repository on which the Repository is archived.
user	Only if working with a Github repository. A character containing a name of a Github user on whose account the repo is created.
branch	Only if working with a Github repository. A character containing a name of Github repository's branch in which Repository is archived. Default branch is master.
fixed	A logical value specifying how artifacts should be searched for. If fixed = TRUE (default) then artifacts are searched exactly to the corresponding pattern parameter. If fixed = FALSE then artifacts are searched using pattern paremeter as a regular expression - that method is wider and more flexible and, i.e., enables to search for all artifacts in the Repository, using pattern = "name", fixed = F
repoDirGit	Only if working with a Github repository. A character containing a name of a directory on Github repository on which the Repository is stored. If the Repository is stored in main folder on Github repository, this should be set to repoDirGit = FALSE as default.
realDBname	A logical value. Should not be changed by user. It is a technical parameter.

Details

searchInRepo searches for an artifact in a Repository using it's Tag (e.g., name, class or archiving date). Tags are submitted in a pattern argument. For various artifact classes different Tags can be searched for. See Tags. If a pattern is a list of length 2, md5hashes of all artifacts created from date dateFrom to data dateTo are returned. The date should be formatted according to the YYYY-MM-DD format, e.g., "2014-07-31".

Tags, submitted in a pattern argument, should be given according to the format: "TagType:TagTypeValue" - see examples.

Value

searchInRepo returns a md5hash character, which is a hash assigned to the artifact when saving it to the Repository by using the saveToRepo function. If the artifact is not in the Repository a logical value FALSE is returned.

Author(s)

Marcin Kosinski, <m.p.kosinski@gmail.com>

See Also

Other archivist: Repository; Tags; archivist-package; copyGithubRepo, copyLocalRepo; createEmptyRepo; deleteRepo; getTagsGithub, getTagsLocal; loadFromGithubRepo, loadFromLocalRepo; md5hash; rmFromRepo; saveToRepo; showGithubRepo, showLocalRepo; summaryGithubRepo, summaryLocalRepo; tarGithubRepo, tarLocalRepo

Examples

```
# objects preparation
## Not run:
  # data.frame object
  data(iris)

 # ggplot/gg object
  library(ggplot2)
  df <- data.frame(gp = factor(rep(letters[1:3], each = 10)),y = rnorm(30))
  library(plyr)
  ds <- ddply(df, .(gp), summarise, mean = mean(y), sd = sd(y))
  myplot123 <- ggplot(df, aes(x = gp, y = y)) +
    geom_point() +  geom_point(data = ds, aes(y = mean),
                               colour = 'red', size = 3)

  # lm object
  model <- lm(Sepal.Length~ Sepal.Width + Petal.Length + Petal.Width, data= iris)

  # agnes (twins) object
  library(cluster)
  data(votes.repub)
  agn1 <- agnes(votes.repub, metric = "manhattan", stand = TRUE)

  # fanny (partition) object
  x <- rbind(cbind(rnorm(10, 0, 0.5), rnorm(10, 0, 0.5)),
             cbind(rnorm(15, 5, 0.5), rnorm(15, 5, 0.5)),
             cbind(rnorm( 3,3.2,0.5), rnorm( 3,3.2,0.5)))
  fannyx <- fanny(x, 2)

  # creating example Repository - that examples will work

  exampleRepoDir <- tempdir()
  createEmptyRepo(repoDir = exampleRepoDir)
  saveToRepo(myplot123, repoDir=exampleRepoDir)
  saveToRepo(iris, repoDir=exampleRepoDir)
  saveToRepo(model, repoDir=exampleRepoDir)
  saveToRepo(agn1, repoDir=exampleRepoDir)
  saveToRepo(fannyx, repoDir=exampleRepoDir)

  # let's see how the Repository look like: show

  showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)
  showLocalRepo(method = "tags", repoDir = exampleRepoDir)

  # let's see how the Repository look like: summary
```

```
summaryLocalRepo( exampleRepoDir )


# search examples

# tag search, fixed version
searchInLocalRepo( "class:ggplot", repoDir = exampleRepoDir )
searchInLocalRepo( "name:myplot123", repoDir = exampleRepoDir )
searchInLocalRepo( "varname:Sepal.Width", repoDir = exampleRepoDir )
searchInLocalRepo( "class:lm", repoDir = exampleRepoDir )
searchInLocalRepo( "coefname:Petal.Length", repoDir = exampleRepoDir )
searchInLocalRepo( "ac:0.797755535467609", repoDir = exampleRepoDir )

# tag search, regex version

searchInLocalRepo( "class", repoDir = exampleRepoDir, fixed = FALSE )
searchInLocalRepo( "name", repoDir = exampleRepoDir, fixed = FALSE )

# Github version
# check the state of the Repository
summaryGithubRepo( user="pbiecek", repo="archivist" )
showGithubRepo( user="pbiecek", repo="archivist" )
showGithubRepo( user="pbiecek", repo="archivist", method = "tags" )

# tag search, fixed version
searchInGithubRepo( "varname:Sepal.Width", user="pbiecek", repo="archivist" )
searchInGithubRepo( "class:lm", user="pbiecek", repo="archivist", branch="master" )
searchInGithubRepo( "name:myplot123", user="pbiecek", repo="archivist" )

# tag search, regex version
searchInGithubRepo( "class", user="pbiecek", repo="archivist",  fixed = FALSE )
searchInGithubRepo( "name", user="pbiecek", repo="archivist", fixed = FALSE )


# date search

# objects archivised between 2 different days
searchInLocalRepo( pattern = list( dateFrom = Sys.Date()-1, dateTo = Sys.Date()+1),
                   repoDir = exampleRepoDir )

# also on Github

searchInGithubRepo( pattern = list( dateFrom = "2014-09-01", dateTo = "2014-09-30" ),
                    user="pbiecek", repo="archivist", branch="master" )


# objects from Today
searchInLocalRepo( pattern = list( dateFrom=Sys.Date(), dateTo=Sys.Date() ),
                   repoDir = exampleRepoDir )

# removing an example Repository
```

```
  deleteRepo( exampleRepoDir )

  rm( exampleRepoDir )

  # many archivist-like Repositories on one Github repository

  searchInGithubRepo( pattern = "name", user="MarcinKosinski", repo="Museum",
  branch="master", repoDirGit="ex1", fixed = FALSE )

  searchInGithubRepo( pattern = "name", user="MarcinKosinski", repo="Museum",
                  branch="master", repoDirGit="ex2", fixed = FALSE )


## End(Not run)
```

`showLocalRepo` *View the Lisf of Artifacts from a Repository*

Description

`showLocalRepo` and `showGithubRepo` functions produce the `data.frame` of the artifacts from a Repository saved in a given `repoDir` (directory). `showLocalRepo` shows the artifacts from the `Repository` that exists on the user's computer, whereas `showGithubRepo` shows the artifacts of the `Repository` existing on a Github repository. To learn more about artifacts visit archivist-package.

Usage

```
showLocalRepo(repoDir, method = "md5hashes")

showGithubRepo(repo, user, branch = "master", repoDirGit = FALSE,
  method = "md5hashes")
```

Arguments

`method`	A character specifying the method to be used to show the Repository. Available methods: `md5hashes` (default), `tags`.
`repoDir`	A character denoting an existing directory of a Repository for which a summary will be returned.
`repo`	Only if working with a Github repository. A character containing a name of a Github repository on which the Repository is archived.
`user`	Only if working with a Github repository. A character containing a name of a Github user on whose account the `repo` is created.
`branch`	Only if working with a Github repository. A character containing a name of Github Repository's branch on which a Repository is archived. Default `branch` is `master`.

repoDirGit — Only if working with a Github repository. A character containing a name of a directory on Github repository on which the Repository is stored. If the Repository is stored in main folder on Github repository, this should be set to repoDirGit = FALSE as default.

Details

showLocalRepo and showGithubRepo functions produce the data.frame of the artifacts from a Repository saved in a given repoDir (directory). showLocalRepo shows the artifacts from the Repository that exists on the user's computer, whereas showGithubRepo shows the artifacts of the Repository existing on a Github repository.

Both functions show the current state of a Repository, inter alia, all archived artifacts can be seen with their unique md5hash or a data.frame with archived Tags can be obtained. Also there is an extra column with a date of creation the Tag or the md5hash.

Value

If parameter method was set as md5hashes a data.frame with artifacts' names and archived md5hashes is returned.

If parameter method was set as tags a data.frame with archived Tags and archived artifacts' md5hashes is returned.

To learn more about Tags or md5hashes check: Tags or md5hash.

Author(s)

Marcin Kosinski , <m.p.kosinski@gmail.com>

See Also

Other archivist: Repository; Tags; archivist-package; copyGithubRepo, copyLocalRepo; createEmptyRepo; deleteRepo; getTagsGithub, getTagsLocal; loadFromGithubRepo, loadFromLocalRepo; md5hash; rmFromRepo; saveToRepo; searchInGithubRepo, searchInLocalRepo; summaryGithubRepo, summaryLocalRepo; tarGithubRepo, tarLocalRepo

Examples

```
# objects preparation
## Not run:
# data.frame object
data(iris)

# ggplot/gg object
library(ggplot2)
df <- data.frame(gp = factor(rep(letters[1:3], each = 10)),y = rnorm(30))
library(plyr)
ds <- ddply(df, .(gp), summarise, mean = mean(y), sd = sd(y))
myplot123 <- ggplot(df, aes(x = gp, y = y)) +
  geom_point() +  geom_point(data = ds, aes(y = mean),
                colour = 'red', size = 3)
```

```
# lm object
model <- lm(Sepal.Length~ Sepal.Width + Petal.Length + Petal.Width, data= iris)


# lda object
library(MASS)

Iris <- data.frame(rbind(iris3[,,1], iris3[,,2], iris3[,,3]),
                   Sp = rep(c("s","c","v"), rep(50,3)))
train <- c(8,83,115,118,146,82,76,9,70,139,85,59,78,143,68,
           134,148,12,141,101,144,114,41,95,61,128,2,42,37,
           29,77,20,44,98,74,32,27,11,49,52,111,55,48,33,38,
           113,126,24,104,3,66,81,31,39,26,123,18,108,73,50,
           56,54,65,135,84,112,131,60,102,14,120,117,53,138,5)
lda1 <- lda(Sp ~ ., Iris, prior = c(1,1,1)/3, subset = train)

# qda object
tr <- c(7,38,47,43,20,37,44,22,46,49,50,19,4,32,12,29,27,34,2,1,17,13,3,35,36)
train <- rbind(iris3[tr,,1], iris3[tr,,2], iris3[tr,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
qda1 <- qda(train, cl)

# glmnet object
library( glmnet )

zk=matrix(rnorm(100*20),100,20)
bk=rnorm(100)
glmnet1=glmnet(zk,bk)

# creating example Repository - that examples will work

exampleRepoDir <- tempdir()
createEmptyRepo(repoDir = exampleRepoDir)
saveToRepo(myplot123, repoDir=exampleRepoDir)
saveToRepo(iris, repoDir=exampleRepoDir)
saveToRepo(model, repoDir=exampleRepoDir)

# show examples

showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)
showLocalRepo(method = "tags", repoDir = exampleRepoDir)

# let's add more artifacts

saveToRepo(glmnet1, repoDir=exampleRepoDir)
saveToRepo(lda1, repoDir=exampleRepoDir)
(qda1Md5hash <- saveToRepo(qda1, repoDir=exampleRepoDir))

# show now

showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)
showLocalRepo(method = "tags", repoDir = exampleRepoDir)
```

```
# what if we remove an artifact

rmFromRepo(qda1Md5hash, repoDir = exampleRepoDir)

# show now

showLocalRepo(method = "md5hashes", repoDir = exampleRepoDir)
showLocalRepo(method = "tags", repoDir = exampleRepoDir)

# GitHub version

showGithubRepo(method = "md5hashes", user = "pbiecek", repo = "archivist")
showGithubRepo(method = "tags", user = "pbiecek", repo = "archivist", branch = "master")

# removing an example Repository

deleteRepo( exampleRepoDir )

rm( exampleRepoDir )

# many archivist-like Repositories on one Github repository

showGithubRepo( user="MarcinKosinski", repo="Museum", branch="master",
repoDirGit="ex1")
showGithubRepo( user="MarcinKosinski", repo="Museum", branch="master",
                repoDirGit="ex2")


## End(Not run)
```

`summaryLocalRepo` *View the Summary of a Repository*

Description

`summaryRepo` summaries the current state of a Repository.

Usage

```
summaryLocalRepo(repoDir)

summaryGithubRepo(repo, user, branch = "master", repoDirGit = FALSE)
```

Arguments

`repoDir`	A character denoting an existing directory of a Repository for which a summary will be returned.
`repo`	Only if working with a Github repository. A character containing a name of a Github repository on which the Repository is archived.

user	Only if working with a Github repository. A character containing a name of a Github user on whose account the repo is created.
branch	Only if working with a Github repository. A character containing a name of Github Repository's branch on which a Repository is archived. Default branch is master.
repoDirGit	Only if working with a Github repository. A character containing a name of a directory on Github repository on which the Repository is stored. If the Repository is stored in main folder on Github repository, this should be set to repoDirGit = FALSE as default.

Details

summaryRepo summaries the current state of a Repository. Recommended to use print(summaryRepo)). See examples.

Value

An object of class repository which can be printed: print(object).

Note

If the same artifact was archived many times it is counted as one artifact or database in print(summaryRepo).

Author(s)

Marcin Kosinski , <m.p.kosinski@gmail.com>

See Also

Other archivist: Repository; Tags; archivist-package; copyGithubRepo, copyLocalRepo; createEmptyRepo; deleteRepo; getTagsGithub, getTagsLocal; loadFromGithubRepo, loadFromLocalRepo; md5hash; rmFromRepo; saveToRepo; searchInGithubRepo, searchInLocalRepo; showGithubRepo, showLocalRepo; tarGithubRepo, tarLocalRepo

Examples

```
# objects preparation
## Not run:
# data.frame object
data(iris)

# ggplot/gg object
library(ggplot2)
df <- data.frame(gp = factor(rep(letters[1:3], each = 10)),y = rnorm(30))
library(plyr)
ds <- ddply(df, .(gp), summarise, mean = mean(y), sd = sd(y))
myplot123 <- ggplot(df, aes(x = gp, y = y)) +
  geom_point() +  geom_point(data = ds, aes(y = mean),
                colour = 'red', size = 3)
```

```
# lm object
model <- lm(Sepal.Length~ Sepal.Width + Petal.Length + Petal.Width, data= iris)


# lda object
library(MASS)

Iris <- data.frame(rbind(iris3[,,1], iris3[,,2], iris3[,,3]),
                   Sp = rep(c("s","c","v"), rep(50,3)))
train <- c(8,83,115,118,146,82,76,9,70,139,85,59,78,143,68,
           134,148,12,141,101,144,114,41,95,61,128,2,42,37,
           29,77,20,44,98,74,32,27,11,49,52,111,55,48,33,38,
           113,126,24,104,3,66,81,31,39,26,123,18,108,73,50,
           56,54,65,135,84,112,131,60,102,14,120,117,53,138,5)
lda1 <- lda(Sp ~ ., Iris, prior = c(1,1,1)/3, subset = train)

# qda object
tr <- c(7,38,47,43,20,37,44,22,46,49,50,19,4,32,12,29,27,34,2,1,17,13,3,35,36)
train <- rbind(iris3[tr,,1], iris3[tr,,2], iris3[tr,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
qda1 <- qda(train, cl)

# glmnet object
library( glmnet )

zk=matrix(rnorm(100*20),100,20)
bk=rnorm(100)
glmnet1=glmnet(zk,bk)

# creating example Repository - that examples will work

exampleRepoDir <- tempdir()
createEmptyRepo(repoDir = exampleRepoDir)
saveToRepo(myplot123, repoDir=exampleRepoDir)
saveToRepo(iris, repoDir=exampleRepoDir)
saveToRepo(model, repoDir=exampleRepoDir)

# summary examples

summaryLocalRepo( repoDir = exampleRepoDir )

# let's add more artifacts

saveToRepo(glmnet1, repoDir=exampleRepoDir)
saveToRepo(lda1, repoDir=exampleRepoDir)
(qda1Md5hash <- saveToRepo(qda1, repoDir=exampleRepoDir))

# summary now

summaryLocalRepo( repoDir = exampleRepoDir )

# what if we remove an artifact
```

```
rmFromRepo(qda1Md5hash, repoDir = exampleRepoDir)

# summary now

summaryLocalRepo( repoDir = exampleRepoDir )

#
# Github version
#

x <- summaryGithubRepo( user="pbiecek", repo="archivist")
print( x )

# removing an example Repository

deleteRepo( exampleRepoDir )

rm( exampleRepoDir )

# many archivist-like Repositories on one Github repository

summaryGithubRepo(user="MarcinKosinski", repo="Museum",
branch="master", repoDirGit="ex2" )


## End(Not run)
```

`Tags` *Tags*

Description

`Tags` are attributes of an artifact, i.e., a class, a name, names of artifact's parts, etc.. The list of artifact tags vary across artifact's classes. To learn more about artifacts visit archivist-package.

Details

`Tags` are attributes of an artifact. `Tags` can be an artifact's `name`, `class` or `archiving date`. Furthermore, for various artifact's classes more different `Tags` are available and can be searched by searchInLocalRepo or searchInGithubRepo functions.

`Tags` are stored in the Repository. If data is extracted from artifact a special `Tag` named `relationWith` is created, and specifies with which artifact this data is related to.

So far supported artifact list is presented below. Objects are divided thematically. The newest list is also available on **archivist** `wiki` on Github[7].

Regression Models

`lm` • coefname

[7] `https://github.com/pbiecek/archivist/wiki/archivist-package---Tags`

- class
- name
- date

`glmnet`
- date
- name
- class

`survfit`
- date
- name
- class
- strata
- type

Plots

`ggplot`
- name
- class
- date
- labelx
- labely

`trellis`
- date
- name
- class

Results of Agglomeration Methods

`twins which is a result of agnes, diana or mona functions`
- date
- name
- class
- ac

`partition which is a result of pam, clara or fanny functions`
- date
- name
- class
- objective

`lda`
- date
- name
- class

`qda`
- date
- name
- class
- terms

Statistical Tests

`htest`
- alternative

- method
- date
- name
- class

When non of above is specified, tags are corresponded by default

`default`
- name
- class
- date

`data.frame`
- name
- class
- date
- varname

Note

One can specify his own `Tags` for artifacts by setting artifact's attribute before call of the `saveToRepo` function like this: `attr(x, "tags" ) = c( "name1", "name2" )`, where `x` is artifact and `name1, name2` are `Tags` specified by an user.

See Also

Functions using `Tags` are:

- searchInLocalRepo,
- searchInGithubRepo.

Other archivist: `Repository`; `archivist-package`; `copyGithubRepo`, `copyLocalRepo`; `createEmptyRepo`; `deleteRepo`; `getTagsGithub`, `getTagsLocal`; `loadFromGithubRepo`, `loadFromLocalRepo`; `md5hash`; `rmFromRepo`; `saveToRepo`; `searchInGithubRepo`, `searchInLocalRepo`; `showGithubRepo`, `showLocalRepo`; `summaryGithubRepo`, `summaryLocalRepo`; `tarGithubRepo`, `tarLocalRepo`

Examples

```
# examples
## Not run:
# data.frame object
data(iris)
exampleRepoDir <- tempdir()
createEmptyRepo(repoDir = exampleRepoDir)
saveToRepo( iris, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteRepo( exampleRepoDir )

# ggplot/gg object
library(ggplot2)
df <- data.frame(gp = factor(rep(letters[1:3], each = 10)),y = rnorm(30))
library(plyr)
```

```
ds <- ddply(df, .(gp), summarise, mean = mean(y), sd = sd(y))
myplot123 <- ggplot(df, aes(x = gp, y = y)) +
  geom_point() +  geom_point(data = ds, aes(y = mean),
                             colour = 'red', size = 3)
exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )
saveToRepo( myplot123, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteRepo( exampleRepoDir )

# lm object
model <- lm(Sepal.Length~ Sepal.Width + Petal.Length + Petal.Width,
           data= iris)
exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )
saveToRepo( model, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteRepo( exampleRepoDir )

# agnes (twins) object
library(cluster)
data(votes.repub)
agn1 <- agnes(votes.repub, metric = "manhattan", stand = TRUE)
exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )
saveToRepo( agn1, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteRepo( exampleRepoDir )

# fanny (partition) object
x <- rbind(cbind(rnorm(10, 0, 0.5), rnorm(10, 0, 0.5)),
          cbind(rnorm(15, 5, 0.5), rnorm(15, 5, 0.5)),
          cbind(rnorm( 3,3.2,0.5), rnorm( 3,3.2,0.5)))
fannyx <- fanny(x, 2)
exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )
saveToRepo( fannyx, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteRepo( exampleRepoDir )

# lda object
library(MASS)

Iris <- data.frame(rbind(iris3[,,1], iris3[,,2], iris3[,,3]),
                   Sp = rep(c("s","c","v"), rep(50,3)))
train <- c(8,83,115,118,146,82,76,9,70,139,85,59,78,143,68,
           134,148,12,141,101,144,114,41,95,61,128,2,42,37,
           29,77,20,44,98,74,32,27,11,49,52,111,55,48,33,38,
           113,126,24,104,3,66,81,31,39,26,123,18,108,73,50,
           56,54,65,135,84,112,131,60,102,14,120,117,53,138,5)
lda1 <- lda(Sp ~ ., Iris, prior = c(1,1,1)/3, subset = train)
exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )
```

```
saveToRepo( lda1, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteRepo( exampleRepoDir )

# qda object
tr <- c(7,38,47,43,20,37,44,22,46,49,50,19,4,32,12,29,27,34,2,1,17,13,3,35,36)
train <- rbind(iris3[tr,,1], iris3[tr,,2], iris3[tr,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
qda1 <- qda(train, cl)
exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )
saveToRepo( qda1, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteRepo( exampleRepoDir )


# glmnet object
library( glmnet )

zk=matrix(rnorm(100*20),100,20)
bk=rnorm(100)
glmnet1=glmnet(zk,bk)
exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )
saveToRepo( glmnet1, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteRepo( exampleRepoDir )

# trellis object
require(stats)
library( lattice)
## Tonga Trench Earthquakes

Depth <- equal.count(quakes$depth, number=8, overlap=.1)
xyplot(lat ~ long | Depth, data = quakes)
update(trellis.last.object(),
       strip = strip.custom(strip.names = TRUE, strip.levels = TRUE),
       par.strip.text = list(cex = 0.75),
       aspect = "iso")

## Examples with data from `Visualizing Data' (Cleveland, 1993) obtained
## from http://cm.bell-labs.com/cm/ms/departments/sia/wsc/

EE <- equal.count(ethanol$E, number=9, overlap=1/4)

## Constructing panel functions on the fly; prepanel
trellis.plot <- xyplot(NOx ~ C | EE, data = ethanol,
                       prepanel = function(x, y) prepanel.loess(x, y, span = 1),
                       xlab = "Compression Ratio", ylab = "NOx (micrograms/J)",
                       panel = function(x, y) {
                         panel.grid(h = -1, v = 2)
                         panel.xyplot(x, y)
                         panel.loess(x, y, span=1)
```

```
                        },
                        aspect = "xy")
exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )
saveToRepo( trellis.plot, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteRepo( exampleRepoDir )

# htest object

x <- c(1.83,  0.50,  1.62,  2.48, 1.68, 1.88, 1.55, 3.06, 1.30)
y <- c(0.878, 0.647, 0.598, 2.05, 1.06, 1.29, 1.06, 3.14, 1.29)
this.test <- wilcox.test(x, y, paired = TRUE, alternative = "greater")
exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )
saveToRepo( this.test, repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )
deleteRepo( exampleRepoDir )

# survfit object
library( survival )
# Create the simplest test data set
test1 <- list(time=c(4,3,1,1,2,2,3),
              status=c(1,1,1,0,1,1,0),
             x=c(0,2,1,1,1,0,0),
             sex=c(0,0,0,0,1,1,1))
# Fit a stratified model
myFit <-  survfit( coxph(Surv(time, status) ~ x + strata(sex), test1), data = test1  )
exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )
saveToRepo( myFit , repoDir=exampleRepoDir )
showLocalRepo( exampleRepoDir, "tags" )[,-3]
deleteRepo( exampleRepoDir )

# origin of the artifacts stored as a name - chaining code
library(dplyr)
exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )
data("hflights", package = "hflights")
hflights %>%
  group_by(Year, Month, DayofMonth) %>%
  select(Year:DayofMonth, ArrDelay, DepDelay) %>%
  saveToRepo( exampleRepoDir, chain = TRUE ) %>%
  # here the artifact is stored but chaining is not finished
  summarise(
    arr = mean(ArrDelay, na.rm = TRUE),
    dep = mean(DepDelay, na.rm = TRUE)
  ) %>%
  filter(arr > 30 | dep > 30) %>%
  saveToRepo( exampleRepoDir )
  # chaining code is finished and after last operation the
  # artifact is stored
showLocalRepo( exampleRepoDir, "tags" )[,-3]
```

```
showLocalRepo( exampleRepoDir )
deleteRepo( exampleRepoDir )

rm( exampleRepoDir )

## End(Not run)
```

`tarLocalRepo` *Create a Tar Archive From an Existing Repository*

Description

`tarLocalRepo` and `tarGithubRepo` create a tar archive from an existing Repository.

Usage

```
tarLocalRepo(repoDir)

tarGithubRepo(repoTo, user, repo, branch = "master", repoDirGit = FALSE)
```

Arguments

`repoDir`	A character that specifies the directory of the Repository which will be tarred.
`repo`	Only if working with a Github repository. A character containing a name of a Github repository on which the Repository to be tarred is archived.
`user`	Only if working with a Github repository. A character containing a name of a Github user on whose account the `repo` is created.
`branch`	Only if working with a Github repository. A character containing a name of Github repository's branch in which Repository to be tarred is archived. Default `branch` is `master`.
`repoDirGit`	Only if working with a Github repository. A character containing a name of a directory on Github repository on which the Repository to be tarred is stored. If the Repository is stored in main folder on Github repository, this should be set to `repoDirGit = FALSE` as default.
`repoTo`	Only if working with a Github repository. A characterthat specifies the directory in which will be created tar archive from `Repository` stored on Github.

Details

`tarLocalRepo` and `tarGithubRepo` create a tar archive from an existing Repository. `tarLocalRepo` tars local `Repository`, `tarGithubRepo` tars `Repository` stored on Github.

Author(s)

Marcin Kosinski, <m.p.kosinski@gmail.com>

See Also

Other archivist: Repository; Tags; archivist-package; copyGithubRepo, copyLocalRepo; createEmptyRepo; deleteRepo; getTagsGithub, getTagsLocal; loadFromGithubRepo, loadFromLocalRepo; md5hash; rmFromRepo; saveToRepo; searchInGithubRepo, searchInLocalRepo; showGithubRepo, showLocalRepo; summaryGithubRepo, summaryLocalRepo

Other archivist: Repository; Tags; archivist-package; copyGithubRepo, copyLocalRepo; createEmptyRepo; deleteRepo; getTagsGithub, getTagsLocal; loadFromGithubRepo, loadFromLocalRepo; md5hash; rmFromRepo; saveToRepo; searchInGithubRepo, searchInLocalRepo; showGithubRepo, showLocalRepo; summaryGithubRepo, summaryLocalRepo

Examples

```
# objects preparation
## Not run:
# data.frame object
data(iris)

# ggplot/gg object
library(ggplot2)
df <- data.frame(gp = factor(rep(letters[1:3], each = 10)),y = rnorm(30))
library(plyr)
ds <- ddply(df, .(gp), summarise, mean = mean(y), sd = sd(y))
myplot123 <- ggplot(df, aes(x = gp, y = y)) +
  geom_point() +  geom_point(data = ds, aes(y = mean),
                colour = 'red', size = 3)

# lm object
model <- lm(Sepal.Length~ Sepal.Width + Petal.Length + Petal.Width, data= iris)

# Local version

exampleRepoDir <- tempdir()
createEmptyRepo( repoDir = exampleRepoDir )
saveToRepo( myplot123, repoDir=exampleRepoDir )
saveToRepo( iris, repoDir=exampleRepoDir )
saveToRepo( model, repoDir=exampleRepoDir )

tarLocalRepo( exampleRepoDir )

deleteRepo( exampleRepoDir )

rm( exampleRepoDir )

# Github version

tarGithubRepo( user="MarcinKosinski",
repo="Museum", branch="master", repoDirGit="ex1" )

tarGithubRepo( user="pbiecek, repo="archivist, repoTo = getwd( ) )
```

```
## End(Not run)
```